

**E**UROPEAN  
**T**ELECOMMUNICATION  
**S**TANDARD

**ETS 300 709**

September 1996

---

Source: ETSI TC-TE

Reference: DE/TE-01067

ICS: 33.020

**Key words:** videotex, VEMMI, multimedia, retrieval services

**Terminal Equipment (TE);  
Enhanced Man Machine Interface service for Videotex and  
Multimedia/Hypermedia retrieval services**

**ETSI**

European Telecommunications Standards Institute

**ETSI Secretariat**

**Postal address:** F-06921 Sophia Antipolis CEDEX - FRANCE

**Office address:** 650 Route des Lucioles - Sophia Antipolis - Valbonne - FRANCE

**X.400:** c=fr, a=atlas, p=etsi, s=secretariat - **Internet:** secretariat@etsi.fr

Tel.: +33 92 94 42 00 - Fax: +33 93 65 47 16

---

**Copyright Notification:** No part may be reproduced except as authorized by written permission. The copyright and the foregoing restriction extend to reproduction in all media.

© European Telecommunications Standards Institute 1996. All rights reserved.



## Contents

Foreword .....	7
1 Scope .....	9
2 Normative references .....	9
3 Definitions and abbreviations .....	10
3.1 Definitions .....	10
3.2 Abbreviations .....	11
4 General model .....	12
4.1 Introduction .....	12
4.2 Definition of the VEMMI elements .....	13
4.2.1 VEMMI object definition and identification .....	13
4.2.2 VEMMI component definition .....	13
4.2.3 VEMMI component item definition .....	14
4.2.4 Resource definition .....	14
4.2.5 Transfer of VEMMI objects .....	14
4.2.6 Resource-file transfer .....	14
4.3 VEMMI plane structure model .....	15
4.3.1 The standard plane for Videotex data .....	15
4.3.2 The VEMMI objects plane .....	16
4.4 Operation modes for VEMMI terminals .....	16
4.4.1 The standard mode .....	16
4.4.2 The VEMMI mode .....	16
4.4.3 Switching between standard mode and VEMMI mode .....	16
4.5 VEMMI elements data content .....	17
4.5.1 Text data definition .....	17
4.5.2 Bitmaps .....	18
4.5.3 Colour .....	19
4.5.4 Videotex data .....	19
4.5.5 Sound .....	19
4.5.6 Graphical data .....	19
4.5.7 Video data .....	19
4.6 VEMMI objects positioning and dimensioning .....	20
4.6.1 Positioning .....	20
4.6.2 Dimensioning .....	21
4.7 VEMMI elements states and state parameters .....	21
4.7.1 Object .....	21
4.7.1.1 Definition of object states .....	22
4.7.1.2 Definition of object state parameters .....	23
4.7.2 Component .....	23
4.7.2.1 Definition of component states .....	23
4.7.2.2 Definition of component state parameters .....	24
4.8 Local action management .....	24
4.9 Memory considerations .....	25
4.10 Common rules for object handling .....	26
4.10.1 Active state and focus management .....	26
4.10.2 Behaviour of the modal mode .....	26
4.10.3 Size considerations and clipping .....	26
4.11 Local object storage .....	26
4.12 Symbolic directory names .....	27
4.13 Specific rules for dedicated terminals and PDAs .....	28
4.14 VEMMI local storage interchange format .....	28
5 Service description .....	29
5.1 Service elements initiated by the VEMMI application .....	30

5.1.1	VEMMI_Open .....	31
5.1.2	VEMMI_Close .....	31
5.1.3	VEMMI_Resume .....	31
5.1.4	VEMMI_Suspend .....	32
5.1.5	VEMMI_Identify_Term_Cap .....	32
5.1.6	VEMMI_Set_Options .....	32
5.1.7	VEMMI_Create_Object .....	32
5.1.8	VEMMI_Open_Object .....	33
5.1.9	VEMMI_Open_Blocking_Object .....	33
5.1.10	VEMMI_Close_Object .....	33
5.1.11	VEMMI_Close_All .....	34
5.1.12	VEMMI_Destroy_Object .....	34
5.1.13	VEMMI_Obj_Access_Disable .....	34
5.1.14	VEMMI_Obj_Access_Enable .....	34
5.1.15	VEMMI_Modify_Component .....	35
5.1.16	VEMMI_Obj_Location_Change .....	35
5.1.17	VEMMI_Load_Col_Table .....	35
5.1.18	VEMMI_Reset_Col_Table .....	36
5.1.19	VEMMI_Open_Application .....	36
5.1.20	VEMMI_Delete_Outdated_Objects .....	37
5.1.21	VEMMI_Store_Objects .....	37
5.1.22	VEMMI_Erase_Objects .....	37
5.1.23	VEMMI_User_Lock .....	37
5.1.24	VEMMI_User_Unlock .....	38
5.1.25	VEMMI_Resource-file_Transfer .....	38
5.2	Service elements initiated by the terminal .....	38
5.2.1	VEMMI_Identify_Term_Cap_Resp .....	38
5.2.2	VEMMI_Object_Retransmission .....	39
5.2.3	VEMMI_User_Data .....	40
5.2.4	VEMMI_Open_Application_Resp .....	41
5.2.5	VEMMI_Store_Objects_Resp .....	41
5.2.6	VEMMI_Error .....	41
5.2.7	VEMMI_Transfer_Acknowledge .....	42
6	VEMMI objects introduction .....	42
6.1	The Application Bar .....	42
6.1.1	Composition .....	43
6.2	The Button Bar .....	43
6.2.1	Composition .....	43
6.3	The Pop-Up Menu .....	43
6.3.1	Composition .....	43
6.4	The Dialogue Box .....	43
6.4.1	Composition .....	44
6.4.1.1	The Separator component .....	44
6.4.1.2	The Frame component .....	44
6.4.1.3	The Text Presentation Area component .....	44
6.4.1.4	The Text component .....	44
6.4.1.5	The Graphic Output Area component .....	44
6.4.1.6	The Sensitive Text component .....	44
6.4.1.7	The Push Button component .....	44
6.4.1.8	The Text Input Field component .....	45
6.4.1.9	The Check Box component .....	45
6.4.1.10	The Radio Button component .....	45
6.4.1.11	The List Box component .....	45
6.4.1.12	The Combination Box component .....	45
6.4.1.13	The Slider Component .....	45
6.4.1.14	The Sensitive Area component .....	45
6.4.1.15	The Multimedia Area component .....	45
6.5	Operative Object .....	45
6.6	Bitmap resource object .....	45
6.7	Videotex Resource Object .....	46
6.8	Text Resource Object .....	46
6.9	Font resource object .....	46

6.10	Metacode object.....	46
6.11	The Message Box.....	46
6.12	Sound Object.....	46
6.13	Video Object.....	46
6.14	Multimedia Resource Object.....	46
7	Functional description.....	46
7.1	General rules for the behaviour of elements.....	46
7.1.1	User interaction.....	46
7.1.2	Local actions and reports.....	46
7.1.3	Relationship between objects and components.....	48
7.1.4	Open/Close of Sound, Video, Resource and Metacode objects.....	48
7.1.5	Maximize operation.....	49
7.1.6	Notational conventions.....	49
7.1.7	Mnemonic.....	50
7.1.8	Activation and Validation.....	50
7.2	Text formats.....	50
7.2.1	VEMMI high quality text.....	50
7.2.1.1	Text attributes.....	50
7.2.1.2	In-text attributes for the definition of sensitive text.....	51
7.2.2	Text labels and titles.....	51
7.3	The Application Bar.....	51
7.3.1	Composition.....	53
7.3.1.1	Menu Choice components of the bar.....	53
7.3.1.2	Menu Choice components of the Pull-Down Menu.....	54
7.3.1.3	Menu Choice components of the Cascading Menu.....	56
7.4	The Button Bar.....	57
7.4.1	Composition.....	58
7.4.1.1	The Button component.....	58
7.5	The Pop-Up Menu.....	59
7.5.1	Composition.....	60
7.5.1.1	Menu Choice components of the Primary Pop-Up Menu.....	60
7.5.1.2	Menu Choice components of the Cascading Menu.....	61
7.6	The Dialogue Box.....	62
7.6.1	Composition.....	64
7.6.1.1	The Separator component.....	64
7.6.1.2	The Frame component.....	65
7.6.1.3	The Text Presentation Area component.....	65
7.6.1.4	The Text component.....	67
7.6.1.5	The Sensitive Text Component.....	71
7.6.1.6	The Graphic Output Area component.....	71
7.6.1.7	The Push Button component.....	73
7.6.1.8	The Text Input Field component.....	74
7.6.1.9	The Check Box component.....	75
7.6.1.10	The Radio Button component.....	76
7.6.1.11	The List Box component.....	78
7.6.1.12	The Combination Box component.....	80
7.6.1.13	The Slider Component.....	82
7.6.1.14	The Sensitive Area Component.....	83
7.6.1.15	The Multimedia Area component.....	84
7.7	The Message Box.....	86
7.8	Operative object.....	88
7.8.1	Requirements for standalone programs.....	88
7.8.2	Requirements for programs with filter interface.....	89
7.9	Bitmap Resource Object.....	90
7.10	Videotex Resource Object.....	91
7.11	Text Resource Object.....	91
7.12	Font resource object.....	91
7.13	Metacode object.....	92
7.14	Sound Object.....	92
7.15	Video Object.....	92
7.16	Multimedia Resource Object.....	92
7.17	The VEMMI content encoding identification catalogue.....	93

8	Complete coded representation of the VEMMI.....	93
8.1	Introduction.....	93
8.2	Notation used .....	93
8.3	Overall switching of coding environment.....	94
8.3.1	Switching into the VEMMI mode.....	96
8.3.2	ISO/IEC 9281 syntax structure .....	96
8.4	VEMMI Command Syntax.....	98
8.5	Objects, components .....	101
8.6	Local actions .....	105
9	Encoding.....	105
9.1	Command structure.....	105
9.2	Object, component and attribute structure .....	106
9.3	Terminal symbols encoding .....	107
9.3.1	Opcodes .....	107
9.3.2	Integers.....	107
9.3.3	Enumerated .....	107
9.3.4	Strings.....	108
9.3.5	NDC .....	108
9.4	Attributes and lower level symbols .....	109
9.5	Opcodes .....	112
9.6	Syntax of the VEMMI_Modify_Component .....	115
9.7	Defaults .....	118
10	Introduction of the VEMMI service into existing Videotex ETSs.....	119
10.1	Introduction of the VEMMI to ETS 300 072.....	119
10.2	Introduction of the VEMMI to ETS 300 223 and ETS 300 079.....	119
	Annex A (normative): CCITT Recommendation T.51 [2] String .....	120
A.1	Scope.....	120
A.2	Graphic character sets .....	120
A.3	Code extension technique .....	123
A.4	Repertoire of the latin-based character set .....	123
A.5	Control functions.....	123
	Annex B (normative): Mandatory subset of ISO 8859 .....	124
	Annex C (normative): Minimum datatype kernel.....	125
	History .....	126

## Foreword

This European Telecommunication Standard (ETS) has been produced by the Terminal Equipment (TE) Technical Committee of the European Telecommunications Standards Institute (ETSI).

Transposition dates	
Date of adoption of this ETS (doa):	30 September 1996
Date of latest announcement of this ETS (doa):	31 December 1996
Date of latest publication of new National Standard or endorsement of this ETS (dop/e):	30 June 1997
Date of withdrawal of any conflicting National Standard (dow):	30 June 1997

Blank page



## 1 Scope

This European Telecommunication Standard (ETS) specifies the data syntax to be used by Videotex and Multimedia/Hypermedia Information retrieval services for implementation of the Videotex Enhanced Man Machine Interface (VEMMI).

In the Videotex case this ETS is applicable to both the Videotex service and the attached Videotex terminals. Those terminals may be connected to the Videotex service via the Public Switched Telephone Network (PSTN), Integrated Services Digital Network (ISDN) or Packet Switched Public Data Network (PSPDN). Typically, the terminals should support ISDN Syntax-Based Videotex (SBV).

The ETS can also be used for any kind of retrieval service (not related to Videotex) by using the relevant underlying platform and content data types.

## 2 Normative references

This ETS incorporates by dated and undated reference, provisions from other publications. These normative references are cited at the appropriate places in the text and the publications are listed hereafter. For dated references, subsequent amendments to or revisions of any of these publications apply to this ETS only when incorporated in it by amendment or revision. For undated references the latest edition of the publication referred to applies.

- [1] CCITT Recommendation T.50 (1992): "International Reference Alphabet (IRA) (Formerly International Alphabet No. 5 or IA5) - Information technology - 7-bit coded character set for information interchange".
- [2] CCITT Recommendation T.51 (1992): "Latin based coded character sets for telematic services".
- [3] CCITT Recommendation T.52 (1993): "Non-Latin coded character sets for telematic services".
- [4] ITU-T Recommendation T.101 (1994): "International interworking for videotex services".
- [5] ITU-T Recommendation T.102 (1993): "Syntax-based videotex end-to-end protocols for circuit mode ISDN".
- [6] ITU-T Recommendation T.105 (1994): "Syntax-based videotex application layer protocol".
- [7] ITU-T Recommendation H.261 (1994): "Video codec for audiovisual services at p x 64 kbit/s".
- [8] ITU-TS Recommendation H.320 (1994): "Narrow-band visual telephone systems and terminal equipment".
- [9] ITU-T Recommendation F.300 (1994): "Videotex service".
- [10] ISO 2022 (1986): "Information technology - ISO 7-bit and 8-bit coded character sets - Code extension techniques".
- [11] ISO 2375 (1985): "Data processing - Procedure for registration of escape sequences".
- [12] ISO/IEC 8632 (1992): "Information technology - Computer graphics - Metafile for storage and transfer of picture description information".
- [13] ISO 8859 (1987): "Information Processing - 8-bit single byte coded graphic character set".
- [14] ISO/IEC 9281 (1990): "Information technology - Picture coding methods".

- [15] ISO/IEC 10646-1 (1993): "Information technology - Universal Multiple-Octet Coded Character Set (UCS); Part 1: Architecture and Basic Multilingual Plane".
- [16] ISO/IEC 10918-1 (1994): "Information technology - Digital compression and coding of continuous-tone still images: Requirements and guidelines".
- [17] ISO/IEC 11172-1 (1993): "Information technology - Coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mbit/s - Part 1: Systems".
- [18] ISO/IEC 11172-2 (1993): "Information technology - Coding of moving pictures and associated audio for digital storage media at up to about 1.5 Mbit/s - Part 2: Video".
- [19] ISO 639 (1988): "Codes for the representation of names of languages".
- [20] ETS 300 072 (1990): "Terminal Equipment (TE); Videotex presentation layer protocol, Videotex presentation layer data syntax".
- [21] ETS 300 073: "Videotex presentation layer data syntax; Geometric Display (CEPT Recommendation T/TE 06-02, Edinburgh 1988)".
- [22] ETS 300 076 (1994): "Terminal Equipment (TE); Videotex, Terminal Facility Identifier (TFI)".
- [23] ETS 300 079 (1991): "Integrated Services Digital Network (ISDN); Syntax-based Videotex, End-to-end protocols, circuit mode DTE-DTE".
- [24] ETS 300 149 (1992): "Terminal Equipment (TE); Videotex, Audio syntax".
- [25] ETS 300 177 (1995): "Terminal Equipment (TE); Videotex, Photographic syntax".
- [26] ETS 300 223 (1993): "Terminal Equipment (TE); Syntax-based Videotex, Common end-to-end protocols".
- [27] ISO/IEC DIS 13818-1: "Information technology - Generic coding of moving pictures and associated audio information; Part 1: Systems".
- [28] ISO/IEC DIS 13818-2: Information technology - Generic coding of moving pictures and associated audio information; Part 2: Video".

### 3 Definitions and abbreviations

#### 3.1 Definitions

For the purposes of this ETS, the following definitions apply:

**controls:** Visual user-interface elements that allow a user to interact with data.

**dedicated terminal:** Designed to support VEMMI but using a different platform than a multipurpose personal computer. Such terminals may have processing, storage and presentation limitations.

**Defined Display Area (DDA):** See ITU-T Recommendation F.300 [9].

**emphasis:** Highlighting, colour change or other visible indication of the condition of an element or choice and the effect of that condition on a user's ability to interact with that element. Emphasis can also give additional information about the state of an object. The method used to emphasize an element is terminal dependent.

**label:** Text data associated with a VEMMI component to inform the user of the purpose of a particular component or item.

**local manager:** See VEMMI local manager.

**mnemonic:** A single, easy-to-remember alphanumeric character that activates a VEMMI Menu Choice component and validates it. A Mnemonic character can also be used to validate an active Push Button.

**modal mode:** When a VEMMI object is "modal" it is not possible for the user to leave this VEMMI object to the benefit of another VEMMI object of the same application with different possible access tools. Each attempt to access another object by the user is refused and possibly indicated by a sound signal.

**resource file transfer:** Mechanism to transfer files referenced by VEMMI resource objects from a VEMMI application to a VEMMI terminal.

**stretched presentation:** Reduced or enlarged display of a bitmap in order to meet given space requirements.

**tiled presentation:** Repeated display of a given bitmap in a horizontal and/or vertical direction in order to meet given space requirements.

**videotex application:** Videotex application using encoded data, protocols and profiles, as defined in the Videotex ETSs referenced in clause 2. A Videotex application does not use a VEMMI service, data and protocols (see ITU-T Recommendation F.300 [9]).

**videotex data:** Data interchanged between a Videotex application and a Videotex terminal.

**VEMMI application:** Application offering an enhanced man/machine interface as described in this ETS.

**VEMMI data:** VEMMI objects description and contents and VEMMI commands exchanged between the VEMMI application and the VEMMI terminal.

**VEMMI local manager:** Software running in the VEMMI terminal to handle and to present the VEMMI objects that are sent to the user by the VEMMI application.

**VEMMI terminal:** A terminal which is able to run a VEMMI local manager.

**Videotex host computer:** See ITU-T Recommendation F.300 [9].

**Videotex terminal:** See ITU-T Recommendation F.300 [9].

### 3.2 Abbreviations

For the purposes of this ETS, the following abbreviations apply:

BIN	Bitmap Identification Number
BMP	Microsoft Windows Device-Independent Bitmap
CD-ROM	Compact Disk-Read Only Memory
CGM	Computer Graphics Metafile
CIN	Component Identification Number
CMI	Coding Method Identifier
CR	Carriage Return
DDA	Defined Display Area
DE	Data Entity
DIB	Device-Independent Bitmap
DLL	Dynamic Link Libraries
DRCS	Dynamically Redefinable Character Set
DS I	Data Syntax according to ITU-T Recommendation T.101 [4], annex B
DS II	Data Syntax according to ITU-T Recommendation T.101 [4], annex C
DS III	Data Syntax according to ITU-T Recommendation T.101 [4], annex D
ESC	Escape
FIN	Font Identification Number
G0	Primary character set of CCITT Recommendation T.51 [2]
G2	Supplementary character set of CCITT Recommendation T.51 [2]
GIF	Graphics Interchange Format

GMT	Greenwich Mean Time
GUI	Graphical User Interface
HTML	HyperText Markup Language
HTTP	HyperText Transfer Protocol
IP	Internet Protocol
IRV	International Reference Version
ISDN	Integrated Services Digital Network
JIS	Japanese Institute for Standardization
JPEG	Joint Photographic Experts Groups
LF	Line Feed
LI	Length Indicator
MDI	More Data Indicator
MIDI	Musical Instrument Digital Interface
MIN	Multimedia Identification Number
MPEG	Moving Picture Experts Group
NDC	Normalized Device Co-ordinate
OIN	Object Identification Number
PCD	Picture Code Delimiter
PCE	Picture Control Entity
PDA	Personal Digital Assistant
PDE	Picture Data Entity
PE	Picture Element
PI	Picture Identifier
PM	Picture Mode
PSPDN	Packet Switched Public Data Network
PSTN	Public Switched Telephone Network
RGB	Red Green Blue
SBV	Syntax-Based Videotex
TCP	Transmission Control Protocol
TE	Terminal Equipment
TFI	Terminal Facility Identifier
TIN	Text Identification Number
TLV	Type Length Value
TV	Television
UI	User Interface
VEMMI	Videotex Enhanced Man Machine Interface
VIF	VEMMI Interchange Format
VIN	Videotex Identification Number
VPDE	Videotex Presentation Data Element
VTX	Videotex

## **4 General model**

### **4.1 Introduction**

Between a host and a VEMMI terminal a VEMMI service handles:

- general VEMMI objects as described in this ETS;
- data contents as defined in this ETS;
- data contents as referenced in this ETS.

A VEMMI terminal may also handle a Videotex application using encoded data and protocols as described in the Videotex ETSs referenced in clause 2.

## 4.2 Definition of the VEMMI elements

The logical units which form the structure of the VEMMI shall be named and defined as follows:

- VEMMI objects;
- VEMMI components or components;
- VEMMI component items or items.

VEMMI element is a generic name used in this ETS to designate an object, a component or an item.

An example is given in figure 1.

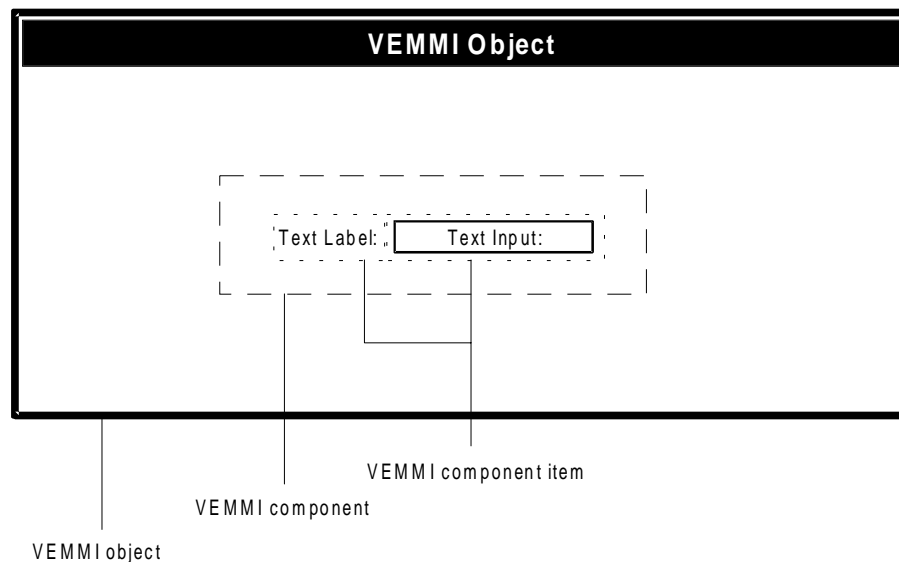


Figure 1: Example showing objects/components/items

### 4.2.1 VEMMI object definition and identification

The following four different types of VEMMI objects are defined in this ETS:

- display objects;
- operative objects;
- resource objects;
- metacode objects.

If not stated otherwise the term object used alone always refers to a display object.

VEMMI objects are the logical units which are used by a VEMMI application to interact with the user.

VEMMI objects can be composed of different components.

The objects are only defined regarding their functionality, their size and position relative to the Defined Display Area (DDA). The representation of the objects is terminal dependent.

Every object shall be identified by an Object Identification Number (OIN) which shall be unique within a VEMMI application at any one time.

### 4.2.2 VEMMI component definition

VEMMI components always belong to a VEMMI object and are only valid within this object. The object, to which a component belongs, is named the parent object.

In order to transport information, components may carry a data content (see subclause 4.5).

The components are only defined regarding their functionality, their type of content and their size and position relative to the object. The representation of the components is terminal dependent. The representation of data content is specified either by this ETS (for the datatypes that are defined within this document) or by the corresponding presentation standard (for datatypes that are defined outside this standard).

Every component shall be identified by a Component Identification Number (CIN) which shall be unique within an object.

#### **4.2.3 VEMMI component item definition**

The sub-unit of a VEMMI component is a component item. Every item is an integral part of a component. The definition of a component item is only valid within this component.

#### **4.2.4 Resource definition**

Resources are elements which can be referenced by components or objects. One resource can be referenced by more than one element. The following resources are defined:

- the colour table which is unique in one application;
- files stored in the terminal (identified by filenames) can contain sound data, operative objects etc.;
- a combination of a font family name and a set of attributes is a resource object. It is identified via a Font Identification Number (FIN);
- a bitmap can be a resource object. It is identified via a Bitmap Identification Number (BIN);
- text can be a resource object. It is identified via a Text Identification Number (TIN);
- Videotex can be a resource object. It is identified via a Videotex Identification Number (VIN);
- Multimedia can be a resource object. It is identified via a Multimedia Identification Number (MIN);
- objects sets stored in the terminal between two sessions (they are identified via attributes).

NOTE: FIN, BIN, VIN, TIN, MIN are OINs. The terms FIN, BIN, MIN and VIN are only used to clearly indicate that the corresponding object is a resource object.

#### **4.2.5 Transfer of VEMMI objects**

VEMMI objects can be transmitted to the VEMMI terminal using a telecommunication network. If they are stored then in the terminal they become local objects. VEMMI objects can also be downloaded using any suitable file transfer. They become local objects as well and are treated in the same way as objects transferred within the VEMMI dialogue and stored by the corresponding service primitive. VEMMI objects may also be transferred to the VEMMI terminal by postal mail (CD-ROM, diskettes etc.).

#### **4.2.6 Resource-file transfer**

VEMMI resource objects can reference files that contain the resource display data. These files are called resource-files.

VEMMI specifies the way the resource-files are transmitted to the terminal (VEMMI resource-file transfer). VEMMI resources data files may also be transferred to the VEMMI terminal using the file transfer used in the standard service in which the terminal operates or by postal mail (CD-ROM, diskettes etc.).

In order to offer a satisfactory level of quality and features matching the VEMMI functionality the resource-file transfer mechanism offers the following facilities:

- the resource-file transfer can be performed without the user being aware of it;
- the resource-file transfer can be performed as a parallel task. During the resource-file transfer the user may continue to interact with the VEMMI application (although if the network speed is not sufficient, the server response time may be adversely affected by the resource-file transfer operation);
- the application may open a VEMMI object (window) displaying the actual status of the transfer (e.g.: a graphics representing the percentage of the resource copied), and optionally offer a method to cancel, hold and resume the transfer (e.g.: using buttons);
- several resource-file transfers can be performed independently;
- possible user interaction on the resource-file transfer operation (abort).

The support of a permanent storage is not mandatory for a VEMMI terminal.

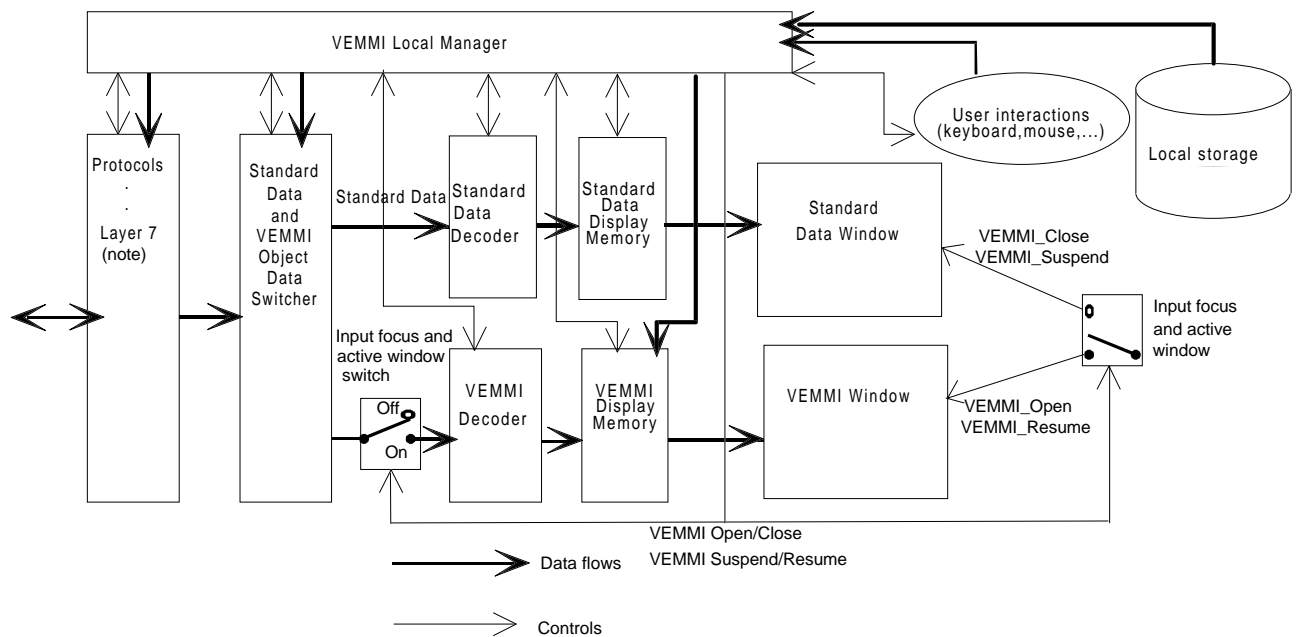
#### 4.3 VEMMI plane structure model

The VEMMI display model consists of two independent planes:

- the standard data memory and standard data window;
- the VEMMI data memory and VEMMI data window.

This model and a possible terminal structure is presented in figure 2.

A VEMMI terminal shall implement the behaviour of this display model. However, no assumption is made on the real physical plane structure of the terminal and how the terminal implements that plane structure model.



NOTE: Standard protocols including SBV based protocols or protocols amended for VEMMI mode as defined in clause 5.

**Figure 2: Example of possible VEMMI terminal structure**

The standard plane is the output area where the retrieval service used as a platform for the VEMMI service displays its data. If a Videotex service is used as a platform the standard plane is the DDA that is used by a regular Videotex application. In the following subclauses it is assumed that Videotex is used as a platform. However, all the rules apply as well if the underlying platform is different from Videotex (e.g. VT100, TV-channel).

The two output planes are independent with respect to their positions and dimensions on the terminal screen.

The terminal has a logical command input queue in which the commands that are received from the network are stored in the order of reception before they are processed by the VEMMI local manager.

##### 4.3.1 The standard plane for Videotex data

The standard plane receives standard Videotex data. It shall continue to support the Videotex data rules and priorities defined in the Videotex ETSs referenced in clause 2.

#### 4.3.2 The VEMMI objects plane

This plane receives VEMMI objects described in this ETS and their encoded data contents. The VEMMI objects plane shall be able to handle object overlapping and restoring mechanisms as referenced in subclauses 4.6 and 4.10.1. Within this plane, and for a given data content, the data rules and priorities as defined in the corresponding standards apply.

#### 4.4 Operation modes for VEMMI terminals

Regardless of the current operation mode of the terminal, it shall always provide specific tools to manage the platform specific functions (disconnect, etc.).

##### 4.4.1 The standard mode

This is the initial mode of operation of a VEMMI terminal when powered-on or reset.

In standard mode, the VEMMI terminal displays standard data in the standard plane. This standard datatype depends on the retrieval service to which the VEMMI terminal is connected. Standard data are fully visible. User inputs are not controlled by the VEMMI local manager.

The terminal shall ignore all VEMMI commands except VEMMI\_Open or VEMMI\_Resume if the VEMMI session previously was suspended with a VEMMI\_Suspend. This command shall perform a switch to the VEMMI mode (VEMMI on/off switches are then switched to the on position).

##### 4.4.2 The VEMMI mode

In VEMMI mode, a VEMMI terminal can receive standard data and VEMMI objects data in two parallel paths, as shown in figure 2.

VEMMI object data are displayed in the VEMMI objects plane. Standard data are displayed in the standard plane. The display order for objects and for components within objects shall correspond to the ascending order of their CINs (component with the smallest CIN is displayed first).

In VEMMI mode, the received standard data is used to update the standard data display memory. An update of the standard data window does not cause its activation.

User inputs are controlled by the VEMMI local manager on the VEMMI window, which is the active window in this mode.

##### 4.4.3 Switching between standard mode and VEMMI mode

Logically, the terminal has two different paths to the communication part, one for the standard data and one for VEMMI data. This allows a distinction between the two data types. On the communication line the two data types can be received interleaved.

The **standard mode** is the initial mode of operation when the connection to the system has been established. The terminal keeps this mode until it receives:

- VEMMI\_Open or;
- VEMMI\_Resume.

and is switched in VEMMI mode.

In **VEMMI mode** the terminal shall process all VEMMI commands and standard data. The commands VEMMI\_Close and VEMMI\_Suspend cause the terminal to leave this mode and the terminal is switched back to the standard mode. With VEMMI\_Suspend the current objects and the VEMMI specific data and states are not destroyed. That means, a temporary switch to the standard mode and a return back to the VEMMI mode with VEMMI\_Resume allows a continuation of the VEMMI dialogue.

The following table shows the active states of the terminal. Standard data shall be processed in all states, independently whether the standard data window is active or not.



**Table 1: Active terminal states**

Command	Process standard data	Active window
VEMMI_Open	Yes	VEMMI window
VEMMI_Resume	Yes	VEMMI window
VEMMI_Close	Yes	Standard data window
VEMMI_Suspend	Yes	Standard data window

#### 4.5 VEMMI elements data content

The following presentation formats, content data types and attributes are defined within this ETS:

- VEMMI high quality text;
- VEMMI bitmaps;
- colour.

The following content datatypes and attributes are defined outside this ETS, but used as data content for VEMMI elements in this ETS:

- Videotex data;
- bitmaps;
- sound;
- graphical;
- video data;
- multimedia data.

A detailed list of all data types that are supported by VEMMI is given in subclause 7.17. A detailed list of all data types that are mandatory to be supported by a VEMMI terminal implementation is given in annex C.

A VEMMI terminal shall understand all commands related to the above mentioned data contents.

It shall not be required to use any specific content datatype in a VEMMI application. However, in order to provide a satisfactory VEMMI service a terminal should, as a minimum requirement, support the datatypes listed in annex C.

##### 4.5.1 Text data definition

Text Data shall be encoded in accordance with ISO 8859 [13]. Depending on the terminal implementation, only a part of ISO 8859 [13] may be supported (see annex B for ISO 8859 mandatory subset of ISO 8859 [13]). In addition, the host application can use the VEMMI\_Identify\_Term\_Cap command to enquire about the character sets supported in the terminal. A specific character set can be selected using the VEMMI\_Set\_Options command. This text data encoding is considered as the basic text encoding. When several character sets are used in the same application (multilingual application), the application can use the registered ISO appropriate designation escape sequences, as defined in ISO 2022 [10] and ISO 8859 [13] to switch between the different character sets.

The following text data encodings may optionally be used:

- ISO/IEC 10646-1 Unicode [15];
- CCITT Recommendation T.51 [2] as given in annex A;
- CCITT Recommendation T.52 [3] (for further study).

For displaying window titles, menu choices and labels, the terminal application can use the local system font. For the application specific text-output, VEMMI offers enhanced capabilities. However, simple text based terminals can use the system font to display the high quality text. These terminals may then ignore the text attributes that they are not able to process and display text using the system font, in the available system colour, on the system background.

Text data can be part of an object or component definition or it can be defined as a resource object. The following text attributes are specified:

- text colour;
- text font;
- text height.

The text colour defines the colour of the characters foreground. The character background is terminal specific. The text font is a set of characters with a particular, similar character design. In this context, an application can choose a font family, the specific font pertaining to this family is selected by the terminal application (in this document, by "font", a font family is meant). The following table presents the VEMMI font families:

Table 2: VEMMI font families

Font family	Font characteristics	Example
SWISS	Proportional, without serifs	Helvetica, Switzerland
ROMAN	Proportional, with serifs	Times Roman
FIXFONT	Monospaced	Courier

The fonts of these families should support variable sizes (scaleable fonts).

The "text height" specifies the height of the font. It is the height of the character cell, including a possible internal leading. The local manager shall insert additional space between adjacent text rows. The height is measured in points (1/72 inch). One point corresponds to 1/400 Normalized Device Co-ordinate (NDC). Italicised characters can be achieved with the "Italic" attribute. "Bold" increases the line thickness and "Underline" draws a line under each character.

EXAMPLE:

**SWISS, 10 points and bold**

*Roman, 16 points,  
italic*

Roman, 8 points and underlined

**FIXFONT, 12 points, white**

A combination of a font and a set of attributes is further named an attributed font and handled as a font resource object. It is referenced via the FIN.

The following control characters should be used to format the text correctly:

- SP, NBSP, SHY: see ISO 8859 [13];
- CR, LF: see CCITT Recommendation T.50 [1].

To provide the "Line Feed" functionality within the text content of a component or a component item the Carriage Return (CR) + Line Feed (LF) control characters given in CCITT Recommendation T.50 [1] shall be used.

#### 4.5.2 Bitmaps

A bitmap is a pixel matrix containing either colour indices which point into the colour table, or Red Green Blue (RGB) components. A bitmap can be referenced by an object or component and when it is instantiated by the terminals it appears as a rectangle with a colour pattern of the corresponding matrix of pixels. The definition is almost device-independent while the relationship between the bitmap bits and the pixels on the device is device specific. Some bitmaps can have associated colour tables different from the one defined in VEMMI.

The bitmap is handled as a resource object and referenced by objects or components via the BIN which is unique in one application. Several bitmaps can be defined at a given point in time.

### 4.5.3 Colour

The colour table provides a method for accessing the colour capabilities of the terminal device. It is assumed that the device can display at least 256 colours simultaneously. Since the device colour table is shared by more than one application, the User Interfaces (UIs) provide mechanisms (e.g. logical colour palettes) to support each application with its own table. This clause defines the colour table for the VEMMI application.

From the 256 colours, the UI reserves 20 colours for a system table. Therefore, the maximum number of colours for the VEMMI colour table is 236. In practice an application will use only a few numbers from these colours. The host application can define the colour table by sending the RGB components. Objects, components or bitmaps can select these colours via indices. By default, 16 colour entries (0 to 15) are predefined. If the application uses a colour index > 15 which was not defined, the displayed colour is terminal dependent.

Colour entries may be in use by other active applications in the terminals. This shall be managed by the local UI.

Monochrome terminals support either two colour indices (e.g. black and white) or the colour table loaded with shades of one colour.

### 4.5.4 Videotex data

Data encoded in accordance with ETS 300 072 [20], ETS 300 073 [21], ETS 300 076 [22] for e.g.: text, geometric.

Data encoded in accordance with ETS 300 149 [24] for audio data.

Data encoded in accordance with ETS 300 177 [25] for photographic data.

The support of Videotex data is optional for a VEMMI terminal.

The host application can inquire about the support of these data types using the VEMMI\_Identify\_Term\_Cap command.

### 4.5.5 Sound

An application can embed references to sound sequences in objects. The terminal stores the sound in files. Upon a user action the sound is processed, possibly with an audio interface card.

The three sound formats are:

- wave (WAVE);
- Musical Instrument Digital Interface (MIDI);
- ETS 300 149 [24] for audio data.

Sound can also be provided using the Videotex audio syntax defined as Videotex data content (see subclause 4.5.4).

### 4.5.6 Graphical data

Graphical data, such as lines, arcs, fill areas and the relevant drawing attributes are supported by this standard (Videotex geometric display and Computer Graphics Metafile (CGM)). An application can also use VEMMI-integrated programmes (via operative objects), in which graphical drawing operations are performed.

### 4.5.7 Video data

This ETS identifies video data formats and provides the container (object) which can be used to embed audiovisual and pure video information in VEMMI applications.

## 4.6 VEMMI objects positioning and dimensioning

### 4.6.1 Positioning

The standard plane shall continue to support the co-ordinate system used at the underlying platform.

The VEMMI objects plane shall support a NDC system for positioning VEMMI objects and components within the DDA (see figure 3). The normalized co-ordinates are theoretically expressed in the range 0;0 to 1;1. The 0;0 co-ordinate origin reference point represents the upper left hand corner of the DDA, the 1;1 co-ordinate point represents the lower right hand corner of a virtual square DDA with a side equal to the unit. For a non-square DDA the unit is equal to the greatest side of the DDA.

For typical display devices, using common 4:3 aspect ratio, the horizontal positioning is in the range 0 to 1 corresponding to the whole width of the DDA, the vertical positioning is in the range 0 to 0,75 corresponding to the whole height of the DDA.

The origin reference point to position an object within the DDA is always the upper left corner of the DDA. The origin reference point to position a component within its parent object (container object) is also the upper left corner of a virtual DDA attached to the upper left corner of the object; that virtual DDA has exactly the same size as the standard DDA but with a position translation up to the object for positioning its own components (see note).

VEMMI items are, generally, implicitly positioned with respect to the width or the height of the nearest item on the left or above.

NOTE: For specific rules for Dialogue Boxes with title and/or border see subclause 7.6.

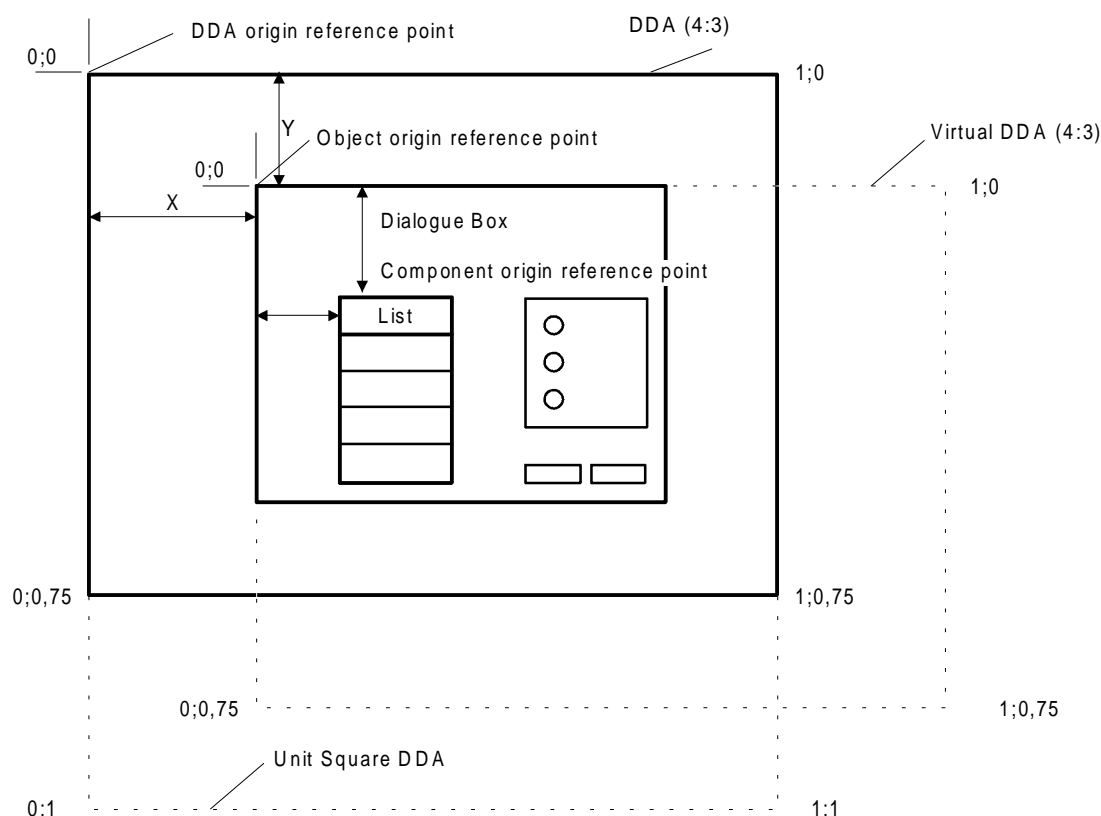


Figure 3: VEMMI positioning NDC space

#### 4.6.2 Dimensioning

The standard plane shall continue to support the co-ordinate system used for dimensioning at the underlying platform.

The VEMMI objects plane and VEMMI objects and components shall support a dimensioning system based NDC.

The width and the height of a VEMMI object or of a VEMMI component, expressed in NDC, are referred with respect to the object positioning reference point (upper left-hand corner), see figure 4.

In a VEMMI element, data content follows the rules for the coding of its relevant data syntax including a possible picture placement or positioning using the relevant co-ordinate system.

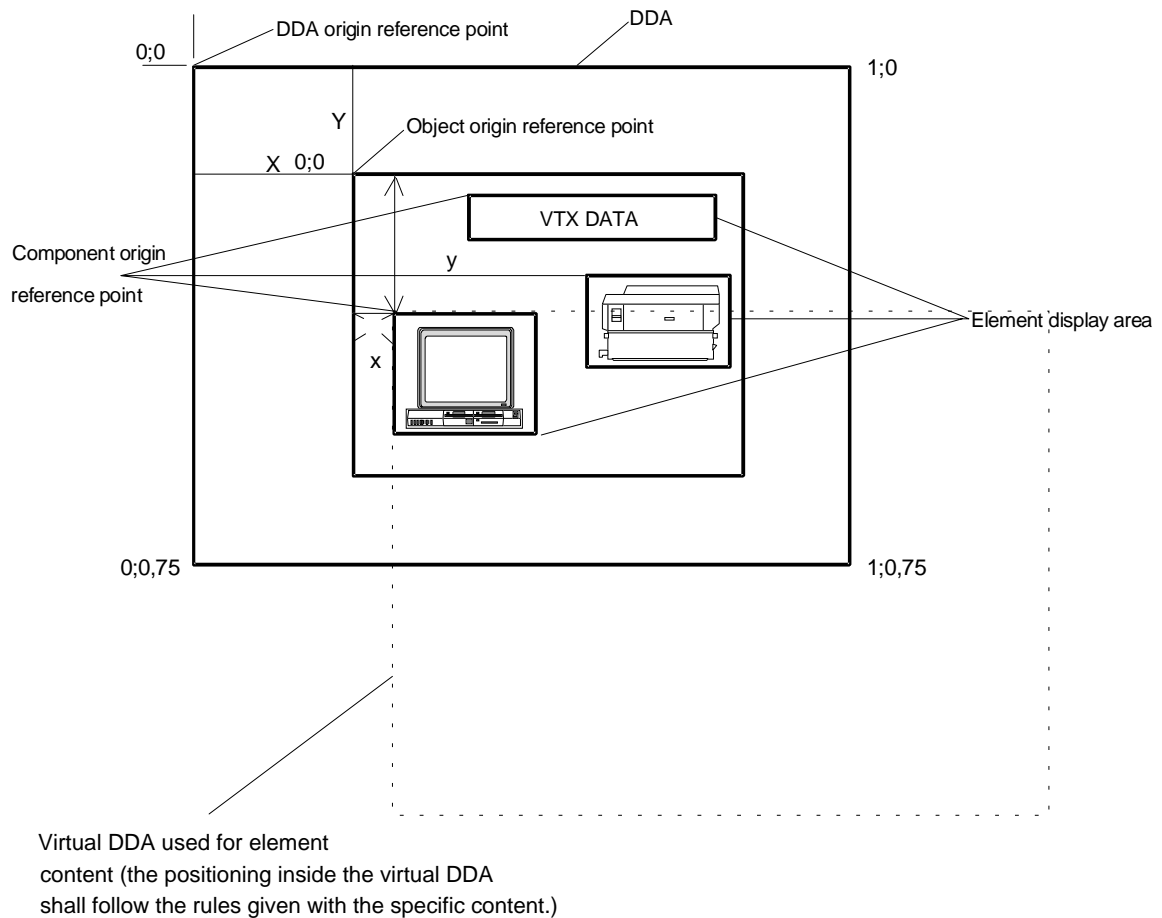


Figure 4: VEMMI positioning system

#### 4.7 VEMMI elements states and state parameters

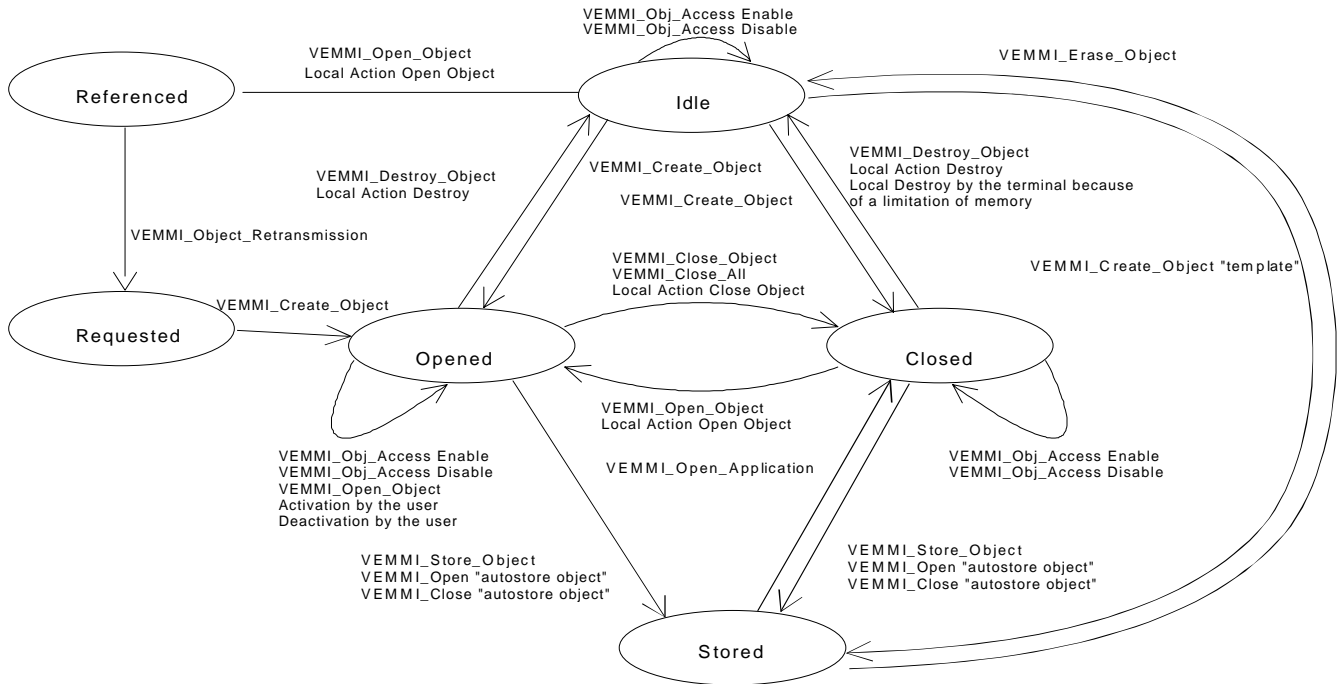
##### 4.7.1 Object

An object can adopt different logical states with different state parameters. These states and state parameters may be results of:

- user inputs;
- local actions;
- actions taken by the VEMMI local manager;
- commands from the VEMMI application directly or via a metacode object.

## Definition of object states

Figure 5 shows the commands which can be used by the VEMMI application to change the object state or its state parameters.



### Figure 5: State diagram for objects

In general, the following logical states for objects are defined with certain restrictions applying to specific objects (see clause 7):

- Opened - an object which exists in the terminal is displayed:
  - a) opening an object is initiated by the VEMMI application, either as initial state at the time of the object creation, by a direct VEMMI command any time during the application or by a local action which is associated to a component and triggered by user interaction.
- Closed: - an object which exists in the terminal is not displayed:
  - a) closing an object is initiated by the VEMMI application, either as initial state at the time of the object creation, by a command any time during the application or by a local action which is associated to a component and triggered by user interaction.
- Referenced: - an object which does not exist in the terminal was referenced by a VEMMI\_Open\_Object command or the local action "Open object". The terminal shall request the retransmission of the specified object with the VEMMI\_Object\_Retransmission command;
- Requested: - after the attempt to open a non-existing object a request for object retransmission shall be sent from the terminal to the VEMMI application. While the terminal is waiting for the creation of the requested object the object is in the requested state;
- Stored: - a copy of an object (including all its components), which was used in an earlier session or in the actual opened session and has been stored permanently in the terminal (e.g. on hard disk):
  - a) a stored object shall implicitly have the closed state. Stored objects are made available for object operations with the Open Application command. Then they behave like received closed objects;
  - b) if an object that was previously stored in the terminal is requested to be stored again by the host application, the newer one shall replace the older one with all its parameters and states;

- c) stored objects can be destroyed with the command VEMMI\_Erase\_Objects.

An object which is in the closed state may be destroyed from terminal memory by a local decision of the VEMMI local manager (e.g. because of a memory limitation). The VEMMI application may be informed about this local deletion using the VEMMI error command "Object Destroy Indication". If the terminal receives a VEMMI\_Open\_Object command referring to an object that has been destroyed or if a local action "Open object" referring to a destroyed object is executed, the terminal shall request the retransmission of the object from the application, using the VEMMI\_Object\_Retransmission command. The VEMMI application shall then create the requested object again using the VEMMI\_Create\_Object command, applying its current state and its current state parameters within the VEMMI application. A VEMMI\_Create\_Object command resulting from a VEMMI\_Object\_Retransmission request shall always create the object in the open state.

#### **4.7.1.2 Definition of object state parameters**

In general, the following state parameters for objects are defined with certain restrictions applying to specific objects (see clause 7):

- active - an object currently has the input focus. The user inputs always refer to the active object. The active object is on top of the DDA. If the active object is inaccessible it is not possible for the user to interact with it. The active state management is described in subclause 4.10.1;
- inactive - the opposite of active;
- accessible - the user can interact with the object. Enabling the interaction with an object is initiated by the VEMMI application, either as initial state at the time of the object creation, by a VEMMI command any time during the application or by a local action which is associated to a component and triggered by user interaction;
- inaccessible - the opposite of accessible.

The active/inactive state parameter applies only to components which are in the opened state. The accessible/inaccessible state parameter is applicable to open and closed components. Any combination of the state parameters active/inactive and accessible/inaccessible can exist for an open component.

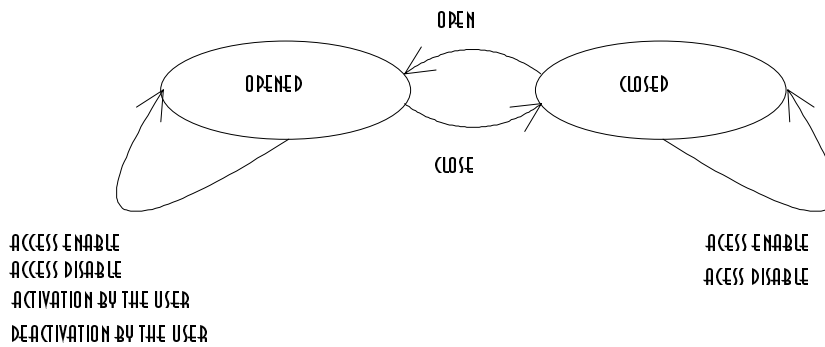
#### **4.7.2 Component**

##### **4.7.2.1 Definition of component states**

A component can adopt different logical states with different state parameters. These states and state parameters may be results of:

- user inputs;
- local actions;
- commands from the VEMMI application.

Figure 6 shows the possible states for components. The states and state parameters of components can be changed during the applications using the VEMMI\_Modify\_Component command.



**Figure 6: State diagram for components**

In general, the following logical states for components are defined with certain restriction applying to specific components (see clause 7).

- Opened: - a component which exists within an object is displayed. Opening a component is initiated by the VEMMI application, either as initial state at the time of the object creation, by a VEMMI\_Modify\_Component command any time during the application or by a local action which is associated to a component and triggered by user interaction;
- Closed - a component which exists in the terminal is not displayed. Closing a component is initiated by the VEMMI application, either as initial state at the time of the object creation, by a VEMMI\_Modify\_Component command any time during the application or by a local action which is associated to a component and triggered by user interaction.

NOTE: The local deletion and retransmission of text components is possible (see subclause 5.2.2). As the corresponding component states (idle, referenced, requested) are not applicable to any other component they are not described explicitly. The retransmission-functionality for text components is similar to the retransmission-functionality for objects.

#### 4.7.2.2 Definition of component state parameters

In general the following state parameters for components are defined with certain restrictions applying to specific components (see clause 7).

- active - a component currently has the input focus. The user input always refers to the active component within the active object. If the active component is inaccessible the user cannot interact with it. The active state management is described in subclause 4.10.1;
- inactive - the opposite of active;
- accessible - the user can interact with the component if it is part of the current active object. Enabling the user interaction with a component is initiated by the VEMMI application, either as initial state at the time of the object creation, by a VEMMI\_Modify\_Component command any time during the application or by a local action which is associated with a component and triggered by user interaction;
- inaccessible - the opposite of accessible.

#### 4.8 Local action management

VEMMI commands are generally issued by the VEMMI application to change the state or the state parameters of an object or component. Only a very limited number of changes of object or component states or their parameters can be initiated by the terminal.

To enable the VEMMI application to interact with the user, the terminal shall provide the capability to report the user inputs on the activation or validation of components.



The result of these interactions is that changes of object or component states, which are the result of user inputs, could be delayed because the user inputs were sent to the VEMMI application first and then the application could react with a VEMMI command to change the state. This procedure might cause unacceptable response times on data links with lower data throughput. To improve the response times, a set of local actions has been defined.

Local actions are part of the component definitions and stored in the terminal at the time of the object creation. Each local action can consist of a list of "Element specific commands", a list of "Report commands" and a list of "General commands" in any order. The following commands are defined:

- Element specific commands:
  - a) open components of the parent object (parameter: list of CINs);
  - b) close components of the parent object (parameter: list of CINs);
  - c) open objects (parameter: list of OINs);
  - d) close objects (parameter: list of OINs) with which the local action is associated;
  - e) report the current values of the component;
  - f) report the values of all components in the parent object;
  - g) report all values of all components in the parent objects that have changed (also via a VEMMI\_Modify\_Component) since the last report or since the object creation if this is the first report.
- General commands:
  - a) user lock;
  - b) restore creation values of all input components in the parent object. This command should only be applied if the object was created with the "StoreCreationValue" attribute set true;
  - c) no operation. This command is used to delete local action definition with the VEMMI\_Modify\_Component command.

These commands are part of the object definition and they can appear in the local action of a component in any possible order. The order of appearance shall be equal to the order of their execution. There are two specific trigger events which induce the performance of local actions:

- activation of a component;
- validation of a component.

A list of "Element specific commands", a list of "Report commands" and a list of "General commands" can be used to specify one local action for a component. No command of a specific type is mandatory for a local action.

There are two specific trigger events which induce the performance of local actions:

- activation of a component;
- validation of a component.

A local action can be associated to each of these trigger events.

#### **4.9 Memory considerations**

VEMMI terminals should have memory capabilities correctly dimensioned in order to provide a satisfactory VEMMI service in close relationship with their data type contents handling.

When there is not enough memory to execute a VEMMI command or a local action the local manager may decide to destroy some closed objects. This mechanism is terminal dependent.

The VEMMI terminal shall send a VEMMI\_Error (Out of memory) message to the VEMMI application when it does not have the possibility to release enough memory to process a received object description with a VEMMI\_Create\_Object command, VEMMI\_Modify\_Component command or VEMMI\_Open\_Object command. In that case it may ignore further VEMMI\_Create\_Object commands as long as the VEMMI application has not released a significant part of memory using VEMMI\_Close\_Object or VEMMI\_Destroy\_Object commands. The VEMMI application may also decide to disconnect the line if the

application cannot be performed correctly due to a lack of memory in the terminal; however, the user shall always be informed of that decision by means of a message in a Message Box.

#### **4.10 Common rules for object handling**

##### **4.10.1 Active state and focus management**

A VEMMI object can become active as the result of:

- being opened by the VEMMI application;
- user access;
- a local action with the element specific command "open object";
- closing the active object. The open object that was most recently active becomes the new active object.

The active state shall be given up by the last opened VEMMI object to a user access to a different opened VEMMI object. When an accessible opened VEMMI object is accessed by the user, the active state shall be given to this VEMMI object by the local manager. The active state is then handled by the local manager up to a new VEMMI object opening or re-opening action from the VEMMI application. There should always be one single active object on the DDA. The active object can be inaccessible.

If objects use a common area on the DDA, successive object activation leads to partial or full object overlapping. The active object shall always be fully visible to the user just after its activation. The previous active object is logically just beneath the active object. This rule shall apply to all previously active objects.

When set to active, the VEMMI object and/or one of its components shall receive its focus from the local manager. The active state of the VEMMI object and/or of the VEMMI components shall be clearly indicated visually to the user. Within the active VEMMI object, not more than one accessible VEMMI component shall receive the active state and the focus.

The first time an object is displayed and consequently activated, the component which gets the focus may be specified by the VEMMI application within the object description. If there is no component specified or the component specified is not opened, then the terminal shall give the focus to the first created opened and accessible component. At other times the manner of giving the focus on activation of an object to one of its components is terminal dependent (same rule as above, memorisation of the last accessed component, the nearest component of the user access to the parent object).

##### **4.10.2 Behaviour of the modal mode**

For a modal active object, any user interaction with another VEMMI object of the current VEMMI application shall be forbidden by the terminal, any tentative attempt shall be indicated to the user (buzzer).

All the commands received from the host and all the local actions resulting from user interactions with the components of the modal object shall be executed.

##### **4.10.3 Size considerations and clipping**

The data content of a VEMMI element shall be sent with dimensions and a possible position compatible with the size description of its container (object or component). A VEMMI terminal shall clip non-consistent data content to the overall dimensions of the container.

For bitmap data content, specific rules apply that are given with the description of the elements.

#### **4.11 Local object storage**

A host application has the possibility to request the loading of objects locally stored in the terminal. Such a set of objects is application specific and contains typically those parts which remain unchanged for a longer time period and it is, therefore, more efficient not to transmit them in each dialogue session to the terminal. The loaded objects are processed in the same way as if they were received during the present session. Upon a host request they can be restored again in order to keep them up-to-date. The objects storage does not imply a removal of the objects from the terminal application, only a copy of the objects is

stored. The dialogue may continue and is not affected by the objects storage. An object set is identified with the application identifier.

The timestamp is received from the host at the beginning of the application. It is stored with the objects and re-used in later sessions. The timestamp value is never calculated by the terminal, it is only used in comparison to find outdated objects. Whenever an object is stored locally in the terminal it shall have an associated timestamp. The value of this timestamp shall be determined as follows:

- if the object was created by the host (by VEMMI\_Create\_Object) during the current session or if any component of the object was modified (by VEMMI\_Modify\_Component) during the current session, the object shall be stored with the timestamp value given by the VEMMI\_Open\_Application command for the current application;
- in all other cases the objects timestamp shall remain unchanged. (The object was loaded without change from the local storage during this session).

Two attributes which are part of the object definition can be used additionally to request the object storage either immediately after the object creation (template) or at the end of the application (autostore).

A stored set of objects contains the following:

- the application identifier;
- the objects including their components and the associated timestamp;
- the referenced files (if a referenced file is not available in the terminal the object is considered incomplete and shall not be stored);
- the relevant loaded colour table.

The storage format is terminal specific because the objects are loaded and stored only by the same terminal. For the purpose of object interchange the VEMMI local storage interchange format is defined in subclause 4.14.

The terminal implementation should provide the user with the appropriate means to manage the local storage and typically the user may limit its maximum size.

#### 4.12 Symbolic directory names

In order to support the host application in locating the correct files, symbolic directory names are defined. They can be used as part of the "filename" parameter. The terminal maps them on real directory res. filenames. In the following example a host application uses a symbolic directory name to identify a bitmap file:

VEMMI\_Create\_Object (OIN=7, ... , filename="\$CD\ARTISTS\DANCER\LISA.BMP", pict file type=BMP).

The string "\$CD" identifies the CD-ROM drive. The terminal substitutes this symbolic name with the real drive identification. The complete file identification might be "H:\ARTISTS\DANCER\LISA.BMP".

The following symbolic names are predefined, the application may use other symbolic names:

Name	Meaning
\$APL	reference directory of the application(s) stored in the terminal
\$FTD	reference directory of downloaded files
\$CD	drive resp. directory name of a CD-ROM drive

These symbolic names can be used in the "filename" parameter of the VEMMI commands. The symbolic name shall appear first and only once in the filename parameter. For the separation of the symbolic name from additional sub-directories and from the filename, the character "\", code 5/12, should be used.

#### 4.13 Specific rules for dedicated terminals and PDAs

All service primitives including their parameters shall be understood also by a dedicated terminal. Regarding the execution of the commands related to specific functionality, a dedicated terminal may apply graceful degradation mechanisms.

The following functionality can be gracefully degraded by a dedicated terminal:

- colour definition;
- in-text attributes (fonts, colours, font attributes);
- local storage;
- NDC positioning and dimensioning.

For dedicated VEMMI terminals only able to handle Videotex data contents on a character basis (alphamosaic, Dynamically Redefinable Character Set (DRCS), Photographic Profile 1), VEMMI objects and components can be physically positioned and dimensioned to the nearest corresponding display character position. Consequently, to avoid side effects between VEMMI objects, NDC object positions should be expressed in a direct multiple of character positions for a VEMMI application using only this type of Videotex data content.

For dedicated VEMMI terminals only able to handle Videotex data contents on a character basis, VEMMI can be used with 80 character mode making use of the appropriate SWITCHING SEQUENCE or the DEFINE FORMAT Videotex Presentation Data Element (VPDE) to switch to the desired Data Syntax. Eighty character mode is not mandatory for a VEMMI terminal.

Dedicated VEMMI terminals shall recognise, but not necessarily process the whole or part of the VEMMI high quality text attributes they are unable to display. These terminals may then ignore the text attributes and display text using the system font, in the available system colour, on the system background.

Dedicated VEMMI terminals shall support the different kinds of VEMMI objects, with possible restrictions applying to the operative objects. As a file management system is not mandatory, they can manage the files they receive in the VEMMI\_Resource-file\_Transfer command in their own internal representation.

Dedicated terminals should support the minimum kernel datatypes defined in annex C. However, when used in a retrieval service where the standard mode is the Videotex mode and when the terminal does not support the Videotex photographic syntax (to be verified using the Terminal Facility Identifier (TFI)) the support of Joint Photographic Experts Groups (JPEG) and Graphics Interchange Format (GIF) is not mandatory. When the terminal supports the Videotex photographic syntax the support of JPEG is mandatory and the support of GIF is recommended. Under the same service conditions, when a terminal does not support the Videotex audio syntax the support of WAVE and MIDI is not mandatory. When the terminal supports the Videotex audio syntax the support of WAVE is mandatory and the support of MIDI is recommended.

Dedicated terminals need not support the stretching of bitmap data to fill the element display area. If the presentation attribute for a bitmap is set to "stretched" the terminal may present the bitmap filled.

VEMMI applications targeting Personal Digital Assistants (PDA)s and dedicated terminals should not use font sizes smaller than 10 point because these terminals will often have limited display capabilities.

#### 4.14 VEMMI local storage interchange format

To allow the interchange of locally stored objects between terminals the following rules shall be followed when storing the objects:

- the VEMMI objects constituting an application are stored independently of each other in a specific sub-directory, in order to allow for an easy insertion, replacement and retrieval of a single object;
- the application objects are then stored in a specific sub-directory bearing the application name (abbreviated to the first 8 characters on platforms that do not support more than 8 characters in filename):

- a) the files in which the objects are stored, are named nnnnnnnn (8 characters with VEMMI Interchange Format (VIF) extension): the name nnnnnnnn represents the object OIN in hexadecimal format (from 1 to  $2^{32-1}$ );
- b) the internal data stored in the files represent the timestamp, immediately followed by the object content in the VEMMI encoded format.

## 5 Service description

This clause describes the service characteristics offered by the VEMMI.

The service is defined between a terminal and a remote application. No assumption is made about the way the terminal accesses the service (PSTN/PSPDN/ISDN) and the way the connection is established.

All VEMMI service elements or VEMMI commands shall be issued only on an established network connection. Before sending the first VEMMI command the VEMMI application should issue a TFI request to ensure that the terminal is able to support the VEMMI standard.

All VEMMI service elements are mandatory for a VEMMI terminal. A VEMMI terminal shall understand all VEMMI service elements including all their parameters.

The VEMMI services are divided into two groups:

- services initiated by the VEMMI application;
- services initiated by the VEMMI terminal.

A second source of commands for a VEMMI terminal is the metacode object.

**Table 3: VEMMI services**

<b>VEMMI Services</b>	<b>Initiated Appl/Term</b>	<b>Function</b>
VEMMI_Open	Application	Switch to VEMMI mode and reset
VEMMI_Close	Application	Switch to standard mode and reset
VEMMI_Suspend	Application	Switch to standard mode
VEMMI_Resume	Application	Switch to VEMMI mode
VEMMI_Identify_Term_Cap	Application	Request for terminal capabilities
VEMMI_Identify_Term_Cap_Resp	Terminal	Response to the "identify terminal capability" request
VEMMI_Set_Options	Application	Set options in the terminal
VEMMI_Create_Object	Application	Object definition
VEMMI_Open_Object	Application	Display object
VEMMI_Open_Blocking_Object	Application	Open object
VEMMI_Close_Object	Application	Clear an object from the screen
VEMMI_Close_All	Application	Clear all the objects from the screen
VEMMI_Destroy_Object	Application	Clear object in the terminal memory
VEMMI_Obj_Access_Disable	Application	User access is not permitted
VEMMI_Obj_Access_Enable	Application	User access is permitted
VEMMI_Modify_Component	Application	Modify components characteristics
VEMMI_Obj_Location_Change	Application	Defines a new object position
VEMMI_Load_Col_Table	Application	Loads a colour table
VEMMI_Reset_Col_Table	Application	Resets a colour table
VEMMI_Open_Application	Application	Opens an application
VEMMI_Open_Application_Resp	Terminal	Response to "open application" request
VEMMI_Delete_Outdated_Objects	Application	Deletes outdated objects
VEMMI_Store_Objects	Application	Stores a set of objects
VEMMI_Store_Objects_Resp	Terminal	Response to a "store object" request
VEMMI_Erase_Objects	Application	Delete objects from the local storage
VEMMI_User_Lock	Application	User inputs are not permitted
VEMMI_User_Unlock	Application	User inputs are permitted
VEMMI_Object_Retransmission	Terminal	Request for object retransmission
VEMMI_User_Data	Terminal	User data in VEMMI mode
VEMMI_Error	Terminal	Error condition or situation
VEMMI_Resource-file_Transfer	Application	Resource-file transfer
VEMMI_Transfer_Acknowledge	Terminal	Resource-file transfer acknowledge
Abbreviations: Appl = Application; Term = Terminal.		

The following services are confirmed explicitly:

- VEMMI\_Identify\_Term\_Cap is confirmed with VEMMI\_Identify\_Term\_Cap\_Resp;
- VEMMI\_Open\_Application is confirmed with VEMMI\_Open\_Application\_Resp;
- VEMMI\_Store\_Objects is confirmed with VEMMI\_Store\_Objects\_Resp.

There is no explicit confirmation for any other of the service elements.

All VEMMI services, except VEMMI\_Open\_Object, which refer to a non-existing object shall be ignored by the VEMMI terminal. If a VEMMI\_Open\_Object refers to a non-existing object the terminal shall request its retransmission.

### **5.1 Service elements initiated by the VEMMI application**

This subclause describes all VEMMI services but only those parameters which are necessary in order to understand the service are listed. For a complete parameter list, see clause 8.

### 5.1.1 VEMMI\_Open

This service element shall be used to switch the VEMMI terminal to the VEMMI operational mode. The switching mechanism is defined in subclause 4.4.3. In VEMMI operational mode the mechanism described in subclause 4.4.2 shall apply.

In addition to the switch to VEMMI operational mode the service element shall perform the following resets:

- close an open application;
- destroy all objects from the set of objects that are in the "opened" or "closed" state;
- all opened objects are cleared from the display;
- release all identification numbers for resource objects;
- set the colour table to the predefined values.

Objects stored in local storage are not affected.

If this command is received in the VEMMI mode no switch shall be necessary, only the resets are performed.

**Table 4**

Parameters	Description
Version	Identifier of the VEMMI version.
PrivateMode	Private parameter which can be used in some "page-based" Videotex services. This mode is not described in this ETS.

### 5.1.2 VEMMI\_Close

This service element shall be used to switch the VEMMI terminal to the standard mode. The switching mechanism is defined in subclause 4.4.3. In standard operational mode, the mechanism described in subclause 4.4.1 shall apply.

In addition to the switch to standard mode the service element shall perform the resets defined in subclause 5.1.1.

**Table 5**

Parameters	Description
None	

### 5.1.3 VEMMI\_Resume

This service element shall be used to switch the VEMMI terminal to the VEMMI operational mode. The parameters of this service element are the same as for the VEMMI\_Open (see subclause 5.1.1). With this service element no reset shall be performed.

**Table 6**

Parameters	Description
Version	Identifier of the VEMMI version.
PrivateMode	Private parameter which can be used in some "page-based" Videotex services. This mode is not described in this ETS.

#### 5.1.4 VEMMI\_Suspend

This service element shall be used to switch the VEMMI terminal to the standard mode. No reset shall be performed.

**Table 7**

Parameters	Description
None	

#### 5.1.5 VEMMI\_Identify\_Term\_Cap

This service element shall be used to request identification of the capabilities of the terminal.

The following information can be obtained:

- VEMMI version identification;
- local storage support (supported/not supported);
- list of preferred user languages (string);
- system type (e.g.: IBM compatible, Macintosh, etc.);
- operating system (e.g.: MS-DOS, OS2, Microsoft Windows, etc.);
- operating system version (e.g.: 3.1, etc.) (string);
- a list of supported data types (reference to the VEMMI content encoding identification catalogue);
- a list of file transfer protocols supported by the terminal.

**Table 8**

Parameters	Description
DesiredInformation	The type of information that is requested from the terminal.

If the DesiredInformation parameter is not present, the terminal shall respond by giving a complete list that identifies all its capabilities.

#### 5.1.6 VEMMI\_Set\_Options

This service element shall be used to set options in the terminal.

The following option can be set:

- if the terminal supports different kinds of text encoding, the desired encoding standard can be selected.

**Table 9**

Parameters	Description
TextStandard	Reference to a text encoding standard in the VEMMI content encoding identification catalogue which shall be used at the beginning of the encoding of datatypes string and character.

#### 5.1.7 VEMMI\_Create\_Object

This service element shall be used to define a VEMMI object and its components in the VEMMI terminal. The VEMMI terminal shall create the object and shall apply the object state and state parameters given in its definition.

OINs shall be unique at one time in an application. The creation of an object with an already existing OIN shall cause the replacement of the object.

CINs shall be unique within an object. The definition of a component with an already existing CIN shall cause the destruction of the previously defined component.



Table 10

Parameters	Description
Object Identification Number	
Template	If an application is open the object shall be stored immediately after creation.
Autostore	If an application is open the object shall be stored at the reception of a VEMMI_Open, a VEMMI_Close or a VEMMI_Open_Application.
NOTE: The parameters Template and Autostore exclude each other.	

#### 5.1.8 VEMMI\_Open\_Object

This service element shall be used to set an object to the opened state in the VEMMI terminal. If the referenced object is displayable the terminal shall display it immediately. If the object is not present in the terminal, it shall request the object by sending a VEMMI\_Object\_Retransmission request to the VEMMI application with the specified OIN, in this case "Rule A" as described in subclause 5.3.2 shall be used. The application shall then create the object again, applying its current state and state parameters.

A VEMMI\_Open\_Object command referring to an already opened object causes the activation of the object.

If a VEMMI\_Open\_Object is applied to a resource object the terminal shall verify whether the object and the referenced resource-file are available. If the object is not available it shall be requested. If the referenced resource-file is not available the terminal shall use the service element VEMMI\_Object\_Retransmission with the appropriate parameters to request the file.

If a VEMMI\_Open\_Object is applied to a metacode object the terminal copies all commands of the metacode object on top of the logical command input queue. The next command read from the command input queue is the first command from the metacode. If not present it shall be requested.

If a VEMMI\_Open\_Object is applied to a operative object the terminal executes the programme. If not present it shall be requested.

Table 11

Parameters	Description
Object Identification Number	

#### 5.1.9 VEMMI\_Open\_Blocking\_Object

This service element performs the same function as VEMMI\_Open\_Object except that the retransmission case rule B as described in subclause 5.2.2 shall be used.

#### 5.1.10 VEMMI\_Close\_Object

This service element shall be used to close a VEMMI object in the VEMMI terminal. The VEMMI terminal shall clear the indicated object from the screen but keep it in it's memory. The current state parameters and values of the components shall not be changed.

When receiving a VEMMI\_Close\_Object indication referring to an active object, the new active object in the VEMMI terminal shall be the opened object that was most recently active in the terminal.

For objects that are not displayable or are already in a closed state the VEMMI\_Close\_Object has no effect.

Table 12

Parameters	Description
Object Identification Number	

#### 5.1.11 VEMMI\_Close\_All

This service element shall be used to close all the VEMMI objects in the VEMMI terminal. A VEMMI\_Close\_All service element shall not change the current operational mode of the terminal.

Table 13

Parameters	Description
None	

#### 5.1.12 VEMMI\_Destroy\_Object

This service element shall be used to remove a VEMMI object from a set of objects that are in the "opened" or "closed" state. The terminal shall remove all data belonging to this object from its memory. If the VEMMI\_Destroy\_Object is applied on an opened object it shall be closed by the local manager and then destroyed. If the object was active, the new active object in the VEMMI terminal shall be the opened object that was most recently active in the terminal.

The VEMMI\_Destroy\_Object shall not be applied on a Resource object that is referenced by other opened objects or whose associated file is referenced by other open objects. If a VEMMI\_Destroy\_Object is applied on a resource object also the file associated to the object shall be deleted.

The VEMMI application shall ensure consistency with regard to the handling of resource-files referenced by Resource objects. If a file referenced by a Resource object is not present at the terminal when needed, the terminal shall use the VEMMI\_Object\_Retransmission primitive to request the file.

Table 14

Parameters	Description
Object Identification Number	

#### 5.1.13 VEMMI\_Obj\_Access\_Disable

This service element shall be used to restrict the user access to an object in the VEMMI terminal. The access shall be restricted until a VEMMI\_Obj\_Access\_Enable service element referring to the same object is received by the VEMMI terminal.

If the object was active, the new active object should be the open object that was most recently active in the terminal. A VEMMI\_Obj\_Access\_Disable Indication referring to an active object shall interrupt the user interaction with this object.

Table 15

Parameters	Description
Object Identification Number	

#### 5.1.14 VEMMI\_Obj\_Access\_Enable

This service element shall be used to permit user access to an object in the VEMMI terminal. The access shall be permitted until a VEMMI\_Obj\_Access\_Disable service element referring to the same object is received by the VEMMI terminal.

A VEMMI\_Obj\_Access\_Enable Indication received in the VEMMI terminal and referring to an inactive object shall not change this state parameter.

Table 16

Parameters	Description
Object Identification Number	

### 5.1.15 VEMMI\_Modify\_Component

This service element shall be used to modify one or several VEMMI components belonging to a VEMMI object created into the VEMMI terminal. The possible modifications in VEMMI components are defined in the component definition tables of subclause 9.6.

If a VEMMI\_Modify\_Component command is applied to an opened object it shall have an immediate visual effect in the VEMMI terminal if the modified part is visible.

A VEMMI\_Modify\_Component command shall not change the active/inactive state parameter of the object in the VEMMI terminal.

Table 17

Parameters	Description
Object Identification Number	
Modification Description	

### 5.1.16 VEMMI\_Obj\_Location\_Change

This service element shall be used only to change the position of a VEMMI object (not component) in the VEMMI terminal screen. If a VEMMI\_Obj\_Location\_Change command is referring to a VEMMI closed object no visible effects shall be encountered. This service element shall not change the active/inactive state parameter of an object.

A VEMMI\_Obj\_Location\_Change service element shall never be applied to an Application Bar.

Table 18

Parameters	Description
Object Identification Number	
XPos	New horizontal position.
YPos	New vertical position.

### 5.1.17 VEMMI\_Load\_Col\_Table

This service element shall be used by the host to define colours in the colour table.

The colours are defined consecutively, starting with the index "ColEntry". The terminal shall send an error message to the host, if the requested colours cannot be defined, e.g. because the device supports less colours than the host requested.

Table 19

Parameters	Description
ColEntry	Colour index from which the colours are defined in ascending order.
ColRGBList	The list consists of triplets which give the red, green and blue component of a colour. Each component value is in the range 0 to 255. The value 0 means no intensity for the component, the value 255 means maximum intensity.
NOTE: If this service is not supported by the terminal, this shall be reported to the using application using the VEMMI_Error "Service not supported".	

### 5.1.18 VEMMI\_Reset\_Col\_Table

This service element shall be used by the host to reset all previously defined colours. This command has no parameters. Only the default colours (indices 0 to 15) remain defined. If the default colours have been redefined with VEMMI\_Load\_Col\_Table, this command sets the default colours to their predefined values.

**Table 20**

Parameters	Description
None	

The default colours are given in the following table:

**Table 21: Default colours**

Colour Index	Red	Green	Blue	Colour
0	0	0	0	Black
1	0	0	168	Blue
2	0	168	0	Green
3	0	168	168	Cyan
4	168	0	0	Red
5	168	0	168	Magenta
6	168	84	0	Brown
7	168	168	168	White
8	84	84	84	Grey
9	84	84	255	Light blue
10	84	255	84	Light green
11	84	255	255	Light cyan
12	255	84	84	Light red
13	255	84	255	Light magenta
14	255	255	84	Yellow
15	255	255	255	Bright white
NOTE: If this service is not supported by the terminal, this shall be reported to the using application using the VEMMI_Error "Service not supported".				

### 5.1.19 VEMMI\_Open\_Application

This service element shall be used by the host to load a set of objects and to define the name and attributes of the current application. Additional support can be provided using symbolic directory names as described in subclause 4.12. Locally stored object sets are identified by the application identifier. A timestamp is used to associate version information with the objects of this application.

The VEMMI\_Open\_Application performs the same resets as the VEMMI\_Open

**Table 22**

Parameters	Description
ApplId	String with the application name.
ApplAddData	Additional application identification data.
Timestamp	Time in seconds since 1.1.1970, Greenwich Mean Time (GMT).

The parameter ApplId shall contain the application name. Additional information, such as name of the company or the organisation to which the application belongs or the author of the application can be given with the parameter ApplAddData, which is not mandatory. The terminal does not need to analyse the content of ApplId, it is only used to identify the requested application.

Application providers should take care to choose unique application identifiers, in order to avoid overwriting of other applications.

#### 5.1.20 VEMMI\_Delete\_Outdated\_Objects

This service element shall be used by the host to request the deletion of outdated local objects of the current application from the terminal storage.

If the host issues a VEMMI\_Delete\_Outdated\_Objects with the timestamp requirements for specific objects the terminal shall react as follows:

- if an object is referenced with a timestamp value higher than the timestamp value of the locally stored object, this object shall be deleted from the local storage. The host application can thus be certain that outdated objects are no longer used.

Table 23

Parameters	Description
UpdateList	This parameter contains groups, each of two elements, a timestamp and a set of objects.

In order to update an application partially, this command is typically sent following a VEMMI\_Open\_Application.

#### 5.1.21 VEMMI\_Store\_Objects

This service element shall be used by the host to request the terminal to store the current loaded objects or a part of them. The timestamp of the current open application, supplied with the VEMMI\_Open\_Application command, is associated to each object.

Table 24

Parameters	Description
Object Identification Numbers	

If the parameter is absent all currently loaded objects are stored.

#### 5.1.22 VEMMI\_Erase\_Objects

This service element shall be used by the host to request the deletion of objects of the current application from the local storage of a terminal. To increase security implementation the VEMMI terminal could ask the user before executing the deletion. (e.g.: one time at the beginning of the application or with every deletion). If no application is open, the command has no effect.

Table 25

Parameters	Description
Object Identification Numbers	

#### 5.1.23 VEMMI\_User\_Lock

This service element shall be used to restrict any user input in the terminal until a VEMMI\_User\_Unlock is sent.

Table 26

Parameters	Description
None	

#### 5.1.24 VEMMI\_User\_Unlock

This service element shall be used to permit user inputs in the terminal.

**Table 27**

Parameters	Description
None	

#### 5.1.25 VEMMI\_Resource-file\_Transfer

This service element shall be used to transfer files referenced by VEMMI resource objects from a VEMMI application to a VEMMI terminal.

A file can be downloaded using one or more VEMMI\_Resource-file\_Transfer commands. A parameter of the VEMMI\_Resource-file\_Transfer command (TransferID) associates the command to a given resource-file transfer. This allows several resource-file transfers to be performed simultaneously with commands pertaining to different resource-file transfers interleaving each other without interference.

The VEMMI\_Resource-file\_Transfer commands pertaining to the same resource-file transfer are numbered consecutively using the parameter BlockNumber.

The first block of a given resource-file transfer shall always contain the following header information of the file:

- filename;
- filesize;
- date of creation.

The first block of a given resource-file transfer shall also contain the number of VEMMI\_Resource-file\_Transfer commands used to transmit the file.

The TransferType parameter of the VEMMI\_Resource-file\_Transfer command indicates if the VEMMI\_Resource-file\_Transfer command of a given resource-file transfer is:

- the first command;
- an intermediate command or;
- if the resource-file transfer is aborted by the VEMMI application.

**Table 28**

Parameters	Description
TransferType	First command, intermediate command, abort.
TransferID	Transfer Identifier.
BlockNumber	Number of the command within a given resource-file transfer.
FileInformation	Header information about the file.
TotalBlocks	Number of VEMMI_Resource_Transfer commands used to transmit a given file.

## 5.2 Service elements initiated by the terminal

### 5.2.1 VEMMI\_Identify\_Term\_Cap\_Resp

This service element shall be used to respond to a VEMMI\_Identify\_Term\_Cap request.

The following information can be given:

- VEMMI version identification;
- local storage support (supported/not supported);
- list of preferred user languages (string);
- system type (e.g.: IBM compatible, Macintosh, etc.);
- operating system (e.g.: MS-DOS, OS2, Microsoft Windows, etc.);
- operating system version (e.g.: 3.1, etc.) (string);
- a list of supported data types (reference to the VEMMI content encoding identification catalogue);
- a list of file transfer protocols supported by the terminal.

**Table 29**

Parameters	Description
SupportedVEMMIVersions	This parameter contains the VEMMI version identification.
ContentList	The list gives references of the supported data types in the VEMMI content encoding identification catalogue.
LocalStorage	Identifies the support of the local storage facility.
PreferredUserLanguages	Languages preferred by the user (for multilingual applications). The first language in the list is the most preferred. The list may contain only one language.
SystemInfo	Information to support the correct use of operative objects (e.g.: System Type, Operating System, Operating System Version).
FileTransfer	A list of file transfer protocols supported by the terminal.

## 5.2.2 VEMMI\_Object\_Retransmission

This service element shall be used to request the VEMMI application to retransmit a VEMMI object. It shall be used only after:

- the reception of a VEMMI\_Open\_Object command;
- a VEMMI\_Open\_Blocking\_Object command;
- a local action "Open object".

Referring to an object that does not exist on the terminal. The service element can also be used by a terminal supporting the anticipation mechanism for text resource objects described in subclause 7.6.1.4 to anticipate user scrolling.

To request a file that is referenced by a Resource object and not available in the terminal, the service element shall be used with the appropriate parameter set.

For the further processing of VEMMI commands until the requested object is received the terminal shall apply one of the following rules:

Rule A:

- inform the user about possible delay due to a retransmission (terminal dependent);
- send a VEMMI\_Object\_Retransmission request;
- execute all VEMMI commands and local actions which do not refer to the requested object;
- memorise all VEMMI commands and local actions which refer to the requested object.

After the recreation and the opening of the requested object, the terminal shall:

- resume the execution of the possibly suspended local action;
- resume the execution of the possibly memorised VEMMI commands received in the order of their reception.

Rule B:

- lock the user;
- inform the user about possible delay due to a retransmission (terminal dependent);
- send a VEMMI\_Object\_Retransmission request;
- suspend and memorise any execution of VEMMI commands or local actions.

After the recreation and the opening of the requested object, the terminal shall:

- resume the execution of the possibly suspended local action;
- resume the execution of the possibly memorised VEMMI commands received in the order of their reception;
- unlock the user.

For referenced bitmap, Videotex or multimedia resources the retransmission rules that shall be applied by the terminal can be specified by the application with an attribute in the encoding of the respective elements.

NOTE: The local destruction and retransmission of text components is a specific functionality for text components only. It is similar to the retransmission-functionality for objects except that the RequestType parameter is set to "text component".

**Table 30**

Parameters	Description
Object Identification Numbers	
RequestType	The parameter indicates if an object, a resource-file or a text component is requested.

### 5.2.3 VEMMI\_User\_Data

This service element shall be used to send user data corresponding to one object, to the VEMMI application.

**Table 31**

Parameters	Description
Object Identification Number	
CompData	User inputs.



#### 5.2.4 VEMMI\_Open\_Application\_Resp

This service element shall be used by the terminal to respond to a VEMMI\_Open\_Application command.

**Table 32**

Parameters	Description
OpenAppIResult	True: the application has been identified and the objects have been loaded. False: the application has not been identified, no objects have been loaded. The parameters that were used with the VEMMI_Open_Application command are used to describe a new application environment.

#### 5.2.5 VEMMI\_Store\_Objects\_Resp

This service element shall be used by the terminal to respond to a VEMMI\_Store\_Objects.

**Table 33**

Parameters	Description
StoreResult	True: the objects have been stored. False: the objects have not been stored.

#### 5.2.6 VEMMI\_Error

This service element shall be used only by the VEMMI terminal to report different error situations to the VEMMI application.

A VEMMI\_Error command with the parameter Type of Error = "Out of memory" is only referring to the specified object. Other objects created after this one may be stored in the terminal.

**Table 34**

Parameters	Description
Object Identification Number	
ErrorType	
Component Identification Number	
ErrorComCode	Identification of the command that caused the error.

The following types of errors are defined:

- General error: a general error has occurred;
- Unknown VEMMI command: the VEMMI command code does not exist;
- Erroneous VEMMI command: the VEMMI command received does not have the mandatory parameters or they have erroneous values;
- Object syntax error: the description of a VEMMI element is not correct in an object creation or component modification;
- Unexpected VEMMI command: the VEMMI command received is correct but it occurs at the wrong point in time;
- Out of memory: the terminal does not have enough memory to store the data corresponding to an object creation or component modification;
- Cannot process audio objects;
- Cannot process video objects;
- Invalid colour index;
- File not found;
- Conversion to bitmap failed;
- Cannot process direct colour definition;

- Operative object was rejected by the user;
- Out of local permanent storage space;
- Object destroy indication (a closed object was destroyed by the terminal);
- Service not supported;
- Object not supported;
- Data content type not supported;
- A datatype not supported has been identified (e.g.: HyperText Markup Language (HTML));
- Version not supported: the VEMMI version requested in the VEMMI\_Open command is not supported by the terminal.

### 5.2.7 VEMMI\_Transfer\_Acknowledge

This service element shall be used by the VEMMI terminal to acknowledge a resource-file transfer. The service element shall be used at the end of every successful resource-file transfer or to indicate a transfer abort.

**Table 35**

Parameters	Description
TransferID	Transfer Identifier.
Acknowledge	Successful transfer or transfer abort.

## 6 VEMMI objects introduction

VEMMI objects offer a basic choice of dialogue elements needed by VEMMI applications.

Following VEMMI objects are defined in this ETS:

- Application Bar;
- Button Bar;
- Pop-Up Menu;
- Dialogue Box;
- Message Box;
- Operative object;
- Bitmap resource object;
- Videotex resource object;
- Text resource object;
- Font resource object;
- Metacode object;
- Sound object;
- Video object;
- Multimedia resource object.

A VEMMI application can be designed using any of the defined objects. No particular VEMMI object or component is mandatory within a VEMMI application. All VEMMI objects may be multiple within a VEMMI application, except the Application Bar. Within an object, all components may be multiple except certain restrictions applying to specific components (see clause 7).

### 6.1 The Application Bar

The Application Bar allows the user to make a choice between the different VEMMI application parts and sub-application parts offered by the selected VEMMI application. When used, the Application Bar is unique and located either on the top (horizontal Bar) or the left side (vertical Bar) of the DDA.

### 6.1.1 Composition

The Application Bar is subdivided into three different logical groups of Menu Choice components. These groups differ in their behaviour and functionality. The three different groups are named:

- Bar;
- Pull-Down Menu;
- Cascading Menu.

The Bar is a horizontal or vertical list of Menu Choice components representing the different parts of the VEMMI application.

The Pull-Down Menus are vertical lists of Menu Choice components associated with the same Menu Choice component of the Bar. The Pull-Down Menus represent the different sub-application parts of the VEMMI application.

The Cascading Menus are vertical lists of Menu Choice components associated with the same Menu Choice component of the Pull-Down Menu. The Cascading Menus represent the different sub-application parts of the VEMMI application.

## 6.2 The Button Bar

The Button Bar permits a choice among a set of alternatives, at a given time, during the execution of the VEMMI application. Each choice is represented by a Button. The Button Bar may be located anywhere in the DDA. The Button Bar can be horizontal or vertical.

### 6.2.1 Composition

The Button Bar is composed of a series of components, named Buttons. The purpose of the Buttons is to trigger a local action to be immediately performed.

## 6.3 The Pop-Up Menu

The Pop-Up Menu offers appropriate choices and sub-choices for a given VEMMI element in its current context. The Pop-Up Menu may be located anywhere in the DDA.

### 6.3.1 Composition

The Pop-Up Menu is subdivided into two different logical groups of Menu Choice components. These groups differ in their behaviour and functionality. The two different groups are named:

- Primary Pop-Up Menu;
- Cascading Menu.

The Primary Pop-Up Menu is a vertical list of Menu Choice components which offers appropriate choices for a given VEMMI element in its current context.

The Cascading Menu is a vertical list of Menu Choice components associated to the same Menu Choice component of the Primary Pop-Up Menu. The Cascading Menu offers appropriate sub-choices for a given VEMMI element in its current context.

## 6.4 The Dialogue Box

The Dialogue Box is the object where the main interaction between the user and the VEMMI application takes place. To enable this interaction, a set of components is defined. These components can be classified as presentation or dialogue components.

Presentation components are inaccessible; their purpose is only to present the different dialogue components coherently and attractively.

Dialogue components permit the interaction between the user and the VEMMI application.

Five presentation components are defined:

- the Separator;
- the Frame;
- the Text Presentation Area;
- the Text component;
- the Graphic Output Area.

Ten dialogue components are defined:

- the Push Button;
- the Text Input Field;
- the Check Box;
- the Radio Button;
- the List Box;
- the Combination Box;
- the Slider;
- the Sensitive Area;
- the Sensitive Text;
- the Multimedia Area.

#### **6.4.1 Composition**

##### **6.4.1.1 The Separator component**

A Separator is a horizontal or vertical solid line. Its goal is to visually separate different dialogue components within the Dialogue Box.

##### **6.4.1.2 The Frame component**

A Frame is a presentation element to visually separate a particular area of the Dialogue Box and its different components.

##### **6.4.1.3 The Text Presentation Area component**

The Text Presentation Area is a rectangular area in which text data is displayed. The purposes of this component are:

- to present text information;
- to title or to label dialogue components;
- to present results from the VEMMI application execution.

##### **6.4.1.4 The Text component**

The purpose of this component is to split large text data in units (text components) and to define the necessary structural information (concatenation of the text components) in order to be displayed in a text area.

##### **6.4.1.5 The Graphic Output Area component**

The purpose of this component is to present graphical data to the user. Several graphical data encoding formats are supported.

##### **6.4.1.6 The Sensitive Text component**

The purpose of this component is to define the activation and validation operations for sensitive text strings.

##### **6.4.1.7 The Push Button component**

The purpose of the Push Button is to trigger a local action to be immediately performed.

#### **6.4.1.8 The Text Input Field component**

The purpose of the Text Input Field is to collect text data, entered by the user. It is a rectangular area composed of a text label associated to an input area.

#### **6.4.1.9 The Check Box component**

The purpose of the Check Box is to enter and to display an independent user choice. A Check Box keeps the value selected or deselected independently of any other Check Boxes.

#### **6.4.1.10 The Radio Button component**

The purpose of the Radio Button is to enter and to display a user choice. The Radio Button permits a single choice among several possibilities offered in a Radio Button group. The selection of one Radio Button leads to the deselection of the other Radio Buttons belonging to the same Radio Button group.

#### **6.4.1.11 The List Box component**

The purpose of the List Box is to offer a single or multiple choice among a list of text items. The list is, generally, not entirely visible to the user, therefore different controls are offered to scroll the list up and down.

#### **6.4.1.12 The Combination Box component**

The purpose of the Combination Box is to combine the functionality of a single choiceList Box with the functionality of a Text Input Field. It contains a list of text items the user can scroll through to complete the Text Input Field. A parameter of the Combination Box specifies whether the Text Input Field content can be edited or not. If the Text Input Field content can be edited, the user can type text directly into the Text Input Field.

A variation of the Combination Box is a Drop Down Combination Box. It is composed of a Combination Box and a Push Button. Only the Text Input Field and the Push Button are displayed until the user selects the associated Push Button. The validation of the Push Button causes the display of the associated List Box.

#### **6.4.1.13 The Slider Component**

The slider offers the selection of an analogue value by moving a adjustable marker on a slide bar between a minimum and a maximum value. The intervals are set by the application.

#### **6.4.1.14 The Sensitive Area component**

The purpose of the Sensitive Area in the Dialogue Box is to offer a selection area associated to a Graphic Output Area.

#### **6.4.1.15 The Multimedia Area component**

The purpose of this component is to present multimedia information to the user and to allow integration of the Internet World Wide Web.

### **6.5 Operative Object**

With this object an application references a program which will be linked to the VEMMI application. This object type provides a method to extent the capabilities of an application during runtime.

### **6.6 Bitmap resource object**

A bitmap resource object contains either the bitmap definition itself or only a reference to a file with the bitmap definition. The object is referenced via the BIN.

## **6.7 Videotex Resource Object**

A Videotex object contains either the Videotex content itself or only a reference to a file with the Videotex content. The object is referenced via the VIN.

## **6.8 Text Resource Object**

This object defines text content as a resource which can be referenced via the TIN. It contains either the text content itself or a reference to a file with the text content.

## **6.9 Font resource object**

This object combines a set of text attributes in on font resource which can be referenced via the FIN.

## **6.10 Metacode object**

The metacode object contains VEMMI commands. This object provides an easy way to avoid unnecessary dialogue steps with the host application.

## **6.11 The Message Box**

The purpose of the Message Box is among others, to display information not requested by a user but sent by the VEMMI application in response to an unexpected event, or when something undesirable occurs.

## **6.12 Sound Object**

The purpose of this object is to link files with sound/audio content to the application.

## **6.13 Video Object**

The purpose of this object is to link files with moving video content to the application.

## **6.14 Multimedia Resource Object**

A Multimedia object contains a reference to a file with the multimedia content. The object is referenced via the MIN.

# **7 Functional description**

## **7.1 General rules for the behaviour of elements**

### **7.1.1 User interaction**

The user shall have the possibility to access and activate the different VEMMI elements. The terminal shall also enable the user to change the values of the VEMMI elements and to validate these inputs.

### **7.1.2 Local actions and reports**

The report of user inputs from the terminal to the server shall be induced by a report command defined in a local action by the VEMMI application. On executing a report command the terminal shall send a VEMMI\_User\_Data command to the VEMMI application. The trigger events which induce the performance of local actions are:

- activation of a component;
- validation of a component.

The available report commands are:

- report OIN + CIN of the component to which the local action is associated;
- report the current values of the component;
- report the values of all components in this object;
- report all values of all components in the parent objects that have changed (also via a VEMMI\_Modify\_Component) since the last report or since the object creation if this is the first report.

If a report command reports more than one value or more than one CIN, the first value (in the encoding of the report) shall be the one of the component that triggered the report.

All local actions associated with a given trigger event shall be executed in the order of their definition and without intermediate execution of commands from the logical command input queue.

Tables 35 and 36 show the possible trigger events and the possible reports for each component.

**Table 36: Trigger events**

Component	Activation	Validation
Menu Choice Bar	✓	✓
Menu Choice Pull-Down	✓	✓
Menu Choice Cascading	✓	✓
Menu Choice Pop-Up	✓	✓
Menu Choice Separator		
Button	✓	✓
Separator		
Frame		
Text Presentation Area		
Text Component		
Push Button	✓	✓
Text Input Field	✓	✓
Check Box	✓	✓
Radio Button	✓	✓
List Box	✓	✓
Combination Box	✓	✓
Slider	✓	✓
Sensitive Area	✓	✓
Sensitive Text	✓	✓
Graphic Output Area		
Multimedia Area	note	note
NOTE: Trigger events are defined by the content.		

Table 37: Reports

Component	CIN	Value of all Cmp	Value (note 1)
Menu Choice Bar	✓		
Menu Choice Pull-Down	✓		
Menu Choice Cascading	✓		
Menu Choice Pop-Up	✓		
Menu Choice Separator			
Button	✓		
Menu Choice	✓		
Separator			
Frame			
Text Presentation Area			
Text Component			
Push Button	✓	✓	
Text Input Field	✓	✓	String
Check Box	✓	✓	Boolean
Radio Button	✓	✓	Boolean
List Box	✓	✓	String list, integer list
Combination Box	✓	✓	String
Slider	✓	✓	Integer, boolean
Sensitive Area	✓	✓	
Sensitive Text	✓	✓	
Graphic Output Area			
Multimedia Area	✓		String (note 2)
NOTE 1: The entries in this column specify the data type of the values that are reported.			
NOTE 2: In controlled mode only.			
Abbreviations:			
Cmp = Components			

The overall structure of local actions is defined in subclause 4.8.

### 7.1.3 Relationship between objects and components

The closed state of an object overrides the opened state of a component. If an object is closed, the entire object including all its components is removed from the screen, regardless of whether its components are in the opened state or not. The opened state of an object does not override the closed state of a component. If an object gets opened, the components which are in the closed state are not displayed.

The inaccessible state parameter of an object overrides the accessible state parameter of a component. If an object is set inaccessible, the entire object including all its components becomes inaccessible, regardless of whether the components are accessible or not. The accessible state parameter of an object does not override the inaccessible state parameter of a component. If an object is set accessible, the user can only interact with those components that are accessible.

### 7.1.4 Open/Close of Sound, Video, Resource and Metacode objects

If a sound object is opened, the sound shall be performed. At the end of the sound output the object shall be considered closed.

The same rule applies to video objects. If the object is opened, the video display process starts and at its end the object is closed.

The open operation of a resource only checks if it is present at the terminal.



If a VEMMI\_Open\_Object is applied to a metacode object the terminal copies all commands of the metacode object on top of the logical command input queue. The next command read from the command input queue is the first command from the metacode. If the commands are copied, it is considered closed.

#### **7.1.5 Maximize operation**

An object can be defined as maximizable. In this case it is displayed with a maximize button. Upon activation of this button it is displayed enlarged. The enlarged object is completed with a restore button and upon activation of the restore button the object is re-displayed at its initial size. These operations are terminal and UI dependent, especially the enlarged size. Only the host application can mark an object as maximizable.

The operation described above is not mandatory for a VEMMI terminal. If a terminal does not implement this operation it shall ignore the corresponding attributes and operations.

There are two recommended strategies for enlarged presentation. The terminal is free to select locally the most appropriate.

##### **Strategy 1 (applicable to the Text Presentation Area, Text Input Field):**

All object elements and components, except the Text Presentation Area (Text Input Field), are enlarged on a terminal specific basis. The Text Presentation Area (Text Input Field) component of a Dialogue Box has a specific behaviour, in order to allow an enlarged object to add locally more text content in its display area. If the Text Presentation Area (Text Input Field) contains in its initial size the whole text, the maximize operation is terminal dependent. If the text is only partly displayed in its initial position, the terminal shall add additional text in the maximized object. The enlarged object can display the text in a new formatted form. This can be set by the host.

##### **Strategy 2 (applicable to the Text Presentation Area and/or Multimedia Area, Text Input Field):**

If the "Maximizable" attribute is set, the terminal shall offer the possibility to view the content of the component in a separate window that is resizeable by the user. If the user locally selects that option, the content of the component shall be copied in a separate window and the component of the Dialogue Box should be closed. If the user locally returns from that maximized presentation the component of the Dialogue Box should be locally opened again and if changed, the content should be copied from the new window to the component.

If an object is closed in its enlarged size and later reopened it is displayed in its initial (not enlarged) size. If a maximized object is deactivated it keeps its size.

Additional local operations on objects, which are not definable by the host can be offered to the user on a terminal specific basis. These can be: minimize, moving, iconize, unidirectional increase and decrease etc.

#### **7.1.6 Notational conventions**

In the next subclauses the following notational conventions apply:

- each object and component description contains an attribute list. These lists contain those attributes that are needed to describe the properties of the elements. The precise syntax of the element encoding is defined in clause 9;
- the identification numbers for resource objects, BIN, FIN, TIN, VIN and MIN are special cases of OINs and used on the same level, e.g. as parameters of VEMMI commands. The CIN differs from them in that it always denotes a component.

The naming conventions for the boolean attributes in the VEMMI element description follows the following rules:

- if the attribute is present in the encoding, it's boolean value corresponds to the value expressed by it's name;

- if the attribute is not present, it's default values apply (the contrary of the value expressed by the name).

EXAMPLE: The default value of the state "Opened" is true. If the "closed" attribute is present in the encoding of a VEMMI element the element shall be in the "Closed" state. If the "closed" attribute is not present in the encoding of a VEMMI element the element shall be in it's default state, the "Opened" state.

### 7.1.7 Mnemonic

When text data is used in a component, the text can contain the character "&" which marks the next character as a mnemonic key. The mnemonic key can be used to validate or activate the component. To present the "&" character in a text it shall be doubled (e.g.: text content ="&Cats && Dogs"; text presented to the user is therefore "Cats & Dogs").

### 7.1.8 Activation and Validation

On user activation a VEMMI dialogue element gets the input focus, the user inputs refer to the active element.

A validation event for Menu Choice components, Button components, Sensitive Text, Sensitive Areas, Check Boxes, Radio buttons and List Boxes is one of the following:

- user activation action followed by a confirmation of the choice (e.g.: activation with the "Tab" key, validation with the "Return" key);
- direct validation with a pointing device (e.g.: mouse-click).

For all other components the validation event is triggered on the loss of input focus.

## 7.2 Text formats

### 7.2.1 VEMMI high quality text

The implementation of text attributes (change of text font, text size, etc. inside a text string) and the definition of sensitive areas for text (sensitive text or "hot-spot") is realized with "In-text attributes". These are applicable for the following elements:

- Text Presentation Area components of a Dialogue box;
- labels.

#### 7.2.1.1 Text attributes

The attributed font can be changed inside a text output operation. The font is referenced via the FIN. A selected new font applies to all subsequent text up to the next in-text attribute which changes the font, or to the end of the component. The initial attributed font applied to the first text of the component is defined as an additional attribute in the component.

The following example shows the use of in-text attributes. The brackets <...> denote attributes of the components with their parameter.

EXAMPLE:

```
.....
<text1>           = "Customer service note:(CR,LF,CR,LF)"
<use FIN 5>
<text2>           = "110 size negatives "
<use FIN 6>
<text3>           = " shall not be cut but shall be returned in strip form for reprinting."
<use FIN 5>
<text4>           = " If they are cut we are unable to make prints from them."
<use FIN 7>
<text5>           = "(CR,LF,CR,LF)Issued by the A.P.L."
.....
```

FIN 5 is defined as ROMAN, height 12  
FIN 6 is defined as ROMAN, height 12, bold, underlined  
FIN 7 is defined as SWISS, height 8

The above text sequence with in-text attributes might look as follows:

Customer service note:

110 size negatives shall not be cut but  
shall be returned in strip form for  
reprinting. If they are cut we are unable  
to make prints from them.

Issued by the A.P.L.

#### 7.2.1.2 In-text attributes for the definition of sensitive text

Each text string defined as sensitive has the reference to a Sensitive Text component. This component contains two attributes which define the activation and validation operations to be performed upon the user interaction with the Sensitive Text.

The two groups of attributes can be mixed in one text component, so that the application can apply text attributes, e.g. colour, to sensitive strings. Such a string may appear green and possibly in addition highlighted, in order to be easy distinguished from the plain text, which might be black on white. With these attributes the application can offer the user "hypertext"-like output with VEMMI.

#### 7.2.2 Text labels and titles

The text contents of labels and titles are clipped at the borders of their containers. This shall be applied to the following attributes:

- all title attributes;
- all label attributes;
- text content attribute of Menu Choice components;
- list text attribute of List Box and Combination Box components.

There shall be no CR + LF in the text content of titles and labels.

### 7.3 The Application Bar

The Application Bar is subdivided into three different logical groups of Menu Choice components. These groups differ in their behaviour and functionality. The three different groups are named:

- Bar;
- Pull-Down Menu;
- Cascading Menu.

The Bar is a horizontal or vertical list of Menu Choice components. The Pull-Down Menus are vertical lists of Menu Choice components associated with the same Menu Choice component of the Bar. The Cascading Menus are vertical lists of Menu Choice components associated with the same Menu Choice component of the Pull-Down Menu. The Bar is implicitly positioned in the top left corner of the DDA. A displacement can be selected for vertical Bars.

All Menu Choice components of the Application Bar shall only have text content. They are generally positioned and dimensioned implicitly by their content. The terminal may ignore CR+LF that are part of the text content of a Menu Choice component.

A Menu Choice Separator component can be used to visually separate Menu Choice components in a logical group. It is defined by setting the attribute "Separator" with the definition of a regular Menu Choice.

The structure of the Application Bar Object is given, by the canonical descending order of its components (path left to right with priority to the depth).

In the following Application Bar example the canonical descending order of the components is: (AB(FG(IJK)H)CDE(L))

A	B	C	D	E
	F			L
	G	I		
	H	J		
		K		

NOTE: The use of parenthesis in the above example is only to show the different levels of encapsulation of the different object groups for the path description.

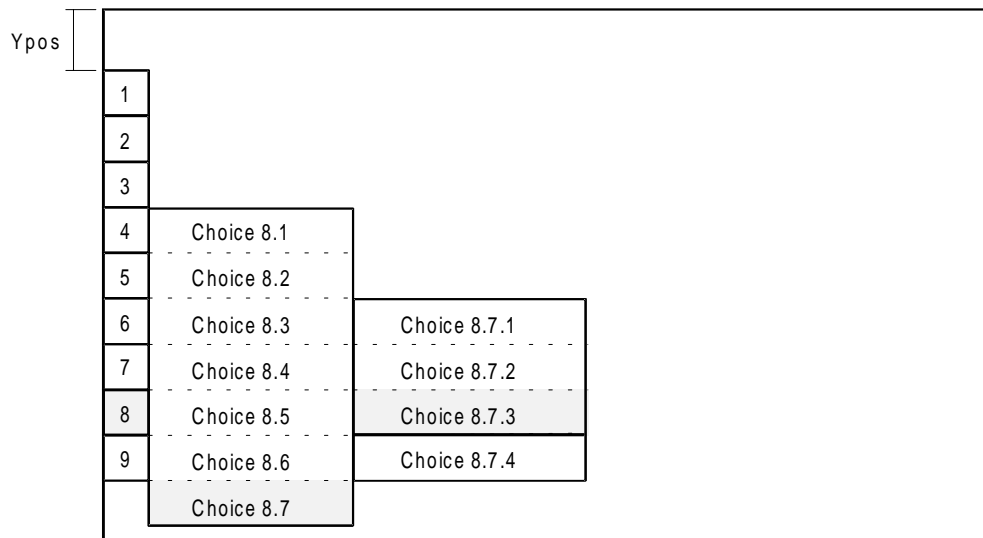
The description of the structure of the object is then:

Object: Application Bar  
 Component: Menu Choice Bar A  
 Component: Menu Choice Bar B  
 Component: Menu Choice Pull-Down F  
 Component: Menu Choice Pull-Down G  
 Component: Menu Choice Cascading I  
 Component: Menu Choice Cascading J  
 Component: Menu Choice Cascading K  
 Component: Menu Choice Pull-Down H  
 Component: Menu Choice Bar C  
 Component: Menu Choice Bar D  
 Component: Menu Choice Bar E  
 Component: Menu Choice Pull-Down L

General visual aspect (see figures 7 and 8):

Choice 1	Choice 2	Choice 3	Choice 4	Choice 5
		Choice 3.1		
		Choice 3.2		
		Choice 3.3		
		Choice 3.4		
		Choice 3.5	Choice 3.7.1	
		Choice 3.6	Choice 3.7.2	
		Choice 3.7	Choice 3.7.3	
		Choice 3.8	Choice 3.7.4	
		Choice 3.9	Choice 3.7.5	

Figure 7: Horizontal Application Bar



**Figure 8: Vertical Application Bar**

Attributes:

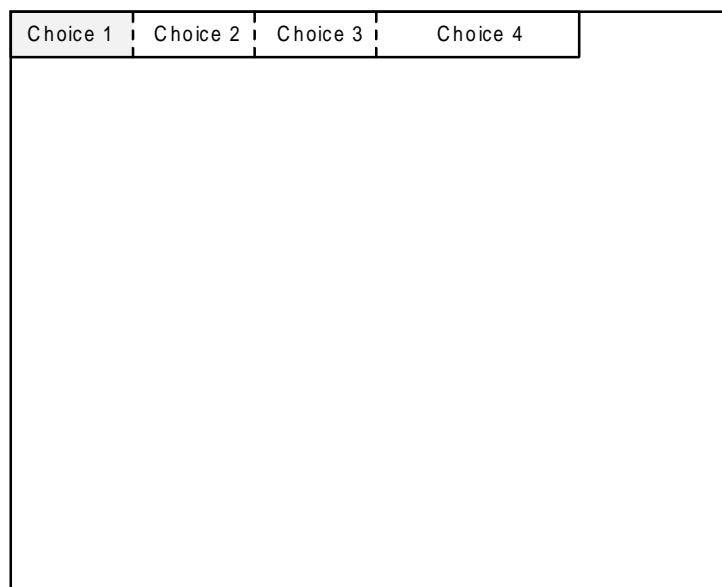
- YPos: This attribute carries the vertical displacement of the element in NDC. It is only applicable to vertical Application Bars.
- Closed: The element shall be in the closed state.
- NotAccessible: The object is not accessible.
- DefActive: This attribute carries the CIN of the Menu Choice which is active by default, the first time the object is opened.
- Vertical: The Application Bar shall be presented vertical. The vertical representation of the Application Bar is an optional feature for a VEMMI terminal. If the vertical attribute is present and set true in the encoding of an Application Bar, the terminal may ignore it and present the Application Bar horizontally. However, no error message shall be send to the host.

### **7.3.1 Composition**

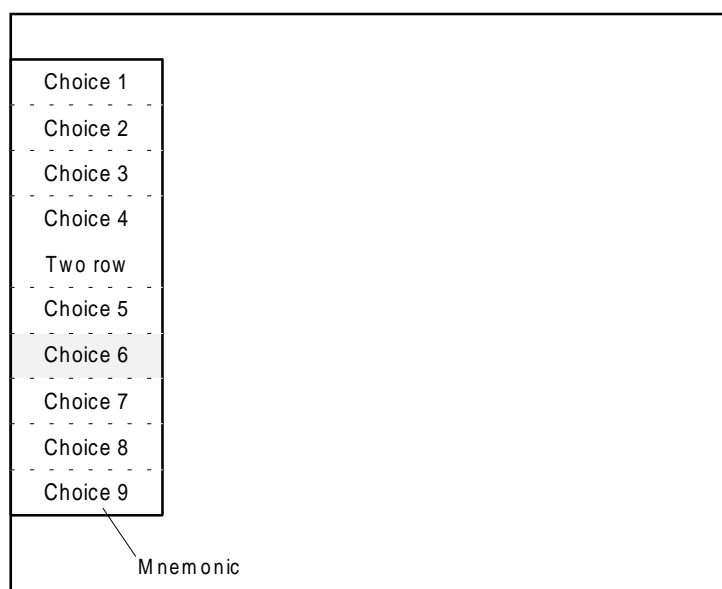
#### **7.3.1.1 Menu Choice components of the bar**

- description: The bar is a consecutive list of Menu Choices positioned side by side horizontally, or vertically. If the space requirements of the Bar are greater than the space available the terminal should either apply a line-fold mechanism or clip each menu choice. However, all menu choices should remain visible;
- behaviour: When the Application Bar is active, the terminal shall emphasize one Menu Choice, and offer shifting and validation facilities to the user. A mnemonic key can be used to validate the Menu Choice (see subclause 7.1.7). If a Menu Choice of the Bar is validated the associated local action shall be performed and then the associated Pull-Down Menu shall be opened (the Menu Choice remains emphasized);
- Interactive functionality:
  - a) shifting;
  - b) activation;
  - c) validation.

Visual aspect (see figures 9 and 10):



**Figure 9: Horizontal Menu Bar**



**Figure 10: Vertical Menu Bar**

Attributes:

- CIN: This attribute carries the Component Identification Number;
- NotAccessible: The element shall not be accessible;
- Text: This attribute carries the text content of the component;
- LocActAct: This attribute carries the code for the local action which is associated to the component and triggered by its activation;
- LocActVal: This attribute carries the code for the local action which is associated to the component and triggered by its validation.

#### 7.3.1.2 Menu Choice components of the Pull-Down Menu

- description: The Pull-Down Menu is a consecutive list of Menu Choices components associated with the same Menu Choice of the Bar and presented vertically on several rows.
  - a) the Pull-Down Menu shall be positioned next to the associated Menu Choice of the Bar;

- b) a Separator can be used in an Pull-Down Menu to visually separate different Menu Choices.
- behaviour: When the Pull-Down Menu is active, the terminal shall emphasize one Menu Choice, and offer shifting and validation facilities to the user. A mnemonic key can be used to validate the Menu Choice (see subclause 7.1.7). If a Menu Choice is validated the associated local action shall be performed and then the associated Cascading Menu shall be opened (the Menu Choice remains emphasized). If no Cascading Menu is associated to the Menu Choice after a user validation, the associated local action shall be performed and then the Pull-Down Menu shall be closed.
- interactive functionality:
  - a) shifting;
  - b) activation;
  - c) validation.

Visual aspect (see figures 11 and 12):

Choice 1	Choice 2	Choice 3	Choice 4	Choice 5
	Choice 2.1			
	Choice 2.2			
	Two row Choice 2.3			
	Choice 2.4			
	Choice 2.5			

Figure 11: Pull-Down Menu

Choice 1	
Choice 2	Choice 2.1
Choice 3	Choice 2.2
Choice 4	Choice 2.3
Two row	Choice 2.4
Choice 5	Two row
Choice 6	Choice 2.5
Choice 7	
Choice 8	
Choice 9	

Figure 12: Pull-Down Menu

Attributes:

- CIN: This attribute carries the Component Identification Number;
- NotAccessible: The element shall not be accessible;
- Text: This attribute carries the text content of the component;

- Separated: A Separator shall be drawn to separate the element from the element above;
- LocActAct: This attribute carries the code for the local action which is associated to the component and triggered by its activation;
- LocActVal: This attribute carries the code for the local action which is associated to the component and triggered by its validation.

### 7.3.1.3 Menu Choice components of the Cascading Menu

Menu Choice components of a Cascading Menu are common to the Pull-Down Menu and the Pop-Up Menu.

- description: The Cascading Menu is a consecutive list of Menu Choices associated with the same Menu Choice of the Pull-Down Menu or Pop-Up Menu and presented vertically on several rows.
  - a) the width allocated for each Menu Choice shall be the same;
  - b) the Cascading Menu shall be positioned next to the associated Pull-Down Menu or Pop-Up Menu;
  - c) a Separator can be used in a Cascading Menu to visually separate different Menu Choices. Figure 13 shows the recommended presentation for a Cascading Menu. If the terminal is not able to present a Cascading Menu as shown in figure 13 it can, as a fallback solution, present the Menu Choices of the Cascading Menu in the Pull-Down Menu but it should visually associate the Menu Choices of the Cascading Menu to the corresponding Menu Choice of the Pull-Down Menu. The terminal should set the Menu Choice in the Pull-Down Menu, to which the Cascading Menu is associated, inaccessible and present it as a title for the Menu Choices of the Cascading Menu (see figure 14).
- behaviour: When the Cascading Menu is active, the terminal shall emphasize one Menu Choice, and offer shifting and validation facilities to the user. A mnemonic key can be used to validate the Menu Choice (see subclause 7.1.7). After the validation of a Menu Choice the associated local action shall be performed and then the Cascading Menu and the Pull-Down Menu (Pop-Up Menu) shall be closed.
- Interactive functionality:
  - a) shifting;
  - b) activation;
  - c) validation.

Visual aspect (see figure 13 and 14):

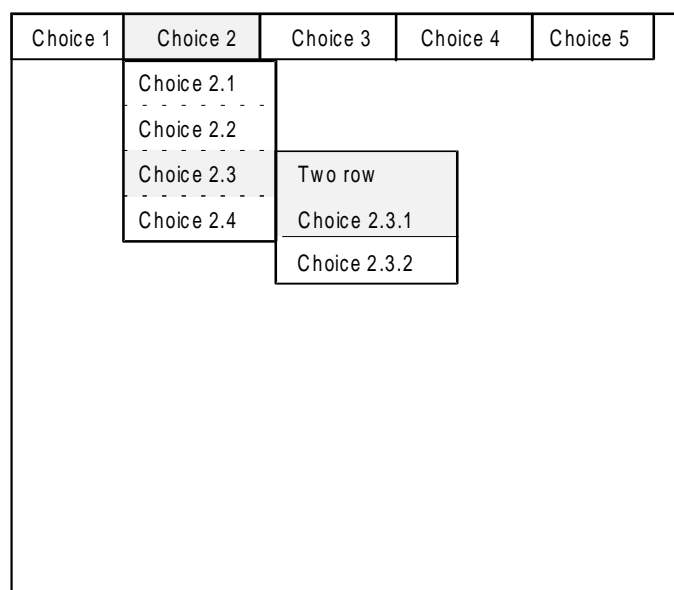
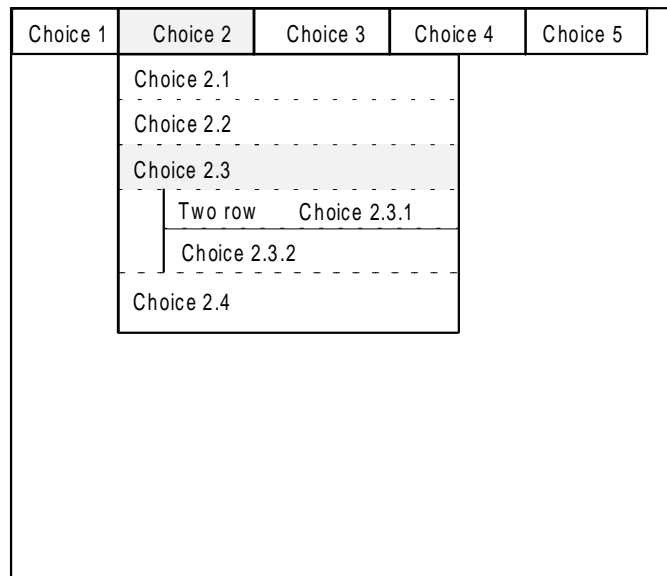


Figure 13: Cascading Menu (recommended presentation)





**Figure 14: Cascading Menu (fallback presentation)**

Attributes:

- CIN: This attribute carries the CIN;
- NotAccessible: The element shall not be accessible;
- Text: This attribute carries the text content of the component;
- Separated: A Separator shall be drawn to separate the element from the element above;
- LocActAct: This attribute carries the code for the local action which is associated to the component and triggered by its activation;
- LocActVal: This attribute carries the code for the local action which is associated to the component and triggered by its validation.

#### **7.4 The Button Bar**

- description: The Button Bar is a consecutive list of Buttons positioned side by side horizontally, or vertically.
  - a) horizontal representation: the height allocated for each Button shall be the same (the height of the biggest button);
  - b) vertical representation: the width allocated for each Button shall be the same (the width of the biggest button);
  - c) a text title may be given to the Button Bar which shall be displayed directly above the buttons. The title height should equal to 1/32 NDC.
- behaviour: When the Button Bar is active, the terminal shall emphasize one Button, and offer to the user shifting facilities from one button to another.
- interactive functionality:
  - a) shifting;
  - b) activation.

Visual aspect (see figure 15):

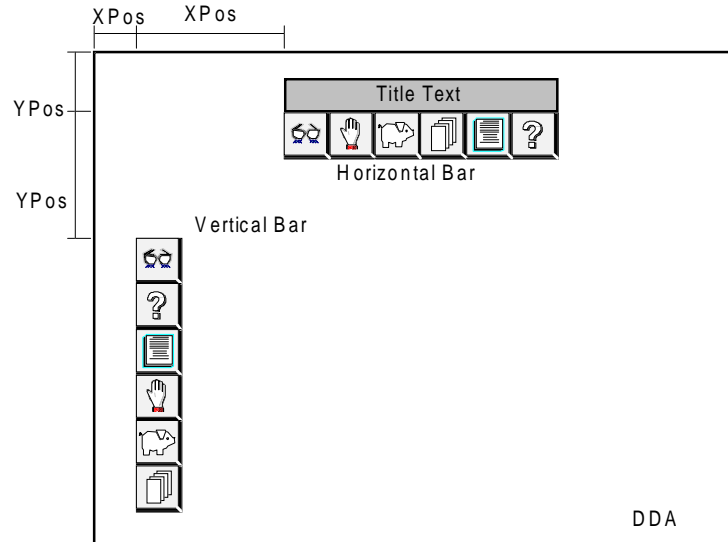


Figure 15: Button Bar

Attributes:

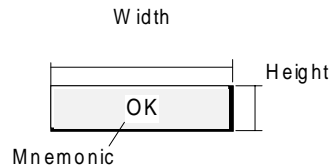
- XPos: This attribute carries the horizontal position of the element in NDC;
- YPos: This attribute carries the vertical position of the element in NDC;
- Vertical: Boolean switch between the alternatives;
- FirstActive: This attribute carries the CIN of the Button which is active by default, the first time the object is opened;
- Modal: The element shall be modal;
- Closed: The element shall be in the closed state;
- NotAccessible: The element shall not be accessible;
- Title: This attribute carries the text title of the object.

## 7.4.1 Composition

### 7.4.1.1 The Button component

- description: The Button is a rectangular area in the DDA.
  - a) a button can have either text content or graphic content. If formed with text data it should be represented by a text label, possibly associated with a terminal dependent graphic to draw the shape of the button. The button graphic shape shall be included in the space allocated by the VEMMI application for the whole Button component. If formed with graphic data and if the graphic shape does not cover the entire space allocated for the button the terminal can resize the content. The terminal may centre the text data in the element display area reserved by the VEMMI application for the component.
  - b) the drawing of the Button and the possible visual trigger effect are terminal dependent.
- behaviour: When a Button is active, the terminal shall emphasize it. A mnemonic key can be used - to validate the Button (see subclause 7.1.7). When a trigger effect is implemented the Button shall go back to the initial display state after user validation.
- Interactive functionality:
  - a) activation;
  - b) validation.

Visual aspect (see figure 16):



**Figure 16: Button**

Attributes:

- CIN: This attribute carries the Component Identification Number;
- Height: This attribute carries the height of the component. If not present the corresponding value of the previously defined Button shall be applied. If the attribute is not present at the first button of a Button Bar the default value shall be applied;
- Width: This attribute carries the width of the component. If not present the corresponding value of the previously defined Button shall be applied. If the attribute is not present at the first button of a Button Bar the default value shall be applied;
- Closed: The element shall be in the closed state;
- NotAccessible: The element shall not be accessible;
- BIN: The BIN of the component;
- Text: The text content of the component;
- LocActAct: This attribute carries the code for the local action which is associated to the component and triggered by its activation;
- LocActVal: This attribute carries the code for the local action which is associated to the component and triggered by its validation.

## 7.5 The Pop-Up Menu

The Pop-Up Menu is subdivided into two different logical groups of Menu Choice components. These groups differ in their behaviour and functionality. The two different groups are named:

- Primary Pop-Up Menu;
- Cascading Menu.

The Primary Pop-Up Menu is a vertical list of Menu Choice components. The Cascading Menu is a vertical list of Menu Choice components associated to the same Menu Choice component of the Primary Pop-Up Menu.

A Menu Choice Separator can be used to visually separate menu choice components in a logical group. It is defined by setting the attribute "Separator" with the definition of a regular Menu Choice.

The structure of the Pop-Up Menu Object is given, with the following conventions, by the canonical descending order of its components (path top down).

In the following Pop-Up Menu example the canonical descending order of the components is: (BFG(IJK)H)

B	
F	
G	I
H	J
	K

NOTE: The parenthesis in the above example are only for showing the different levels of encapsulation of the different object groups for the path description.

The description of the structure of the object is then:

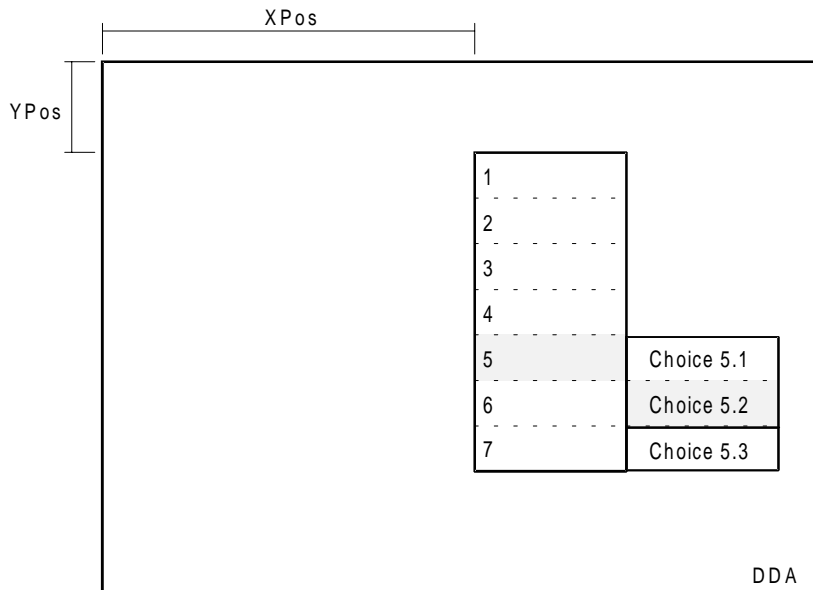
Object: Pop-Up Menu;  
Component: Menu Choice Pop-Up B;

Component: Menu Choice Pop-Up F;  
Component: Menu Choice Pop-Up G;  
Component: Menu Choice Cascading I;  
Component: Menu Choice Cascading J;  
Component: Menu Choice Cascading K;  
Component: Menu Choice Pop-Up H.

- Interactive functionality:

- a) shifting;
- b) activation.

General Visual Aspect (see figure 17):



**Figure 17: Pop-Up Menu**

A text title may be given to the Pop-Up Menu which shall be displayed in its first row of the Primary Pop-Up Menu.

Attributes:

- XPos: This attribute carries the horizontal position of the element in NDC;
- YPos: This attribute carries the vertical position of the element in NDC;
- Title: This attribute carries the title of the object. The title shall be displayed in the first row of the object;
- TitleFont: This attribute carries the FIN of the title;
- FirstActive: This attribute carries the CIN of the component which is active by default, the first time the object is opened;
- Modal: The element shall be modal;
- Closed: The element shall be in the closed state;
- NotAccessible: The element shall not be accessible.

## 7.5.1 Composition

### 7.5.1.1 Menu Choice components of the Primary Pop-Up Menu

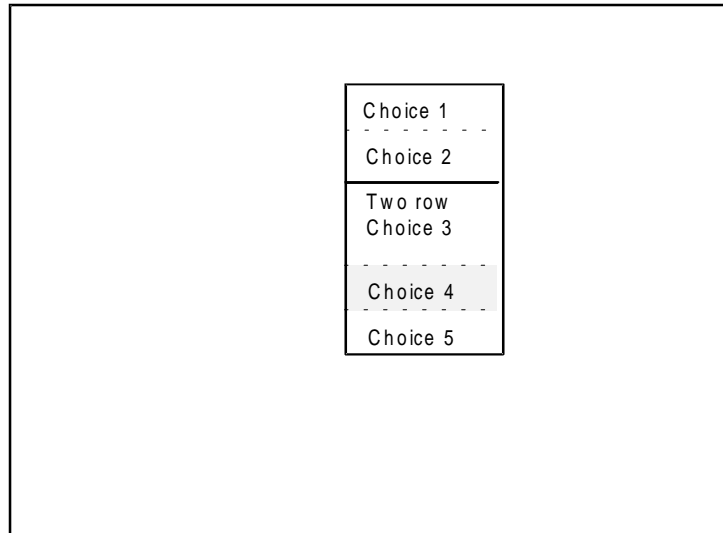
- description: The Primary Pop-Up Menu is a consecutive list of Menu Choices and presented vertically on several rows. A Separator can be used in a Primary Pop-Up Menu to visually separate different Menu Choices;
- behaviour: When the Primary Pop-Up Menu is active, the terminal shall emphasize one Menu Choice, and offer to the user shifting and validation facilities. A mnemonic key can be used to validate the Menu Choice (see subclause 7.1.7). If a Pop-Up Menu component is validated the associated local action shall be performed and then the associated Cascading Menu shall be

opened (the Pop-Up Menu component remains emphasized). If no Cascading Menu is associated to the Menu Choice after a user validation, the associated local action shall be performed and then the Pop-Up Menu shall be closed;

- Interactive functionality:

- a) shifting;
- b) activation;
- c) validation.

**Visual aspect (see figure 18):**



**Figure 18: Primary Pop-Up Menu**

Attributes:

- CIN: This attribute carries the Component Identification Number;
- Closed: The element shall be in the closed state;
- NotAccessible: The element shall not be accessible;
- Text: This attribute carries the text content of the component;
- Separated: The elements shall be separated;
- LocActAct: This attribute carries the code for the local action which is associated to the component and triggered by its activation;
- LocActVal: This attribute carries the code for the local action which is associated to the component and triggered by its validation.

#### **7.5.1.2 Menu Choice components of the Cascading Menu**

- description, behaviour, interactive functionality, attributes:

See subclause 7.3.1.3.

Visual aspect (see figure 19):

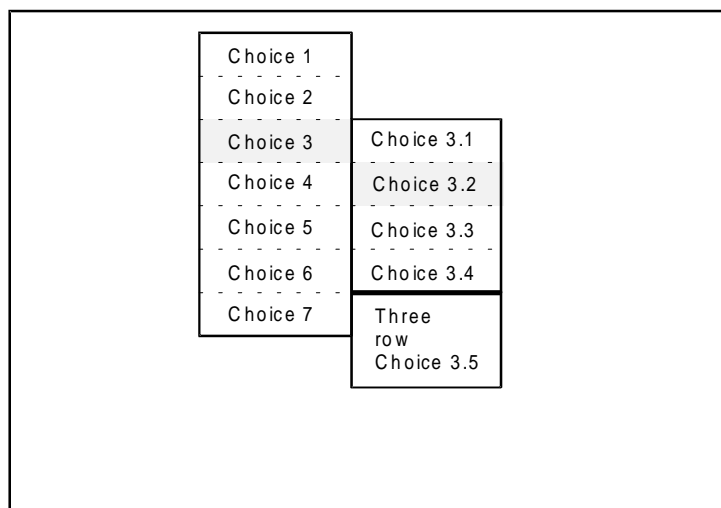


Figure 19: Recommended presentation of a Pop-Up Menu with associated Cascading Menu

## 7.6 The Dialogue Box

- description: The Dialogue Box is a rectangular area in the DDA which contains VEMMI components in order to establish user interaction;

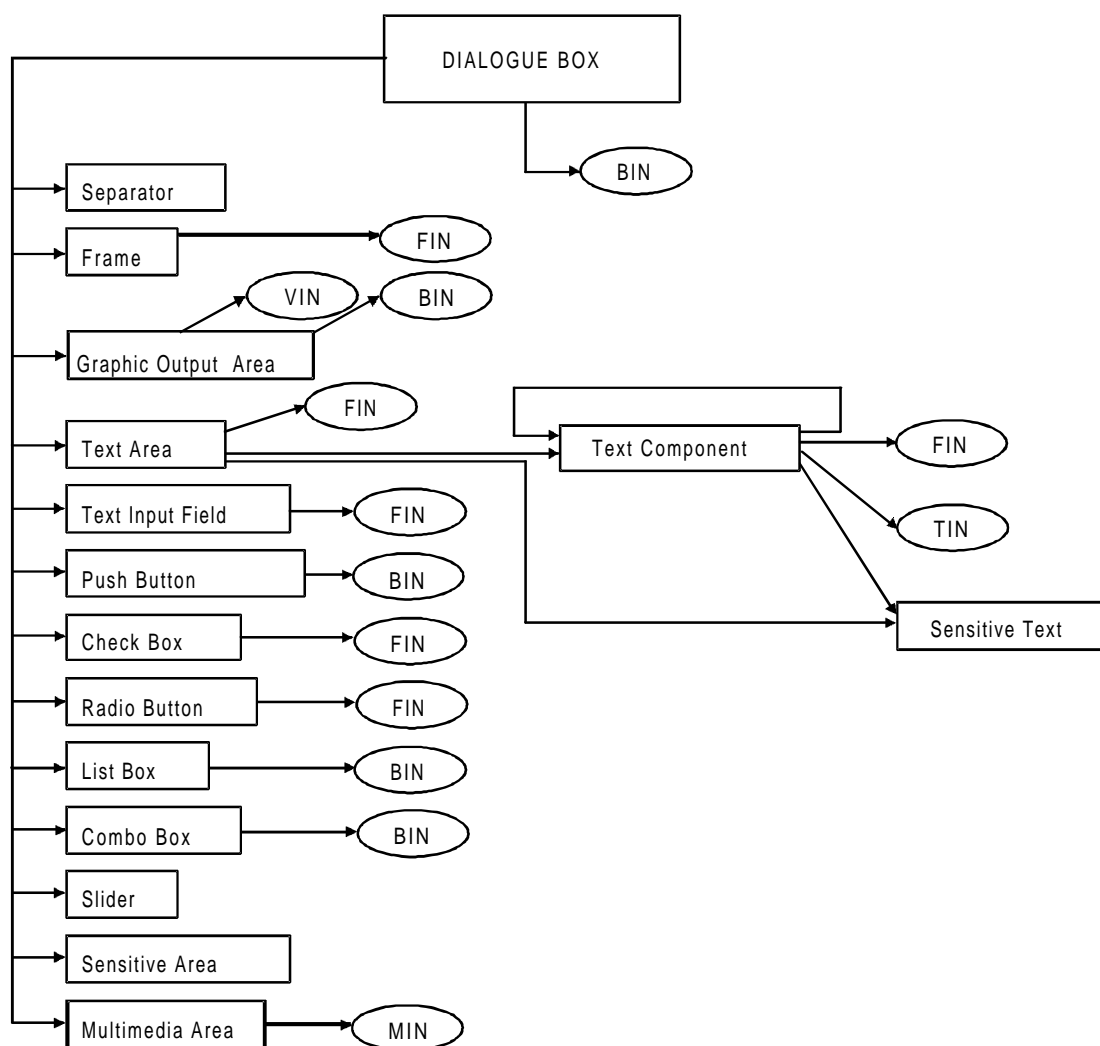


Figure 20: Element relationships within the Dialogue Box

- a) the components themselves contain (or have references to ) other components or resource objects. Figure 20 shows the relationship. Each arrow indicates a possible reference to a component or a resource object. These references can be multiple as well as the dialogue box can contain several components of the same type;
  - b) a text title may be given to the Dialogue Box which shall be displayed in its first row. The title height equals 1/32 NDC;
  - c) the Dialogue Box can be provided with a border area which should be equal to one character position. When present, this border shall be included in the dimensions defined by the VEMMI application. When a border is requested, a frame shall be drawn by the terminal. This frame drawing is terminal dependent (it should not be wider than 1/160 NDC);
  - d) the object origin of the Dialogue Box is the top left corner of the box rectangle, independent whether the box has a title or not. The components are positioned relatively to this point for Dialogue Boxes that do not have a text title. In the case of boxes with a text title this point is moved downwards by a distance corresponding to the titles height and the borders width and horizontally by the distance equal to the border width;
  - e) The background area of the Dialogue Box is the rectangle defined by the box dimensions, except the title bar and the border. The background can be coloured uniform or with a data content.
- behaviour: The user can activate any component with a pointing device or with the keyboard. If the user activates the components with the keyboard keys providing the functionality "Previous" and "Next", the order of activation should correspond to the CIN order of the components of the Dialogue Box;
  - Interactive functionality:
    - a) activation;
    - b) maximisation.

Visual aspect (see figure 21):

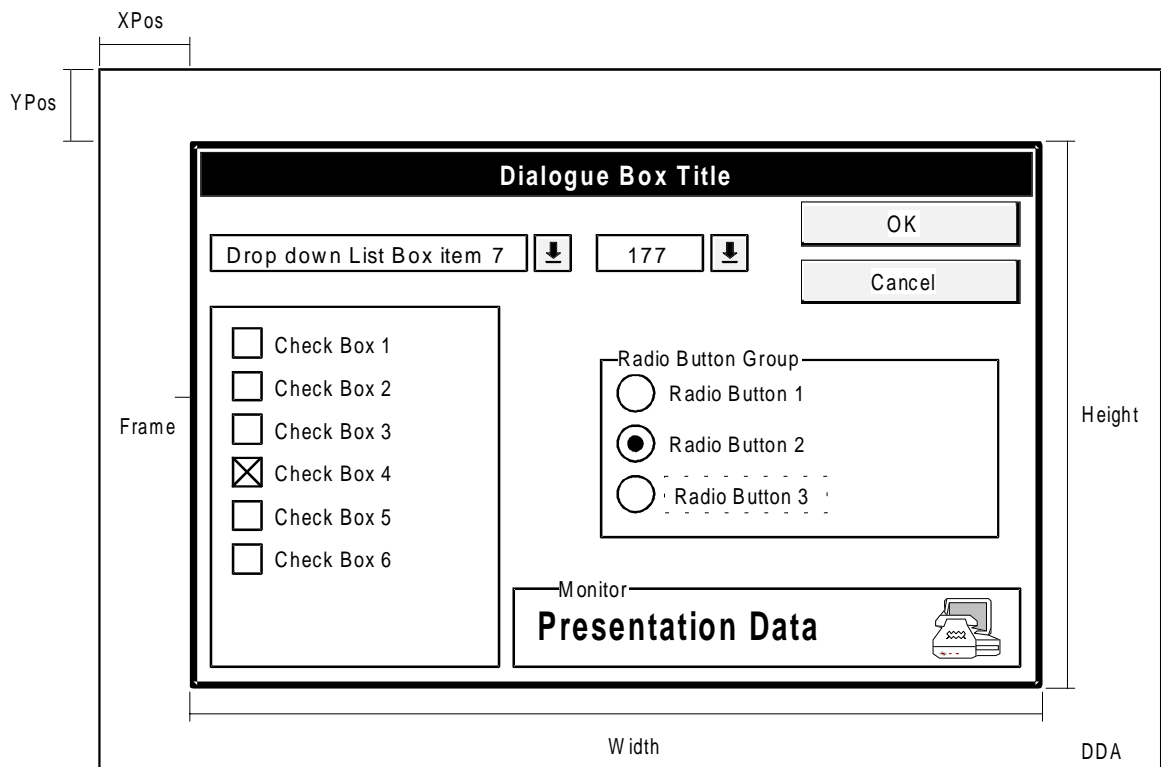


Figure 21: Dialogue Box

Attributes:

- XPos: This attribute carries the horizontal position of the element in NDC;
- YPos: This attribute carries the vertical position of the element in NDC;

- Width: This attribute carries the width of the object;
- Height: This attribute carries the height of the object;
- NoBorder: No border shall be drawn;
- Title: This attribute carries the title of the object. The title shall be displayed in the first row of the object;
- FirstActive: This attribute carries the CIN of the component which is active by default, the first time the object is opened;
- Modal: The element shall be modal;
- Closed: The element shall be in the closed state;
- NotAccessible: The element shall not be accessible;
- StoreCreationValues: The terminal shall memorise the initial values of the components at the time of the component creation (either object creation or addition of a new component to an existing object). If a local action "restore initial values" is applied the these stored values shall be restored;
- Maximizable: The object is maximizable;
- Colour: Colour index of the background;
- BIN: Bitmap identification number to fill the Dialogue Box Background;
- DispType: Specifies the display mode of the background bitmap: centred, stretched (default) or tiled.

## 7.6.1 Composition

### 7.6.1.1 The Separator component

- description: The Separator is either a horizontal or a vertical solid line (thickness e.g.: 1/160 NDC)
- behaviour: The Separator shall be inaccessible;
- interactive functionality:

a) none.

Visual aspect (see figure 22):

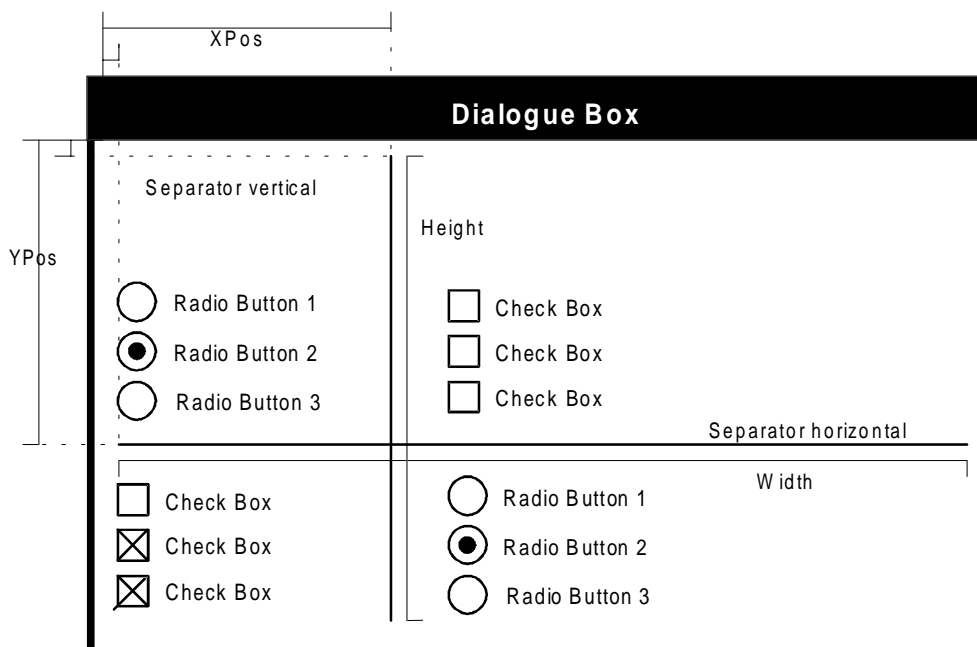


Figure 22: Separator

Attributes:

- CIN: This attribute carries the Component Identification Number;
- XPos: This attribute carries the horizontal position of the element in NDC;
- YPos: This attribute carries the vertical position of the element in NDC;
- Vertical: The element shall be drawn vertical;
- Height: This attribute carries the height of the component. It is only applicable to vertical Separators;



- Width: This attribute carries the width of the component. It is only applicable to horizontal Separators;
- Closed: The element shall be in the closed state;
- Colour: Colour index.

#### 7.6.1.2 The Frame component

- description: A Frame consists of four solid lines (thickness e.g.: 1/160 NDC) which visually separate a rectangular area of the Dialogue Box. The Frame can have an associated title. A terminal may ignore the LabelFont attribute of the frame component;
- behaviour: The Frame shall be inaccessible;
- interactive functionality:

a) none.

Visual aspect (see figure 23):

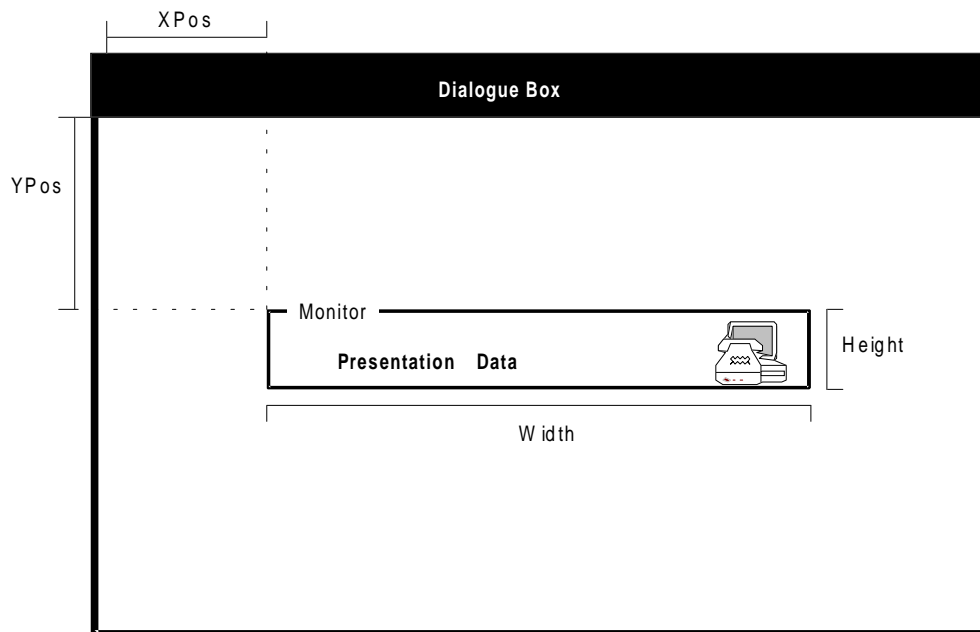


Figure 23: Frame

Attributes:

- CIN: This attribute carries the Component Identification Number;
- XPos: This attribute carries the horizontal position of the element in NDC;
- YPos: This attribute carries the vertical position of the element in NDC;
- Width: This attribute carries the width of the component;
- Height: This attribute carries the height of the component;
- Closed: The element shall be in the closed state;
- TextLabel: This attribute carries the text label of the component;
- LabelFont: Specifies the FIN for the text label;
- Colour: Colour index.

#### 7.6.1.3 The Text Presentation Area component

- description: The Text Presentation Area is an area intended to present text data to the user. Basically, this element is used to present text that fits in one window (the space defined by one text area component) or to present large text data where the user has to use scrolling tools in order to navigate through the text. The Text Presentation Area has its starting location in the top left corner of the rectangle defined by the component and the text is displayed row by row from the left to the right. The text content can contain "In-Text" attributes to switch between different attribute fonts and to define "sensitive text strings". The "In-Text" attributes can appear multiple, in any order, and they

are evaluated during the display operation sequentially and on the basis of structural information contained in text components ("previous, next" attributes).

- scrolling, Border: If the space needed for the presentation of the data is bigger than the space allocated for the component, vertical scrolling tools shall be provided by the terminal application. The terminal may present:
  - a) scrolling tools next to the right side of the area;
  - b) reduced scrolling tools as buttons overlapping the area;
  - c) cursor keys that provide the scrolling functionality.

The space needed for the scrolling tools is included in the overall dimensions of the component.

The component can be provided with a border area. When present, the border shall be included in the overall dimensions of the component. When a border is requested a frame shall be drawn by the terminal. The frame drawing is terminal dependent. It should not be wider than 1/160 NDC. If scrolling tools are provided next to the right side of the component the border shall be extended to include them visually in the box.

- Maximize operation: The size of the Dialogue Box can be changed by the user with the scrolling tools provided by the local Graphical User Interface (GUI) and via a defined maximize button. The recommended strategy is described in subclause 7.1.5. These are default recommendations. If the attribute "NoFormat" is selected and the terminal applies strategy 1, the following applies:
  - a) increasing the width should not change the text form, no additional text should be added;
  - b) decreasing the width should clip the text on the new border;
  - c) increasing the height should add new text lines;
  - d) decreasing the height should clip the text;
  - e) if the attribute "MaximTxt" is set the text shall be increased and decreased proportionally.
- Interactive functionality:
  - a) activation and validation of sensitive text, defined via in-text attributes.

NOTE: To transfer a large amount of text data it is recommended not to use the direct text definition but to use the Text Resource object and transmit the data using the VEMMI resource-file transfer.

Visual aspect (see figure 24):

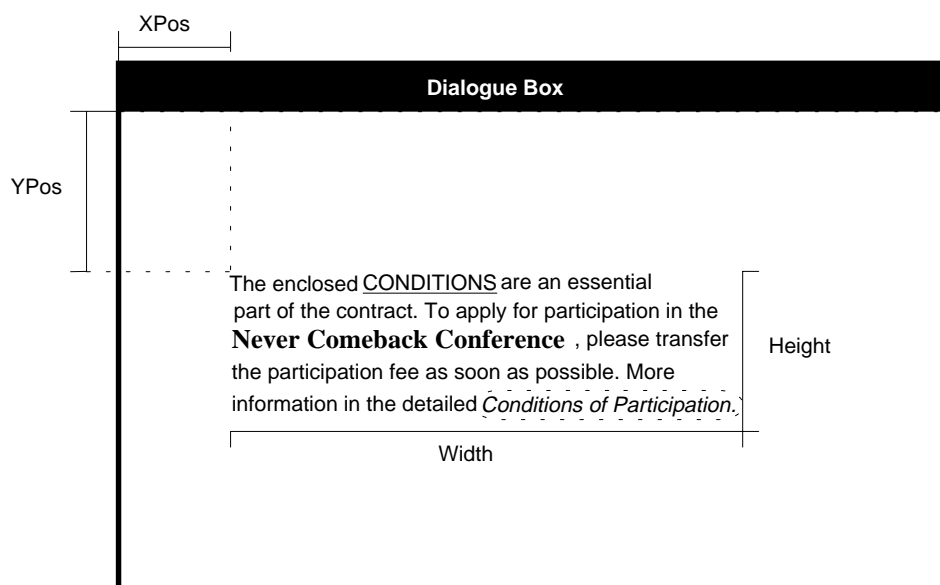


Figure 24: Text Presentation Area

Attribute:

- CIN: This attribute carries the Component Identification Number;
- XPos: This attribute carries the horizontal position of the element in NDC;
- YPos: This attribute carries the vertical position of the element in NDC;
- Width: This attribute carries the width of the component;
- Height: This attribute carries the height of the component;
- NoScrollingTools: No scrolling tools shall be drawn by the terminal;
- NoFormat: There are no formatting rules for the maximization of the dialogue box;
- MaximTxt: Local maximization can maximize the text of the component;
- NoBorder: No Border shall be drawn to frame the element;
- InitialFnt: FIN of the attribute font to be applied on the first text;
- InText: Text content possibly including in-text attributes, and references to font objects;
- TextCompRef: Reference to a text component;
- Closed: The element shall be in the closed state.

#### 7.6.1.4 The Text component

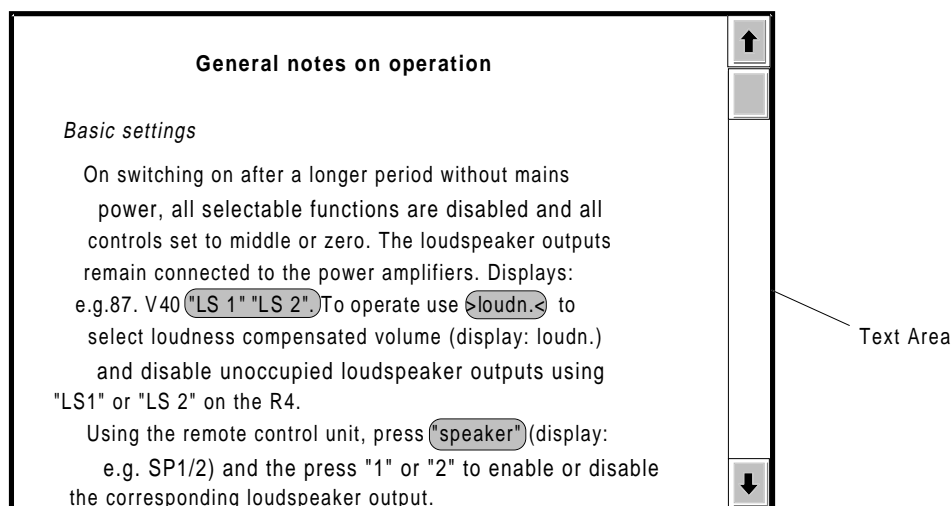
- description: This component is used to split large text data in units (text components) and to define the necessary structural information (concatenation of the text components) in order to be displayed in a text area. This structural information references the previous and next text components, defined with the attributes "NextText" and "PreviousText";
- display concept: The components are not necessarily of the same length. During the display of concatenated text components, the user should have the impression of a continuous text. The continuity should not be interrupted by a switch from one component to the other. Vertical scrolling capabilities, on a row-by-row and page-by-page basis, shall be offered within consecutive components. This display continuity is interrupted in two cases:
  - a) a component contains no further concatenation, no "NextText" during a forward scrolling or no "PreviousText" during a backward scrolling. The display does not continue because the just displayed text pertains to a topic which has reached its logical end. Figure 29 shows two sequences of concatenated components: CINs 7,8,9 and CIN 17;
  - b) "Jumps", due to the validation of sensitive text as part of the displayed components or other interactive components of the dialogue box. These components can contain as a local action an element specific command, e.g. "open component of parent object". In order to display the text component in a specific text area the local actions shall contain the "open component" command (encoding: specific command type is equal to "100", followed by two CINs) with two CINs, the first specifying the text area the second specifying the text component. The referenced parent component is a text component which will be displayed in the starting position and not consecutively to the last displayed component. This effect is shown in figures 27 and 28.

The text to be displayed is contained in the component definition, in another referenced text component or in a referenced text resource object. If a referenced resource object has not been created in the terminal, it can be requested from the host application with the command VEMMI\_Object\_Retransmission with attribute RequestType set to resource. If the text file content of the resource object is not available, the terminal can request the file from the server using VEMMI\_Object\_Retransmission with attribute RequestType set to file.

If such a retransmission request is performed as a result of a user interaction (when the object or content should be displayed), rule B for retransmission shall be applied (see subclause 5.2.2). If enabled by its storage capabilities, the VEMMI terminal may decide to request the transmission of resource objects or contents adjacent to the one currently displayed, anticipating user scrolling. In this case, until the requested object or text file needs to be displayed to the user, the rule A for retransmission shall be applied.

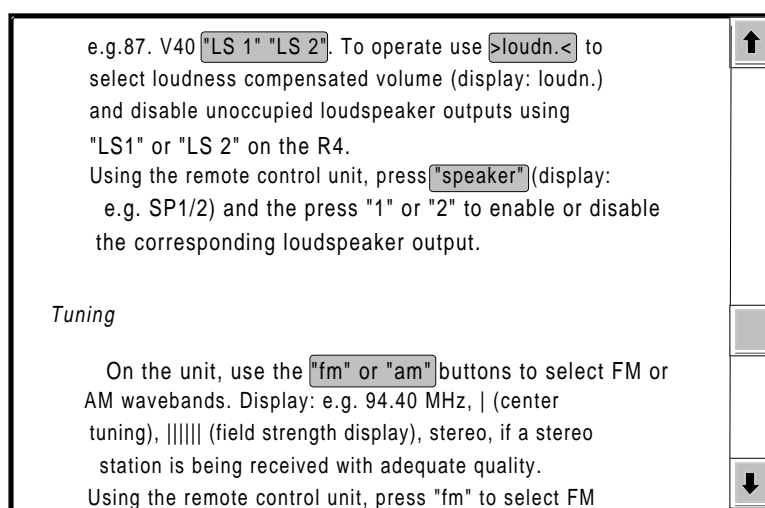
NOTE: To transfer a large amount of text data it is recommended not to use the direct text definition but to use the Text Resource object and transmit the data using the VEMMI resource-file transfer. Local destruction and retransmission of text components are also permitted (see subclause 5.2.2).

Visual aspect:



**Figure 25: A text area with text component(s), scroll bars and sensitive texts (Attribute "NoFormat" is not set)**

Now, the user becomes active and scrolls downward:



**Figure 26**

In addition the user enlarges the window on both directions:

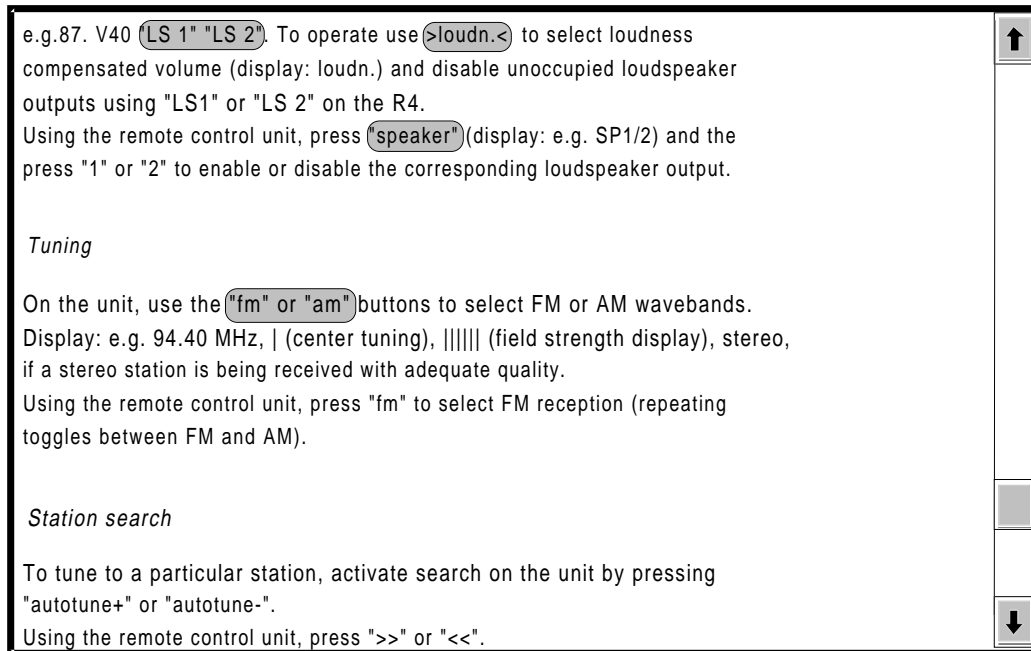


Figure 27

And finally, clicks on the hot spot "speaker":

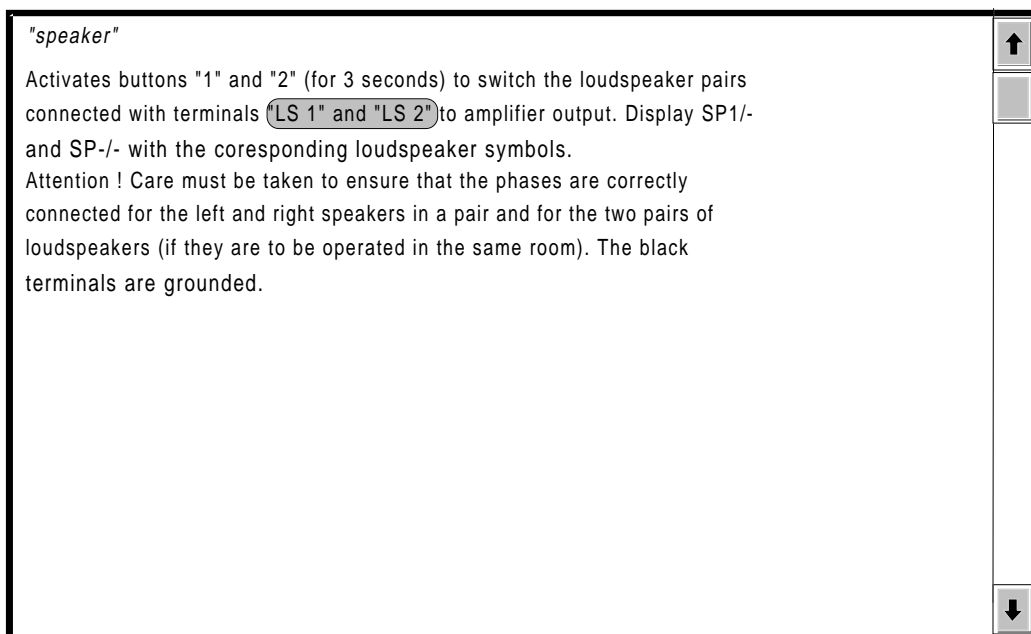


Figure 28

The "speaker" text is displayed.

A possible component architecture (text area component with CIN=10) for the display sequence showed if figures 25 to 29 can be the following (the in-text attributes are not listed):

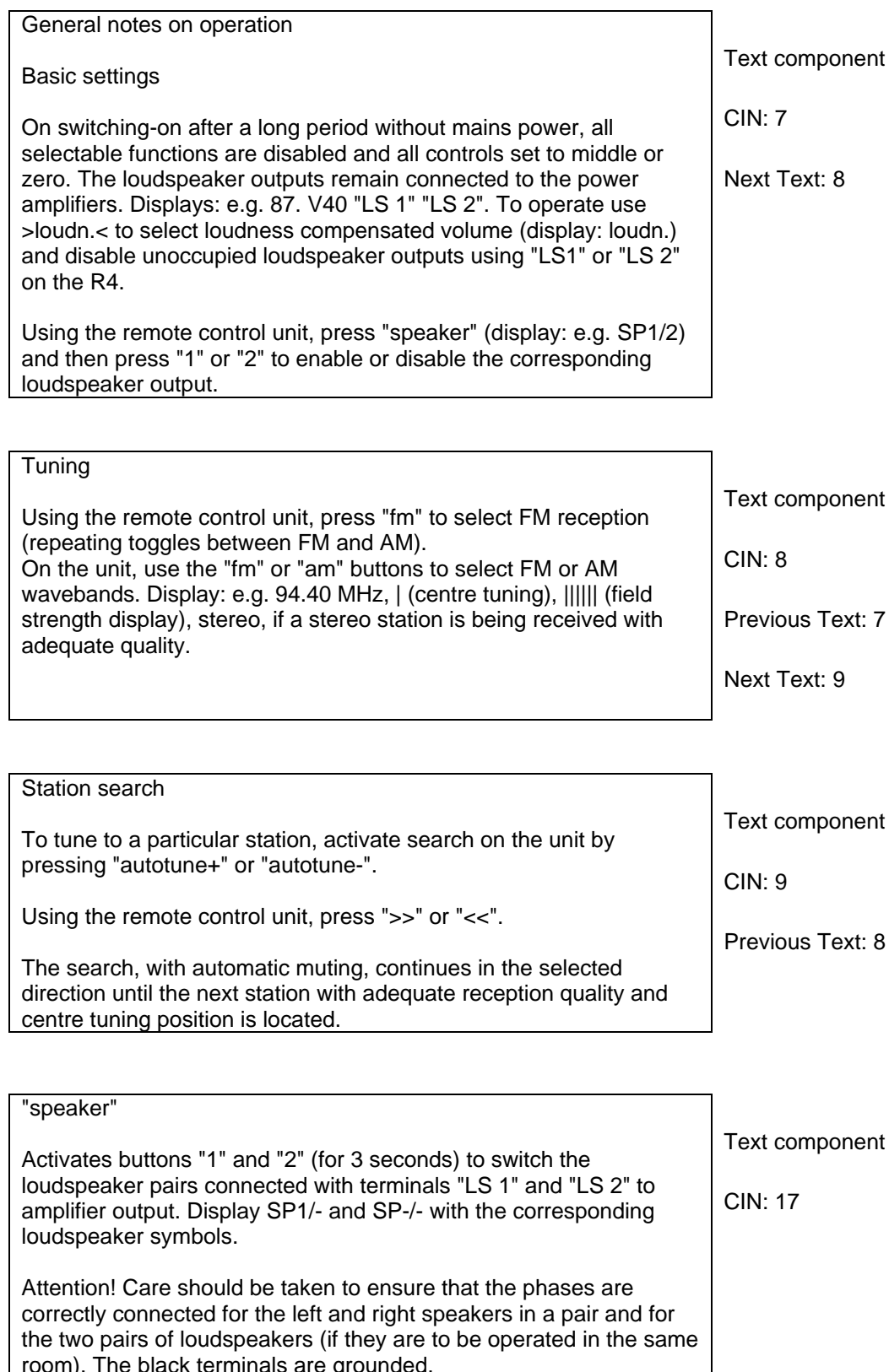


Figure 29: Text component architecture

The local interactivity can be implemented by the correct cooperation of the local actions (LocActVal parameter) which are part of the sensitive text and other interactive elements of the dialogue box. In the example case, the user can skip from the text of figure 27 to figure 28 with the Sensitive Text "Speaker" if the Sensitive Text contains the local action "open component (10,17)".

The content of a text component should always start with an in-text reference to a Font Resource object to set the text attributes for the following text. This will ensure that if the user jumps between non-consecutive text components in a chain, the correct text presentation is always selected. If the text content of the text component does not start with a font selection the following rules apply:

- on scrolling down between two consecutive text blocks the text content of the latter block shall be displayed starting with the text attributes that were last in use (text attributes do not change);
- on jumping between two non consecutive text blocks or on scrolling backwards the text content of the jumped-to block shall be displayed starting with the text attributes that are defined in the "<initial font>" attribute of the given text area component.

NOTE: The information contained in the PreviousText and NextText attributes can be used as an indication how to draw (position of the slider) the local scrolling tools.

Attributes:

- CIN: Component Identification Number;
- PreviousText: CIN of the backward concatenated text component;
- CurrentText: Text definition, including references to font resource objects, text resource objects and sensitive text components;
- NextText: CIN of the forward concatenated text component.

#### **7.6.1.5 The Sensitive Text Component**

This component defines the activation and validation operations for sensitive text strings. These are part of text components.

Visual aspect: see subclause 7.6.1.4.

Attributes:

- CIN: Component Identification Number;
- NotAccessible: The element shall not be accessible;
- LocActAct: Specifies the local actions which are associated with the component and triggered by its activation;
- LocActVal: Specifies the local actions which are associated with the component and triggered by its validation.

#### **7.6.1.6 The Graphic Output Area component**

- description: The Graphic Output Area is a rectangle for the display of graphical data (bitmaps, Videotex data etc.). A bitmap is referenced via a BIN. Four different display modes exist: the bitmap can be centred in the Graphic Output Area, it can be stretched to cover the whole area, it can be tiled or it can be mapped. In any case, the displayed part of the bitmap shall not exceed the space reserved with this component. If the Graphic Output Area references a non existing Resource object the terminal shall request the Resource object from the VEMMI application using the VEMMI\_Object\_Retransmission command with the <request type> attribute set to resource.

If the Graphic Output Area references a Resource object which again references a non existing file the terminal shall request the file from the VEMMI application using the VEMMI\_Object\_Retransmission command with the <request type> attribute set to file. The terminal may allocate on the screen the space needed to display this bitmap. If the bitmap file is undergoing a resource file transfer operation, the terminal may choose to display it at the end of the file transfer or if possible to decode it and display it on the flow during the resource file transfer operation.

Videotex data are referenced via a VIN. In any case the Videotex data shall not exceed the space reserved with this component. The VTX data decoder shall always be in the initial state (40x24

mode, etc.) when starting to decode VTX data content. Within a given VTX data content the 80x24 VTX mode may be invoked. No more than two DRCs shall be defined in a terminal at a given time. If DRCs are redefined there is no guaranteed effect.

Several display modes can be selected for the Graphic Output Area. Selecting different display modes will result in different visual representations of the same bitmap. The following display modes are defined:

- a) centred: the bitmap is centred in the Graphic Output Area;
- b) stretched: the bitmap is stretched to cover the entire Graphic Output Area (default);
- c) tiled: the bitmap is repeated horizontally and vertically to cover the entire Graphic Output Area;
- d) mapped: the dimensions of the bitmap are greater than the dimensions of the Graphic Output Area. Scrolling tools are offered to virtually move the bitmap in the Graphic Output Area to be able to view it entirely. The Graphic Output Area acts like a window through which a part of the bitmap can be seen. In this mode an attribute can be set specifying the initial position of the bitmap within the Graphic Output Area before any user scrolling. The attribute gives the point relative to the size of the bitmap that should be in the centre of the Graphic Output Area in NDC.

The space allocated for the scrolling tools shall be included in the dimensions of the component.

If a terminal is not able to apply any of the advanced bitmap display modes it may display the bitmaps in their original size and clip the bitmap to the overall dimensions of the Graphic Output Area. The initial positioning in the mapped mode may also be ignored.

- behaviour: It is inaccessible;
- interactive functionality:

- a) scrolling if display type is set to "mapped".

Visual aspect (see figure 30):

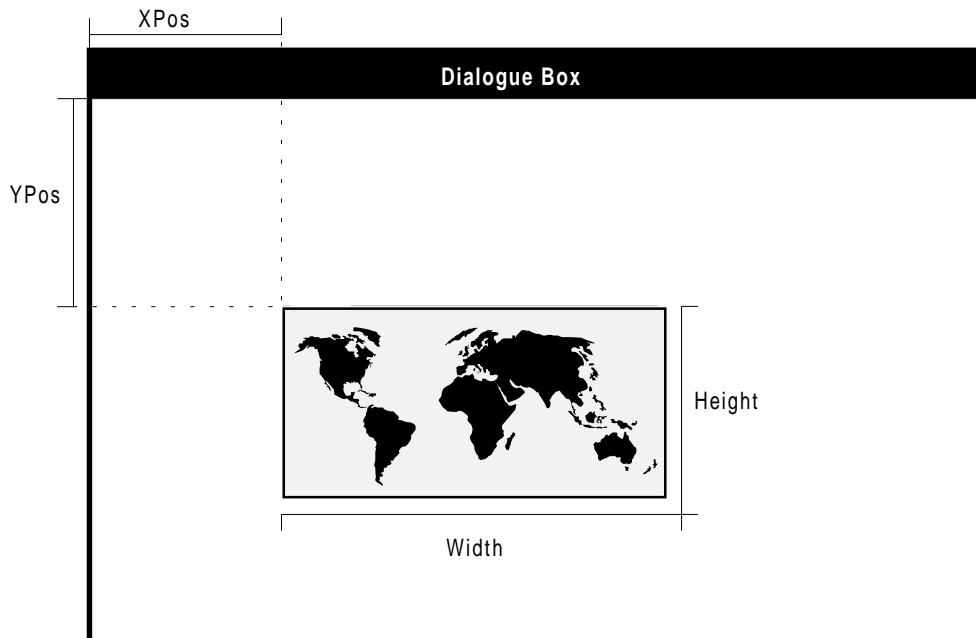


Figure 30: Frame and a bitmap



Attributes:

- CIN: Component Identification Number;
- XPos: Horizontal position of the top left corner of the bitmap rectangle;
- YPos: Vertical position of the top left corner of the bitmap rectangle;
- Width: Width of the bitmap rectangle;
- Height: Height of the bitmap rectangle;
- Closed: The element shall be in the closed state;
- DispType: Specifies the display mode of the bitmap: centred, stretched (default), tiled or mapped;
- BIN: Bitmap Identification Number;
- VIN: Videotex Identification Number.

#### 7.6.1.7 The Push Button component

Description, Behaviour, Interactive functionality: See subclause 7.4.1.1.

Visual aspect (see figure 31):

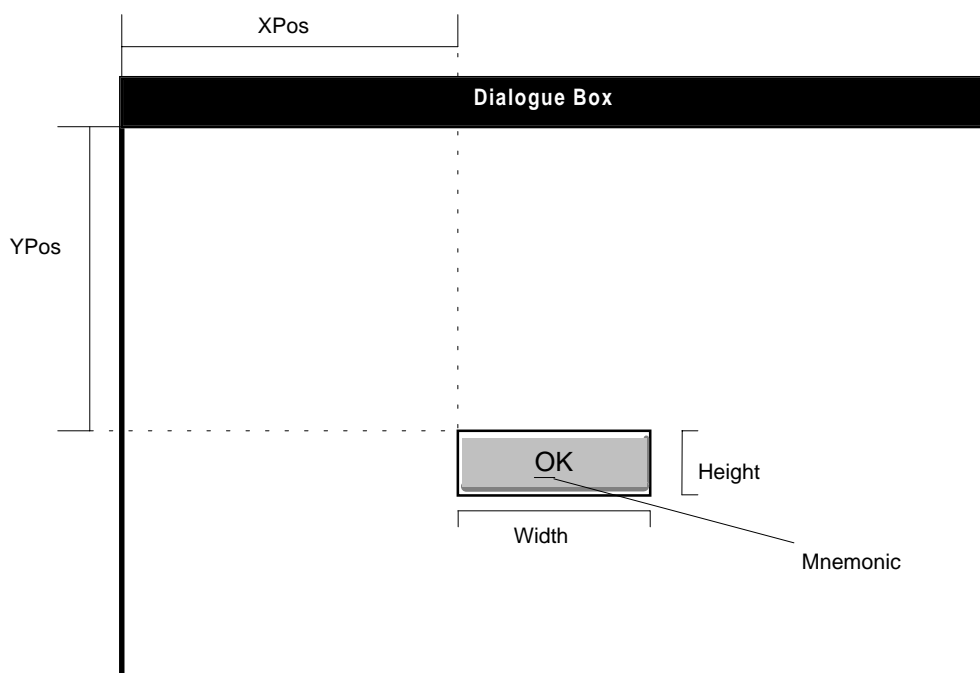


Figure 31: Push Button

Attributes:

- CIN: This attribute carries the Component Identification Number;
- XPos: This attribute carries the horizontal position of the element in NDC;
- YPos: This attribute carries the vertical position of the element in NDC;
- Width: This attribute carries the width of the component;
- Height: This attribute carries the height of the component;
- Closed: The element shall be in the closed state;
- NotAccessible: The element shall not be accessible;
- BIN: The BIN of the component;
- LabelFont: FIN of the label;
- Text: The label of the component;
- LocActAct: This attribute carries the code for the local action which is associated to the component and triggered by its activation;
- LocActVal: This attribute carries the code for the local action which is associated to the component and triggered by its validation.

#### 7.6.1.8 The Text Input Field component

Description: The Text Input Field is composed of two items:

- the text label;
- the input area.

An attribute font can be associated to the text label. The input area has its starting location immediately after the text label. The type of the input data can be specified by the application. The following predefined types are available:

- any text character;
- alphabetic (A to Z, a to z, diacritical characters);
- numeric (0 to 9, +, -, comma, dot, space);
- alphanumeric (alphabetic, numeric).

An attribute of the Text Input Field specifies whether the user inputs are echoed or not. The VEMMI application can define one character to echo all user inputs.

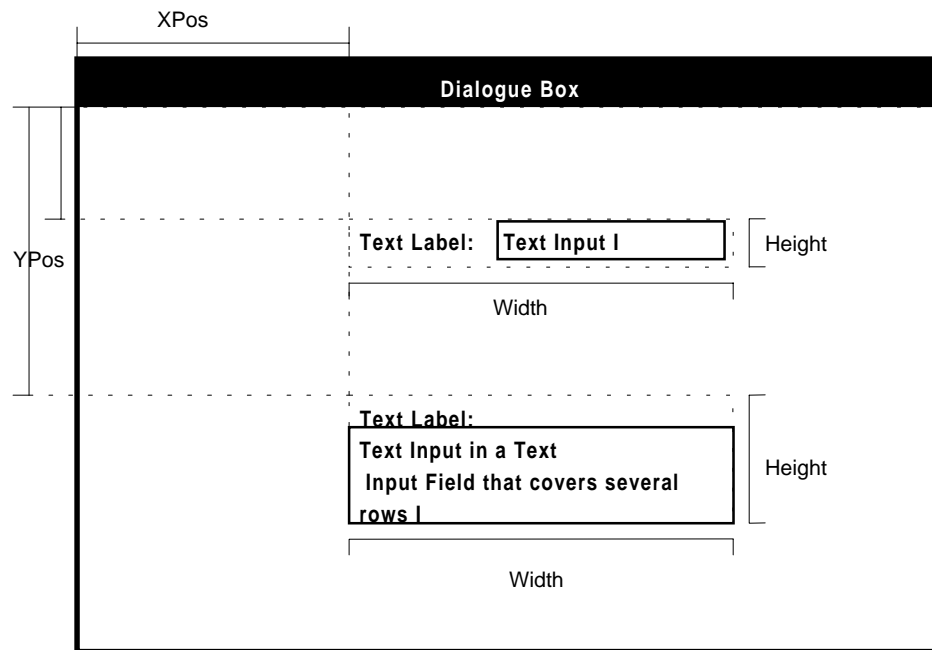
A mnemonic key can be used to activate the Text Input Field (see subclause 7.1.7).

- behaviour: If the space allocated for the input area in a one line Text Input Field is not sufficient to display all characters entered by the user, the input area shall offer horizontal scrolling facilities. If the space, allocated for the input area in a Text Input Field with more than one line, is not sufficient to display all characters entered by the user, the input area shall offer vertical scrolling facilities. The space needed to present scrolling facilities shall be included in the overall dimensions of the component. In order to display all possible characters, it is recommended to give enough space for user inputs. If the maximum number of the input characters is reached this should be indicated to the user;
- constraints: The maximum length of the text label is limited to the first line of the Text Input Field component width. For one line input fields the label is placed in front of the input area and left aligned, while the input area is aligned to the right side. For multiline Text Input Fields it is placed in the top row and shall be left aligned. There is no space reserved for the text label if the attribute TextLabel is not present.

In case of several one-line input fields that should be presented left aligned on a terminal, text labels should not be used. The Text Presentation Area can then be used to label the input field.

- Interactive functionality:
  - a) activation;
  - b) scrolling;
  - c) local editing functionality of the input area (terminal dependent).

Visual aspect (see figure 32):



**Figure 32: Text Input Field**

**Attributes:**

- CIN: This attribute carries the Component Identification Number;
- XPos: This attribute carries the horizontal position of the element in NDC;
- YPos: This attribute carries the vertical position of the element in NDC;
- Width: This attribute carries the width of the component;
- Height: This attribute carries the height of the component;
- Closed: The element shall be in the closed state;
- NotAccessible: The element shall not be accessible;
- MaximTxt: Local maximization can maximize the text of the component;
- DefValue: This attribute carries the default value for the component;
- TextLabel: This attribute carries the text label of the component;
- LabelFont: Specifies the FIN for the text label;
- InputType: This attribute specifies which type of user inputs shall be accepted by the component;
- EchoType: This attribute specifies which type of echo shall be made on user inputs;
- EchoChar: This attribute carries the character that shall be displayed to echo user inputs if the EchoType is set to "echo defined character";
- MaxChar: This attribute carries the maximum number of characters the user can enter in one line;
- MaxLine: This attribute carries the maximum number of lines the user can enter in a multiline input field;
- Multiline: Indicates a multiline input field;
- InputTransformation: The attribute specifies if the terminal shall convert the characters entered by the user to upper case characters, to lower case characters or if no conversion shall be done;
- LocActAct: This attribute carries the code for the local action which is associated to the component and triggered by its activation;
- LocActVal: This attribute carries the code for the local action which is associated to the component and triggered by its validation.

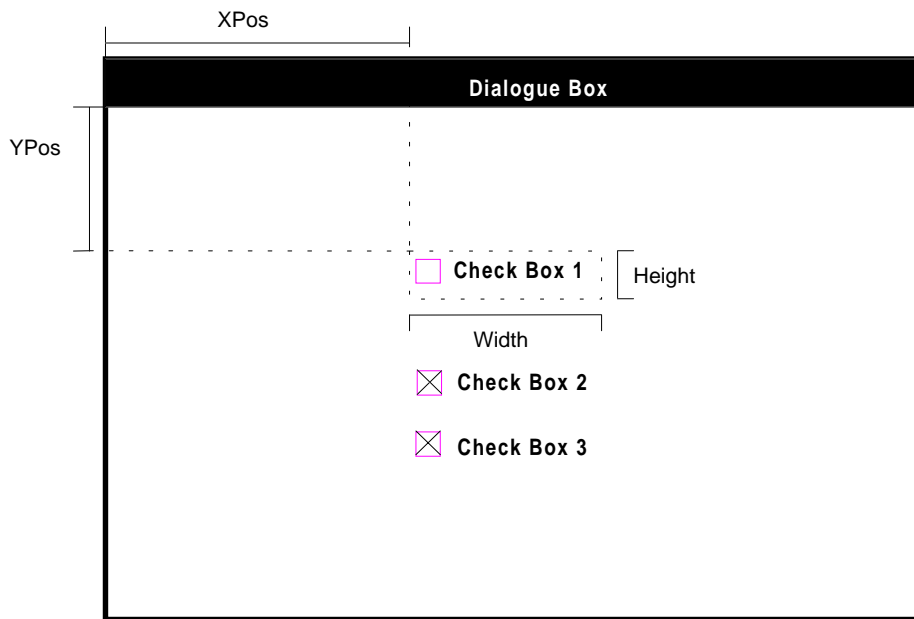
**7.6.1.9 The Check Box component**

- description: The Check Box component is composed of two items:
  - a) the text label item;
  - b) the check item.

A mnemonic key can be used to activate the component (see subclause 7.1.7). The check item shall support two different choices, visually identifiable, representing the two possible values marked or unmarked. The space allocated for the component shall include, in its dimensions, the space needed to represent this check item.

- behaviour: A Check Box acts like a switch. When switched by the user, its value changes from unmarked to marked or from marked to unmarked. The value of Check Boxes is independent of the value of any other components.
- interactive functionality:
  - a) activation;
  - b) switch between the values marked and unmarked;
  - c) validation.

Visual aspect (see figure 33):



**Figure 33: Check Box**

Attributes:

- CIN: This attribute carries the Component Identification Number;
- XPos: This attribute carries the horizontal position of the element in NDC;
- YPos: This attribute carries the vertical position of the element in NDC;
- Width: This attribute carries the width of the component;
- Height: This attribute carries the height of the component;
- Closed: The element shall be in the closed state;
- NotAccessible: The element shall not be accessible;
- TextLabel: This attribute carries the text label of the component;
- LabelFont: FIN of the label;
- DefMarked: The default value of the element shall be marked;
- LocActAct: This attribute carries the code for the local action which is associated with the component and triggered by its activation;
- LocActVal: This attribute carries the code for the local action which is associated with the component and triggered by its validation.

#### **7.6.1.10 The Radio Button component**

- description: The Radio Button component is composed of two items:
  - a) the text label item;
  - b) the check item.

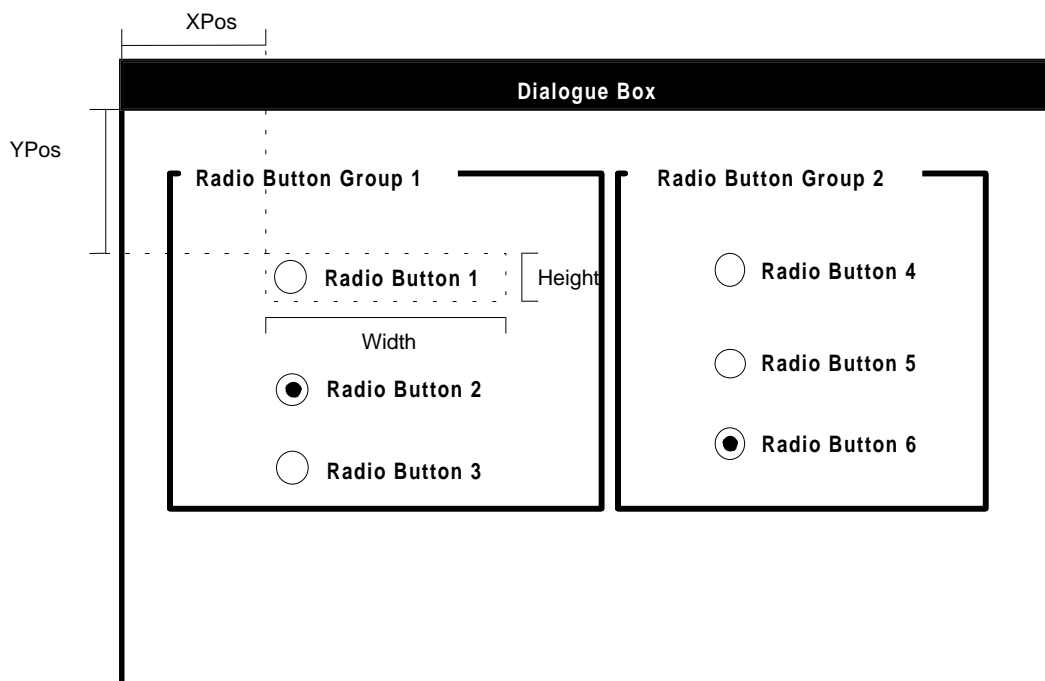
A mnemonic key can be used to activate the component (see subclause 7.1.7).

The check item shall support two different choices, visually identifiable, representing the two possible values, marked and unmarked. The space allocated for the component shall include, in its dimensions, the space needed to represent this check item.

Radio Buttons are typically used in groups to provide a single-choice field.

- behaviour: No more than one Radio Button shall be marked in a Radio Button group at a given time. The attempt to mark a second Radio Buttons shall cause the unmarking of the previously marked Radio Button, whatever its state (opened, closed, accessible, inaccessible) within the application may be.
- interactive functionality:
  - a) activation;
  - b) switch between the values marked and unmarked;
  - c) validation.

Visual aspect (see figure 34):



**Figure 34: Radio Buttons with a Frame and a Text Presentation Area**

Attributes:

- CIN: This attribute carries the Component Identification Number;
- XPos: This attribute carries the horizontal position of the element in NDC;
- YPos: This attribute carries the vertical position of the element in NDC;
- Width: This attribute carries the width of the component;
- Height: This attribute carries the height of the component;
- Closed: The element shall be in the closed state;
- NotAccessible: The element shall not be accessible;
- TextLabel: This attribute carries the text label of the component;
- LabelFont: FIN of the label;
- Group: This attribute carries the identifier of the Radio Button group. Only one Radio Button in a group can be marked at one specific time;
- DefMarked: The default value of the element shall be marked;
- LocActAct: This attribute carries the code for the local action which is associated with the component and triggered by its activation;
- LocActVal: This attribute carries the code for the local action which is associated with the component and triggered by its validation.

#### 7.6.1.11 The List Box component

- description: The List Box component is a rectangular area in which a list of text items is offered to the user for choice. If the total number of items exceeds the number of presented items, scrolling facilities shall be offered to the user through dedicated scrolling controls. In this case the space allocated for the component shall include in its dimensions the space needed to represent the scrolling facilities. The tools to provide the scrolling functionality are terminal dependent (vertical scrollbar, buttons, etc.). When the entire list is visible, the scrolling controls may be invisible. A list can have several columns. Consequently list items can consist of several cells.

Small icons may be added to the cells in order to visually emphasize their meaning. The referenced bitmap can be resized to fit to the local List Box representation.

The whole list is composed of list items. If the "sorted" attribute is selected, the list items should be sorted in their alphabetical order (according to the first cell). If "sorted" is not selected the list shall be presented in the ascending order of the list item indices. However, the correspondence between the list items and the indices is always maintained. For each of the columns of a List Box an alignment parameter can be specified;

- behaviour: When the List Box is active the terminal shall offer shifting facilities to the user. A List Box contains a list of items that the user can scroll through and select from. Single or multiple choice can be made by the user, depending on the list attributes.

There may be local actions associated with each List Box item. Additionally, the List Box may also have local actions associated. Upon activation and validation of a List Box item, the local actions associated with the List Box item shall be executed. If there is no local action associated with the selected item, the local action associated with the List Box itself shall be executed. In this case, any OIN appearing in element specific commands of the List Box component is incremented by the value of the list index of the List Box item.

A List Box with multiple choice can not be validated;

- interactive functionality:
  - a) activation;
  - b) single or multiple choice;
  - c) validation (single choice);
  - d) scrolling;
  - e) shifting.

Visual aspect (see figure 35):

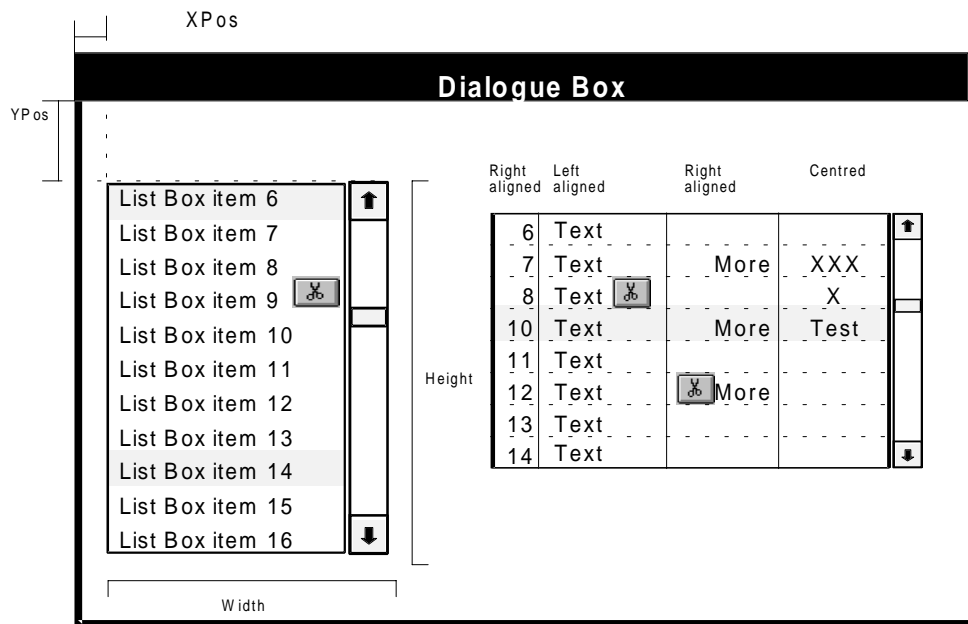


Figure 35: List Box

Attributes:

- CIN: This attribute carries the Component Identification Number;
- XPos: This attribute carries the horizontal position of the element in NDC;
- YPos: This attribute carries the vertical position of the element in NDC;
- Width: This attribute carries the width of the component;
- Height: This attribute carries the height of the component;
- Closed: The element shall be in the closed state;
- NotAccessible: The element shall not be accessible;
- DefItem: This attribute carries the index of the list item that is selected by default when the component is opened for the first time;
- Sorted: The list items are presented sorted in alphabetical order;
- MultipleChoice: The element allows multiple choices;
- Arrayed: The component shall consist of several columns;
- ColumnWidth: Width of the columns;
- Alignment: Text alignment in the columns (left, right, centred);
- ReportIndex: The report command shall report the index of the List item selected not the item itself. The index report can lead to ambiguous interpretation of reported data when the server is simultaneously updating the content of list items;
- LocActAct: This attribute carries the code for the local action which is associated with the component and triggered by its activation;
- LocActVal: This attribute carries the code for the local action which is associated with the component and triggered by its validation.

List item attributes:

- ListIndex: This attribute carries the index of the list item in the component description. It allows in a modify command the modification of a specific list item. The list indices shall be defined in an ascending order. The list indices may not be consecutive in order to allow easy updating of the list (inserting of list items). The terminal shall present a consecutive list to the user, sorted by the ascending value of the list indices (if Sorted is not set) corresponding to the list text. If in modify command a list index for a list item is used which is already existing than the existing list item is replaced. If in a modify command a list index of a list item is used that is not existing in the terminal, the new list item shall be added. If in a modify command a list index is referenced and the "delete list item" attribute is set the terminal shall delete the list item;
- Text: This attribute carries the text content of a list cell;
- IconReference: The BIN of the Bitmap Resource Object representing the icon;

- ItemLocAct: This attribute carries the code for the local action which is associated to the item and triggered by its activation;
- ItemLocVal: This attribute carries the code for the local action which is associated to the item and triggered by its validation.

#### 7.6.1.12 The Combination Box component

- description: The Combination Box is composed of two parts:
  - a) a single choice List Box;
  - b) a one line text field located at the top of the list.

If the total number of items exceeds the number of presentable items, scrolling shall be offered to the user through dedicated scrolling facilities. In this case the space allocated for the component shall include in its dimensions the space needed to represent the scrolling facilities. The tools to provide the scrolling functionality are terminal dependent (vertical scrollbar, buttons, etc.).

Small icons may be added to the list items in order to visually emphasize their meaning. The referenced bitmap can be resized to fit to the local Combination Box representation.

The whole list is a component composed of list items. The CR+LF characters shall not be present in a list item of the Combination Box. If the "sorted" attribute is selected, the list items should be sorted in their alphabetical order. If "sorted" is not selected, the list shall be presented in the ascending order of the list item indices. However, the correspondence between the list items and the indices is always maintained.

A variation of the Combination Box is a drop down Combination Box. It is composed of a Combination Box and a Push Button. Only the text field and the Push Button are displayed until the user validates the associated Push Button. The validation of the Push Button causes the display of the associated List Box;

- behaviour: When the Combination Box is active the terminal shall offer shifting facilities to the user. A Combination Box contains a list of items that the user can scroll through and select from to complete the text field. A parameter of the Combination Box specifies if the text field is editable or not. If the text field is editable, the user can type text directly into the text field. A parameter of the component specifies if the text entered by the user shall match one of the items contained in the list.

There may be local actions associated with each List Box item. Additionally, the Combination Box may also have local actions associated. Upon activation and validation of a List Box item, the local actions associated with the List Box item shall be executed. If there is no local action associated with the selected item, the local action associated with the Combination Box itself shall be executed. In this case, any OIN appearing in element specific commands of the List Box component is incremented by the value of the list index of the listbox item (no increment in case of direct input in the input field).

If the Combination Box is designed to be a drop down Combination Box, the validation of the Push Button shall switch between displaying and hiding the List Box;

- interactive functionality:
  - a) activation;
  - b) single selection;
  - c) validation;
  - d) scrolling;
  - e) editing.

Visual aspect (see figure 36):



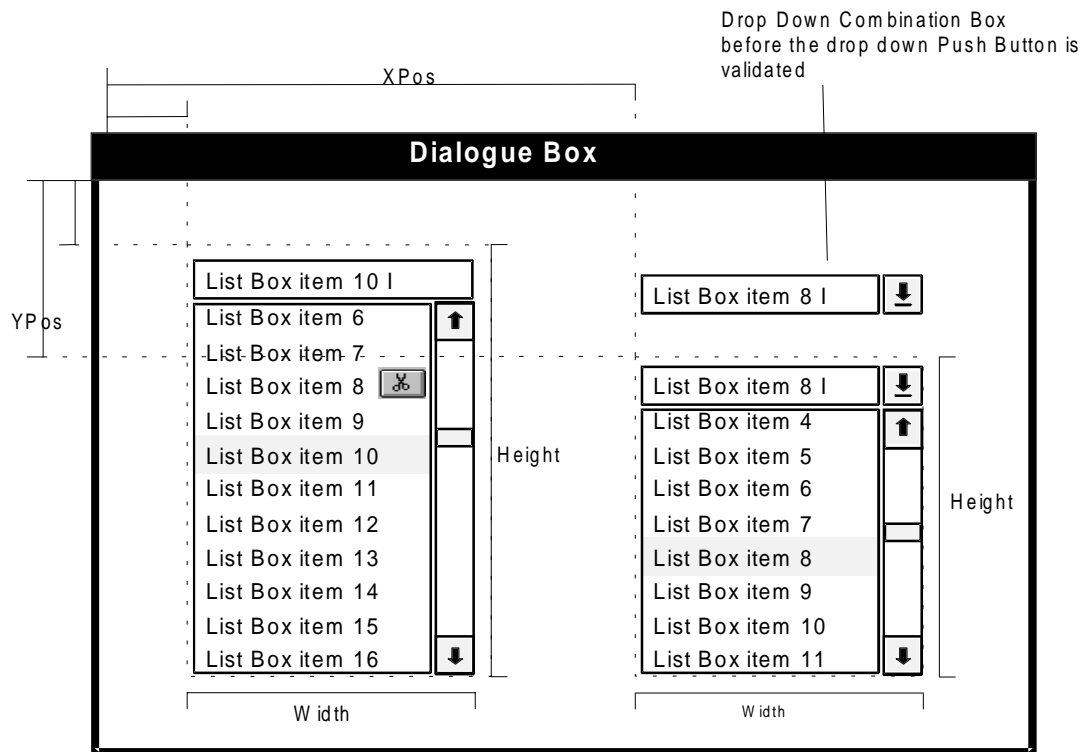


Figure 36: Combination Box

Attributes:

- CIN: This attribute carries the Component Identification Number;
- XPos: This attribute carries the horizontal position of the element in NDC;
- YPos: This attribute carries the vertical position of the element in NDC;
- Width: This attribute carries the width of the component;
- Height: This attribute carries the height of the component;
- Closed: The element shall be in the closed state;
- NotAccessible: The element shall not be accessible;
- DefItem: This attribute carries the index of the list item that is selected by default when the component is opened for the first time;
- NoEdit: The text field of the component shall not be editable;
- NoConsistency: This attribute is applied if "NoEdit" is not set. It indicates that the text entered by the user can be any text string, not necessarily one of the defined text items;
- MaxChar: This attribute carries the maximum number of characters the user can enter in the component. It shall only be present if the "EditableInput" Parameter is set true;
- NoDropDown: The element shall not be drop down;
- NotSorted: The list items are presented sorted in alphabetical order;
- LocActAct: This attribute carries the code for the local action which is associated to the component and triggered by its activation;
- LocActVal: This attribute carries the code for the local action which is associated to the component and triggered by its validation.

List item attributes:

- ListIndex: This attribute carries the index of the list item in the component description. It allows in a modify command the modification of a specific list item. The list indices shall be defined in an ascending order. The list indices may not be consecutive in order to allow easy updating of the list (inserting of list items). The terminal shall present a consecutive list to the user, sorted by the ascending value of the list indices (if Sorted is not set) corresponding to the list text. If in modify command a list index for a list item is used which is already existing, then the existing list item is replaced. If in a modify command a list index of a list item is used that is not existing in the terminal, the new list item shall be added. If in a modify command a list index is referenced and the "delete list item" attribute is set the terminal shall delete the list item;
- Text: This attribute carries the text content of a list item;
- IconReference: The BIN of the Bitmap Resource Object representing the icon;

- ItemLocAct: This attribute carries the code for the local action which is associated to the item and triggered by its activation;
- ItemLocVal: This attribute carries the code for the local action which is associated to the item and triggered by its validation.

#### 7.6.1.13 The Slider Component

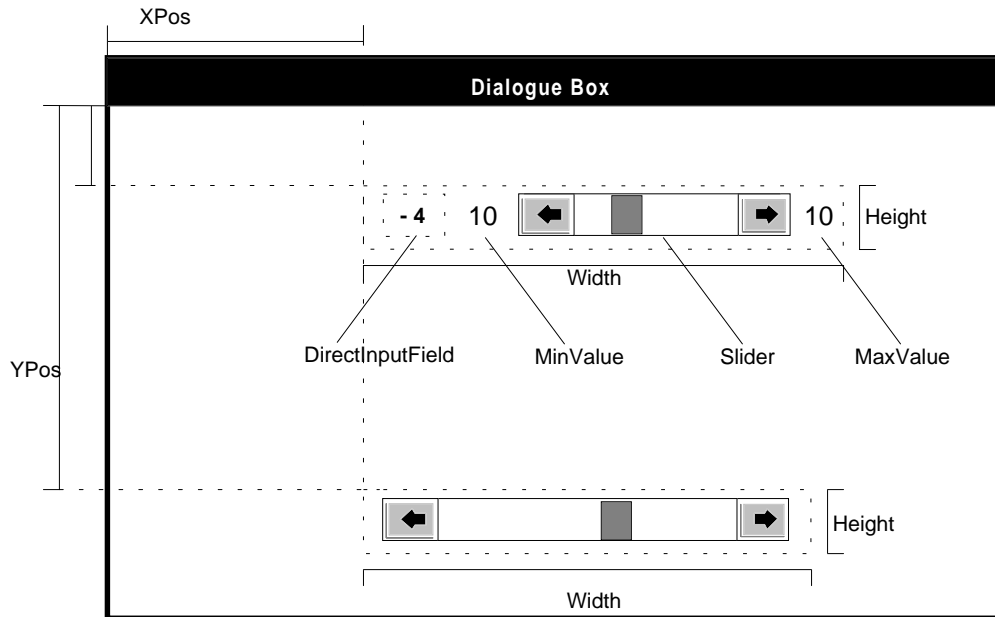
- description: The slider offers the selection of an analogue value by moving an adjustable marker on a slide bar between a minimum and a maximum value. The intervals are set by the application. The Slider component consists of the following items:
  - a) Slider (with the movable marker, the initial marker position shall correspond to the default value, set by application);
  - b) DirectInputField (optional), an input and output field which contains:
    - 1) the initial value when the Slider is activated;
    - 2) the values in text form corresponding to the slider movements;
    - 3) an Input field for user inputs of the slider position.

The Slider Component may be oriented in vertical and horizontal direction. All items shall fit in the rectangle determined by the component dimensions. The DirectInputField should be located at the left side of a horizontal slider and on top of a vertical slider.

A local action which is triggered on the validation shall be performed when the user deactivates the component;

- behaviour: When the Slider Component is active the marker can be moved by striking the cursor keys or by mouse clicks. If the optional DirectInputField is present, the user should be able to put in values directly by key strokes. The changed values shall be displayed immediately in the DirectInputField. The marker shall be moved onto a position which is relative to this value input. In the other case, when the marker is moved, the displayed value shall refer to this movement. If the MinValue and MaxValue attributes are present, an indication about the valid input range could be displayed to the user;
- interactive functionality:
  - a) local input functionality of the DirectInputField (application dependent);
  - b) local positioning of the slider marker;
  - c) validation by deactivation of the component;
  - d) activation.

Visual aspect:



**Figure 37: Slider Component**

Attributes:

- CIN: This attribute carries the Component Identification Number;
- XPos: This attribute carries the horizontal position of the element in NDC;
- YPos: This attribute carries the vertical position of the element in NDC;
- Width: This attribute carries the width of the component;
- Height: This attribute carries the height of the component;
- Closed: The element shall be in the closed state;
- NotAccessible: The element shall not be accessible;
- MinValue: The lowest adjustable value;
- MaxValue: The highest adjustable value;
- Increment: The interval size between two adjustable values;
- InitialValue: The initial value, when the Slider Component is opened;
- Negative: This attribute can be associated with MinValue, MaxValue, InitialValue and also indicates that the values are negative;
- DirectIn: This attribute indicates that MinValue and MaxValue could be displayed, also the user's direct input should be allowed;
- Vertical: The element shall be presented vertical;
- LocActAct: Specifies the code for the local action which is associated with the component and triggered by its activation;
- LocActVal: Specifies the code for the local action which is associated with the component and triggered by its validation.

#### 7.6.1.14 The Sensitive Area Component

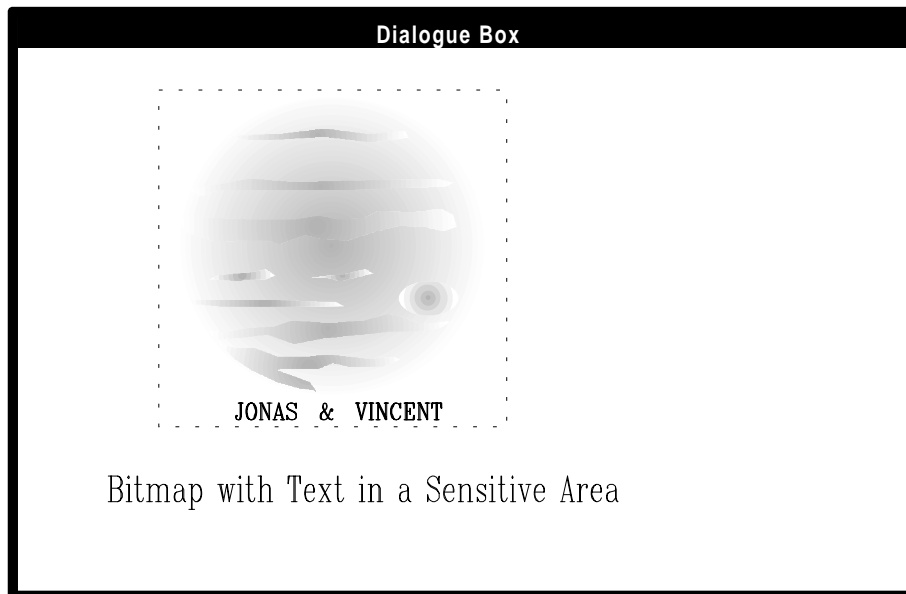
- description: A Sensitive Area component is an area, non materialized, in which the application permits a validation operation. The Sensitive Area component is intended to be used associated with a Graphic Output Area component of the Dialogue Box. For the use with text components, the in-text attributes for the definition of sensitive text are more appropriate. The sensitive area is a rectangle inside the dialogue box or it can be locally associated to the content of a graphic display area. In the latter case all positioning and all reports are done relative to the content of the area and the Sensitive area shall be scrolled with the bitmap content.

If the "locator" attribute is set, the terminal shall send upon validation not only it's CIN to the VEMMI application, but also the coordinates of the cursor in NDC at the time of the validation with respect to the component origin;

- behaviour: When active a visible effect (e.g. thread, dot lines) can be implemented;

- constraints: A Sensitive Area shall not be covered by other components or items which are sensitive to user interaction, in order to avoid conflicts;
- Interactive functionality:
  - a) activation;
  - b) validation.

Visual aspect:



**Figure 38: Bitmap, Text and Sensitive Area**

Attributes:

- CIN: Specifies the Component Identification Number;
- XPos: Specifies the horizontal position sensitive area;
- YPos: Specifies the vertical position of the sensitive area;
- Width: Specifies the width of the sensitive area;
- Height: Specifies the height of the sensitive area;
- Closed: The element shall be in the closed state;
- NotAccessible: The element shall not be accessible;
- Locator: The position of the cursor in NDC at the time of the validation with respect to the component origin shall be sent with each report command;
- LocActAct: Specifies the local actions which are associated with the component and triggered by its activation;
- LocActVal: Specifies the local actions which are associated with the component and triggered by its validation. If the locator attribute is set, each report action shall also contain the coordinates of the cursor in NDC at the time of the validation with respect to the component origin;
- Content Associated: The component is locally associated with a Graphic Display Area;
- AssGraphicReference: CIN reference to the associated graphic area. This attribute is only applicable in content associated mode.

#### **7.6.1.15 The Multimedia Area component**

- description: The Multimedia Area is an area intended to present multimedia information to the user. Multimedia information consists of several kinds of monomedia (text, graphics, bitmaps, audio, video) that have defined relations (spatial and temporal) and that can be scrolled together as a whole.

For the time being only HTML as a content coding format for the Multimedia Area is identified;

- scrolling, Border: If the space needed for the presentation of the data is bigger than the space allocated for the component, vertical and/or horizontal scrolling tools shall be provided by the terminal application. The terminal may present:
  - a) scrolling tools next to the right side of the area for vertical scrolling and scrolling tools at the bottom for horizontal scrolling;
  - b) reduced scrolling tools as buttons overlapping the area;
  - c) keys that provide the scrolling functionality.

The space needed for the scrolling tools is included in the overall dimensions of the component.

The component can be provided with a border area. When present, the border shall be included in the overall dimensions of the component. When a border is requested a frame shall be drawn by the terminal. The frame drawing is terminal dependent. It should not be wider than 1/160 NDC. If scrolling tools are provided next to the right side of the component the border shall be extended to include them visually in the box;

- input control: There are two different strategies for a Multimedia Area how to handle user inputs and how two be updated. The two alternatives can be selected in the encoding of the component.

The first alternative is that user inputs are not controlled by the VEMMI local manager, but treated as specified by the content encoding format.

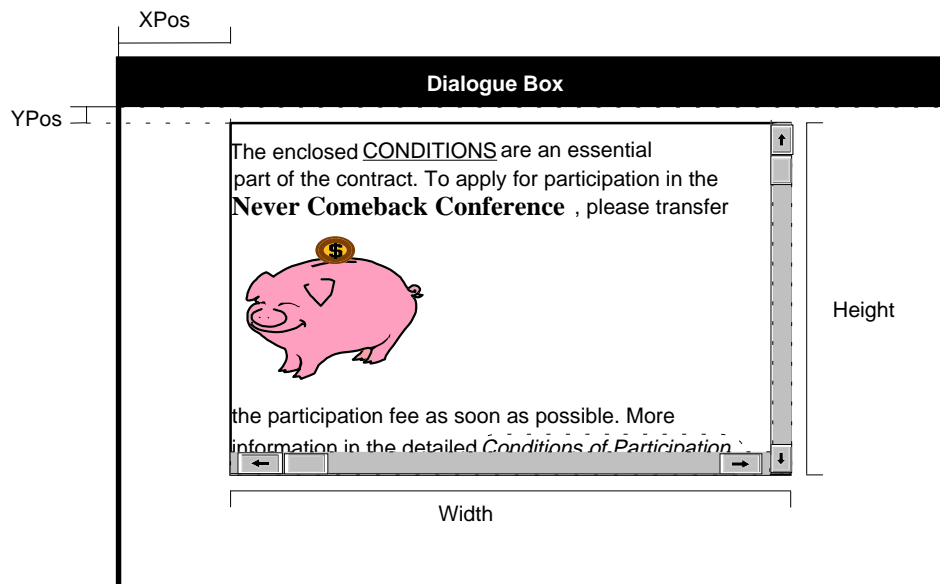
EXAMPLE 1:                   A click on a HTML link induces a HTTP (or other appropriate Internet protocol) request on the underlying protocol stack. Data sent in response to such a request are presented directly in the Multimedia Area (not using the VEMMI\_Modify\_Component). This procedure is appropriate for a VEMMI implementation on top of the TCP/IP protocol stack.

The second alternative is that user inputs in the Multimedia Area are controlled by the VEMMI local manager.

EXAMPLE 2:                   The click on a HTML link induces a VEMMI\_User\_Data primitive carrying the URL (similar to an HTTP request only included in a VEMMI primitive send to the VEMMI host across the VEMMI protocol stack). The VEMMI host may then use the information to retrieve the appropriate document and send the data to the terminal using VEMMI\_Modify\_Component. This procedure is appropriate for a VEMMI implementation on top of a non-TCP/IP protocol stack.

- interactive functionality:
  - a) scrolling;
  - b) functionality offered by the data content (HTML).

Visual aspect (see figure 39):



**Figure 39: Multimedia Area**

Attribute:

- CIN: This attribute carries the Component Identification Number;
- XPos: This attribute carries the horizontal position of the element in NDC;
- YPos: This attribute carries the vertical position of the element in NDC;
- Width: This attribute carries the width of the component;
- Height: This attribute carries the height of the component;
- Closed: The element shall be in the closed state;
- NotAccessible: The element shall not be accessible;
- Maximizable: The terminal shall offer the local maximisation option as described in subclause 7.1.5 (strategy 2 should be implemented);
- NoBorder: The element shall not have a border;
- MultimContentType: The datatype of the content of the component. For the time being only HTML is identified;
- MultimediaContent: Direct definition of a Multimedia content or reference to a Multimedia resource object;
- DirectNavigation: The terminal should offer the possibility for the user to enter directly navigation information related to the Multimedia Area. (direct URL-Input in case of HTML content). The input field for the direct navigation information is terminal dependent;
- NonControlledMode: The VEMMI manager does not control the data handling (user input, content updates.) in the Multimedia Area.

## 7.7 The Message Box

- description: The Message Box is a rectangular area in the DDA which contains text information.

Four specified types of message are defined:

- a) general message;
- b) information message;
- c) warning message;
- d) action message.

This information can be used to add an icon characterizing the type of the message to the presentation of the message box.

An attribute of the Message Box specifies if the terminal should perform a sound at the time the Message Box is opened.

The Message Box can be provided with a border area which should be equal to 1/160 NDC. When a border is requested, a frame shall be drawn by the terminal. This frame drawing is terminal dependent;

- behaviour: An attribute of the Message Box specifies the destroy event of a Message Box. The following five values are available:
  - a) destroy by any user interaction;
  - b) close by any user interaction;
  - c) destroy by the user validation of an implicitly defined button;
  - d) close by the user validation of an implicitly defined button;
  - e) no implicit destroy event is defined.

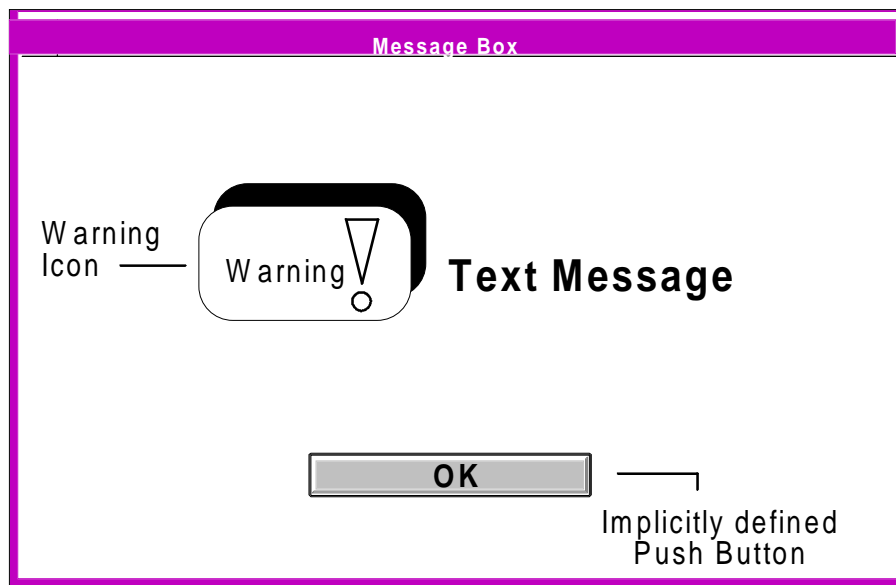
The destroying or closing of the Message Box shall not induce any report to the server.

If no position is defined by the VEMMI application the terminal shall centre the Message Box in the DDA.

If no dimensions are defined by the VEMMI application the terminal shall calculate the appropriate size of the Message Box, in order to have enough room to display the text message and the possible Push Button. The size of the possible Push Button is terminal dependent. The label of the Push Button should have a semantics similar to "OK";

- interactive functionality:
  - a) activation;
  - b) validation.

Visual aspect (see figure 40):



**Figure 40: Message Box with a warning message**

Attributes:

- XPos: Specifies the horizontal position of the message box;
- YPos: Specifies the vertical position of the message box;
- Width: Specifies the width of the message box;
- Height: Specifies the height of the message box;
- Closed: The element shall be in the closed state;
- MessageType: One of: general, information, warning, action;
- Modal: The element shall be modal;
- Title: This attribute carries the title of the object. The title shall be displayed in the first row of the object;

- NoBorder: The element shall not have a border;
- AttributedText: Text of the message with in-text attributes;
- MaxTime: Maximum time the object stays on the screen. If this time elapses the object is destroyed regardless the specified destroy event;
- Destroy event: The events which lead to the destruction of the message box;
- NoSound: The terminal should not perform a sound when the message box is opened.

## **7.8 Operative object**

With this object an application references a program which will be linked to the VEMMI application. This object type provides a method to extend the capabilities of an application during runtime. Via the service VEMMI\_Create\_Object the host application defines the operative object. The program is started with VEMMI\_Open\_Object.

The program itself has been downloaded via file transfer as part of the operating system or its presence has been negotiated in another way. It is referenced by its filename. It is up to the host to ensure that the called program is started without error, by providing the correct program parameters. An often used method to extend the functions of programs during runtime is the use of Dynamic Link Libraries (DLL)s.

The program referenced with an operative object can be of the following types:

- standalone program;
- program with a filter interface.

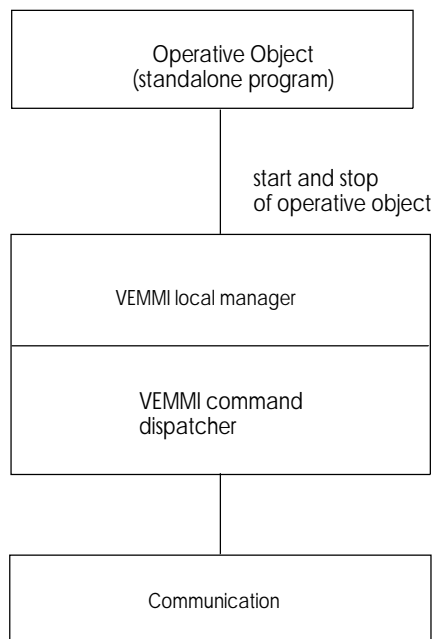
For the purpose of operative objects, the services VEMMI\_Create\_Object, VEMMI\_Open\_Object and VEMMI\_Destroy\_Object are used. The program name and the program parameters are defined as attributes of the operative object.

Two parameters, program description and the source identification, can be used by the terminal to evaluate information to be presented to the user or only to be recorded. This precaution is necessary in order to help the user to detect programs which have been linked to the VEMMI application with intent of sabotage. Their effect on the terminal may not be harmless. The terminal can answer the host "The user refused the start of the program!". The basis on which such an answer is generated is application dependent, but at least the information is available on a structured basis. If the terminal cannot find the program the host needs to be informed with a specific error message.

### **7.8.1 Requirements for standalone programs**

The program has no data exchange channel with the VEMMI local manager during the execution. Program parameters can be defined by the host application and supplied to the program when started. The operative object has no visual effect on the terminal and cannot be activated directly by the user, but the started program may use the local UI on an application specific basis.





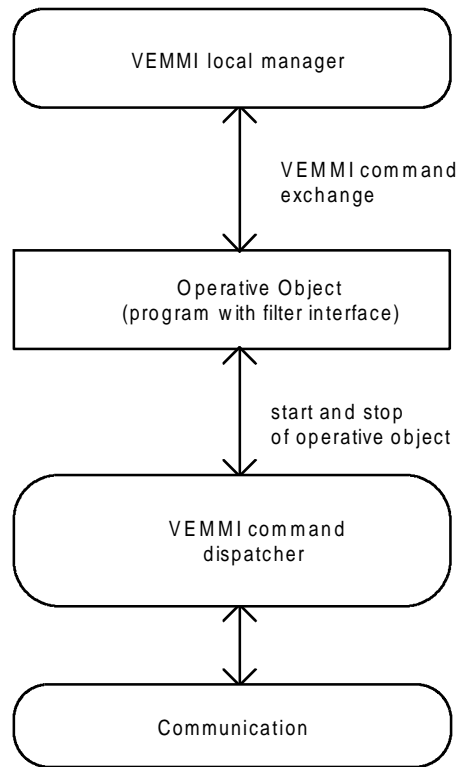
**Figure 41: Standalone programs**

The VEMMI command dispatcher is responsible for delivering the VEMMI commands between the VEMMI local manager and the communication part. All these interfaces are terminal and implementation dependent. They are used in this description to explain the integration of the two types of operative objects in the VEMMI terminal architecture.

#### **7.8.2 Requirements for programs with filter interface**

The program with filter interface has the possibility to access the VEMMI command dispatcher. The command dispatcher forwards the commands to the operative object, which decides to process them or forward them to the VEMMI local manager. This mechanism works also for the sending direction. The operative object has no visual effect on the terminal and cannot be activated directly by the user, but the started program may use the local UI on an application specific basis or through the VEMMI local manager. All these interfaces of such a program are terminal and implementation dependent and not further described in this document. If an operative object receives a VEMMI\_Close\_Object with its own OIN from the host or it is terminated by the VEMMI local manager, it shall stop and the VEMMI local manager shall continue. All data exchange channels to the VEMMI local manager are disabled.

There may be several operative objects with filter interface open at a given time.



**Figure 42: Programs with filter interfaces**

Attributes:

- Closed: The element shall be in the closed state;
- ProgName: The name of the program;
- ProgFilename: Filename of the program;
- ProgDescr: Description of the program;
- ProgAbout: Source identification of the program;
- ProgPar: List of parameters for the start of the program (optional);
- ProgType: Program type, standalone or with filter interface;
- DefaultDirectory: Default working directory for the operative object.

## 7.9 Bitmap Resource Object

- description: A bitmap object contains either the bitmap definition itself or only a reference to a file with the bitmap definition;
- Direct Bitmap Definition: The pixel matrix contains indices into the colour table or the RGB-components for each pixel. The attribute list is composed of two parts: the first is common to the two cases and the second is dependent from the colour definition. The first part is as follows:
  - a) BmWidth: Width of the bitmap in pixel;
  - b) BmHeight: Height of the bitmap in pixel;
  - c) BmCompr: Compression type for the colour list. This value is currently not defined and only uncompressed colour lists can be specified(optional).

In the case of a colour index definition the second part of the attribute list is as follows:

- a) BmBitsPerPixel: Number of bits per pixel. The value shall be 1, 4 or 8 (optional, default value: 1);
- b) BmClrEntry: The colour table index on which all other colour indices from the index list are based(optional, default value: 0);
- c) BmClrIdxList: List of colour indices. This list contains  $BmWidth * BmHeight$  colour indices. Depending on the number of bits per pixel, each colour index is in the range 0 to  $2^{BmBitsPerPixel} - 1$ . The colour indices are ordered row by row from left to right. The start

point is the lower left corner of the bitmap. A colour index 0 means a colour table index equal to BmClrEntry. All other indices are relative to BmClrEntry.

In the case of a colour component definition the second part of the parameter list is as follows:

- a) BmBitsPerComp: Number of bits per colour component. The value shall be in the range 1 to 8 (optional, default value: 8);
  - b) BmClrCompList: List of RGB triplets. This list contains BmWidth \* BmHeight triplets. The RGB-triplets are ordered row by row from left to right. The start point is the lower left corner of the bitmap.
- File Bitmap Definition: Such a bitmap is defined from a picture file stored in the terminal. The file has been transmitted with a resource-file transfer service or by other means. A parameter indicates the file coding. If the file could not be converted in a suitable bitmap (upon VEMMI\_Create\_Object) the terminal shall send an error message to the host:
- a) Filename: Name of the file with the bitmap definition;
  - b) PictFileType: Type of the picture file (one of JPEG, GIF, BMP).

The picture files conform to one of the following specifications:

- a) JPEG: Baseline system from ISO/IEC 10918 [16];
- b) GIF: Graphics Interchange Format (sm) Version 89a. CompuServe Incorporated Columbus, Ohio, USA. Although the version 89a is referenced a terminal has to support only the functions contained in the version 87;
- c) BMP: Microsoft Windows Device-Independent Bitmap (DIB) with RLE4, RLE8 and 1, 4, 8 or 24 bits per pixel.

#### **7.10 Videotex Resource Object**

- Description: A Videotex object contains either the Videotex content itself or only a reference to a file with the Videotex content.

Attributes:

- VTX: VTX content;
- Filename: Name of the file with the VTX content.

#### **7.11 Text Resource Object**

- description: This object defines text content as a resource which can be referenced via the TIN. It contains either the text content itself or a reference to a file with the text content. In both cases the text content can contain in-text attributes, with references to font resource objects.

Attributes - Direct Text Resource Definition

- Text: Text;
- FIN: Reference to a font resource object.

Attributes - File Text Resource Definition

- Filename: Name of the file with the text content;
- TextFileType: One of text or references to a font resource objects and text.

#### **7.12 Font resource object**

This object combines a set of text attributes in on font resource which can be referenced via the FIN.

Attributes:

- FnFamily: The font family: SWISS, ROMAN or FIXFONT. (optional, default value: SWISS);
- FnHeight: Character height. (optional, default value: 10);
- FnBold: Character weight bold (optional);

- FnUnderline: Underlined (optional);
- FnItalic: Italic (optional);
- FnColour: Font colour: colour index. (optional, default: black).

### 7.13 Metacode object

The metacode object contains VEMMI commands. This object provides an easy way to avoid unnecessary dialogue steps with the host application. If a VEMMI\_Open\_Object is applied to a metacode object the terminal copies all commands of the metacode object on top of the logical command input queue. The next command read from the command input queue is the first command from the metacode. The commands (the content of the metacode object) are processed in the same way as if they were received from the host application. For the content of the metacode object the following restrictions apply:

- the creation of a metacode object is not allowed;
- only host commands are allowed;
- a VEMMI\_Open\_Object command of another metacode object is not allowed (a VEMMI\_Open\_Blocking\_Object is allowed).

Attributes:

- VEMMICommands: VEMMI commands.

### 7.14 Sound Object

The purpose of this object is to link files with sound/audio content to the application. A parameter indicates the sound format. Opening the object starts the sound processing. At the end the object is automatically closed.

Attributes:

- Closed: The initial state of the object shall be closed;
- Filename: Filename;
- SoundForm: Sound format ( one of: WAVE, MIDI, ETS 300 149).

### 7.15 Video Object

The purpose of this object is to link files with moving video content to the application. Opening the object starts the video display process. At the end the object is automatically closed. The object defined at the time being has no specific video format. This shall be negotiated outside.

Attributes:

- Closed: The initial state of the object shall be closed;
- Filename: Filename.

### 7.16 Multimedia Resource Object

- Description: A Multimedia object contains a reference to a file with the Multimedia content.

Attributes:

- Filename: Name of the file with the Multimedia content.

## 7.17 The VEMMI content encoding identification catalogue

The following table provides an overview of content encoding identifications defined in the VEMMI catalogue:

**Table 38: VEMMI content encoding identification catalogue**

<b>For text data encoding:</b>	
	CCITT Recommendation T.51 [2] String as defined in annex A
	CCITT Recommendation T.52 [3]
	VEMMI high quality text
	ISO 8859 [13] series. (fully formed accented char.)
	ISO/IEC 10646-1 [15] (multi-octet character set (Unicode))
	Shift JIS Code (for Japanese characters)
<b>For still picture data encoding:</b>	
	ETS 300 177 [25] for photographic data.
	ISO/IEC 10918 [16] (JPEG)
	VEMMI device independent bitmap
	Graphics Interchange Format (sm) Version 89a. CompuServe Incorporated Columbus, Ohio, USA
	Microsoft Windows device-independent bitmap (DIB) with RLE4, RLE8 and 1, 4, 8 or 24 bits per pixel
<b>For graphic data encoding:</b>	
	ETS 300 073 [21] (Geometric Display, Videotex)
	ISO/IEC 8632 [12] (CGM)
<b>For audio data encoding:</b>	
	ETS 300 149 [24] (Audio syntax, Videotex)
	WAVE format
	MIDI format
<b>For moving picture data encoding and audiovisual data encoding:</b>	
	ISO/IEC 11172 -1 [17] (MPEG1 System)
	ISO/IEC 11172 -2 [18] (MPEG1 Video)
	ISO/IEC 13818 -1 [27] (MPEG2 System)
	ISO/IEC 13818 -2 [28] (MPEG2 Video)
	ITU-T Recommendation H.261 [7] (Video codec for AV services)
	ITU-T Recommendation H.320 [8] (videophone)
	AVI
	Quicktime
<b>For multimedia data encoding</b>	
	HTML

## 8 Complete coded representation of the VEMMI

### 8.1 Introduction

This clause contains the syntax notation, the VEMMI header and the syntax of the commands, objects, components and local actions.

### 8.2 Notation used

The notation a/b with a, b = 0 to 15 denotes a byte value in hexadecimal.

The bits in a byte are noted b8, b7, to , b1. The bit b8 is the high-order or most significant bit.

The syntax of the commands, objects, components and local actions is defined with a formal grammar using the following notation:

<symbol>	- nonterminal
<SYMBOL>	- terminal
<a/b>	- terminal, hexadecimal value: a, b = 0 to 15
<symbol>*	- 0 or more occurrences
<symbol>+	- 1 or more occurrences
<symbol>o	- optional ( 0 or 1 occurrences )
<symbol>n*	- 0 or 1 or 2 or n occurrences
<symbol-1>:= <symbol-2>	- symbol-1 has the syntax of symbol-2
<symbol-1> <symbol-2>	- symbol-1 always precedes symbol-2
<symbol-1>   <symbol-2>	- symbol-1 or alternatively symbol-2
<symbol-1: symbol-2>	- symbol with the stated value
{ comment }	- explanation of a symbol or production

The following list specifies the precedence of operators. The highest precedence level is at the top of the list. Precedence levels are separated by lines.

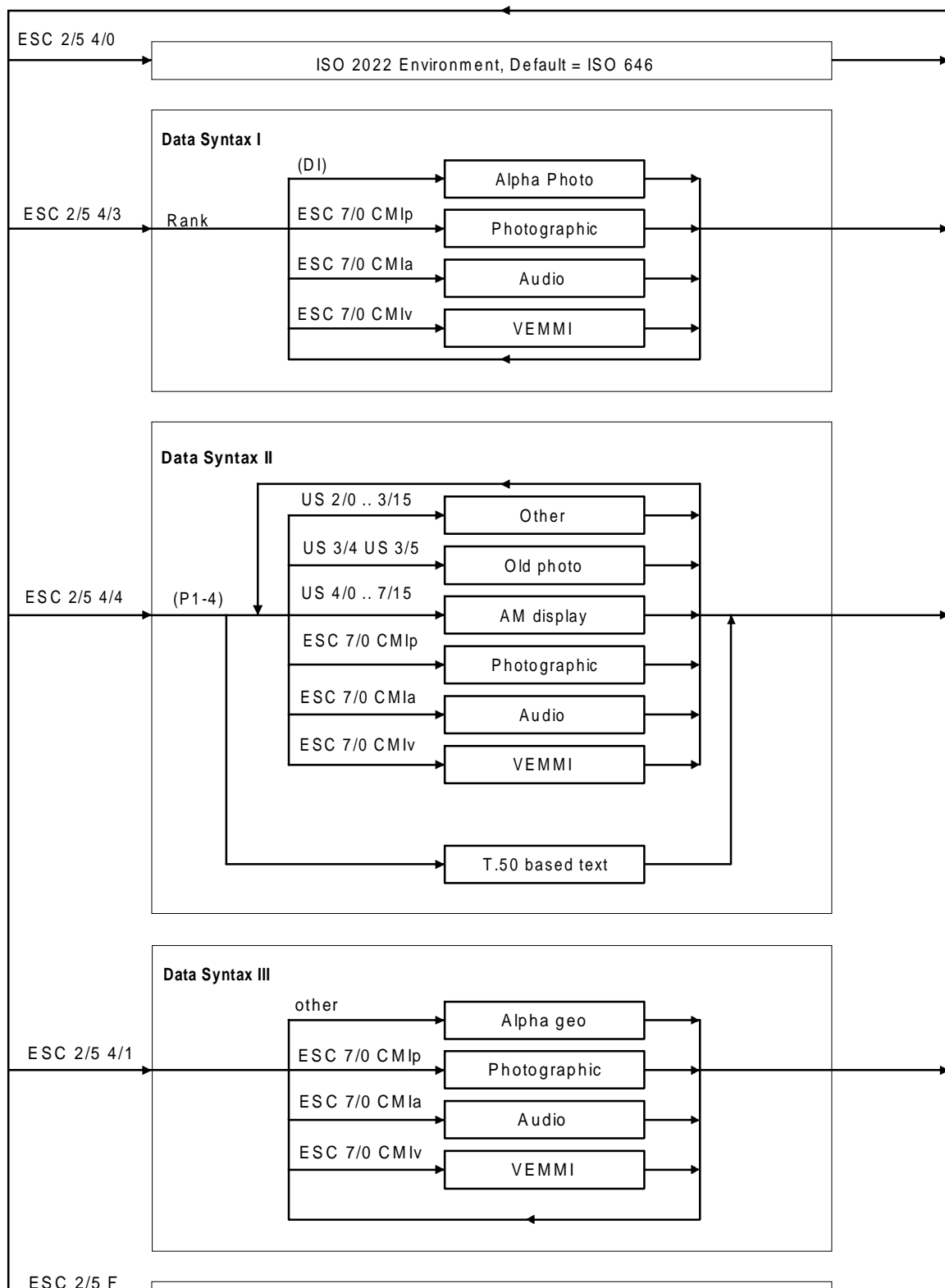
Operator	Meaning
*	0 or more occurrences
+	1 or more occurrences
o	optional ( 0 or 1 occurrences )
n*	0 or 1 or 2 or .. or n occurrences
	alternative
<..> <..>	sequence
:=	production

### 8.3 Overall switching of coding environment

ISO/IEC 9281 [14] describes a technique for identifying coding methods. The Videotex VEMMI mode is one of the coding methods identified by ISO/IEC 9281 [14]. The diagram in figure 43 gives an overview of the relationship between the Videotex data syntaxes and ISO/IEC 9281 [14] coding environments.

From an ISO 2022 [10] environment, a Videotex data syntax can be explicitly entered using an ESC 2/5 F code. This is also the mechanism used for entering from an ISO 2022 environment [10] into an ISO/IEC 9281 [14] environment. The F code "final byte" is allocated and registered, according to ISO 2022 [10] by the registration authority ISO 2375 [11]. According to ISO 2375 [11], appendix B, the Videotex data syntaxes are regarded as "coding systems different to that of ISO 2022 [10]". The F codes are 4/3 for CCITT data syntax I, 4/4 for CCITT data syntax II and 4/1 for CCITT data syntax III.

NOTE: The name of the terminal symbols in clause 8 does not refer to concepts presented in clause 4 to clause 7.



Key to figure 43

DI is data syntax I specific  
P is a profile in data syntax II  
F is a final code assigned by the ISO 2022 [10] registration authority  
CMla is any Coding Method Identifier (CMI) for Videotex audio data  
CMlp is any CMI for Videotex photographic data  
CMlv is any CMI for VEMMI data

Figure 43: Global switching mechanism

Rank is data syntax I specific.

Since a Videotex terminal usually begins operation, by default, in one of the data syntaxes, it shall not be mandatory to first send an ESC 2/5 F code (F is 4/1, 4/3 or 4/4). The diagram shows how these codes can be used to switch a Videotex terminal supporting more than one data syntax, from one data syntax to another.

### 8.3.1 Switching into the VEMMI mode

A Videotex terminal operating within one of the data syntaxes (i.e., a coding system other than that described by ISO 2022 [10]) can enter the ISO/IEC 9281 [14] environment of the VEMMI mode according to their own rules. In the case of Videotex for switching into the ISO/IEC 9281 [14] environment of the VEMMI mode, the Picture Code Delimiter (PCD) of the first Picture Element (PE) is used. The CMI is used to distinguish between picture coding methods. In the case of Videotex this shall be, for example, a distinction between audio, Photographic and VEMMI data.

NOTE: As ISO/IEC 9281 [4] was developed especially for the identification of picture coding in ISO 2022 [10] environments, the word "picture" is often used in the definitions, even when applicable to e.g. "audio". ISO has already accepted use of ISO/IEC 9281 [14] for non-pictorial information.

### 8.3.2 ISO/IEC 9281 syntax structure

The high level format of the syntax is as defined in ISO/IEC 9281 [14].

In the following description of the syntax, 8 bit coding is assumed, thus the word "byte" is used with bit 8 set to zero. The coding described in ISO 9281 [14] is also valid in a 7 bit environment. In this case the word "byte" shall be interpreted as meaning a "7 bit byte" and the most significant bit, bit 8, shall not be used.

The structure of the coding is as follows:

<VEMMI>:: =	<PE>
PE:: =	PCE PDE
PCE:: =	PCD CMI LI
PCD:: =	01/11 07/00
CMI:: =	PM PI
PM:: =	02/05 (Videotex VEMMI mode)
PI:: =	04/00
LI <sup>1</sup> :: =	x111 1111 <byte1><byte2>...<byte n>
<byte k>:: =	x10D DDDD (k=n)   x11D DDDD (1=<k<n)

x indicates don't care.

D indicates binary number 0 or 1.

Each piece of information, in this case encoded VEMMI data, is encoded as one or more Picture Entities (PEs). A PE (see figure 44) consists of Picture Control Entity (PCE) which is followed by the actual data packed into a Picture Data Entity (PDE).

---

<sup>1</sup> ISO/IEC 9281 [14], subclause 5.2.7 should be consulted for the description of the Length Indicator.



Picture Entity				
Picture Control Entity				Picture Data Entity
Picture Code Delimiter	CMI		Length Indicator	
	Picture Mode	Picture Identifier		

Figure 44: Structure of a PE

NOTE 1: In Videotex VEMMI mode, the size of a file containing an encoded VEMMI object can be rather large. One object may be transmitted in several PEs. The use of several PEs could facilitate the termination of the transmission of an object by the user.

The PCE consists of a PCD and a CMI followed by a Length Indicator (LI).

The PCD is a fixed sequence of two octets: 01/11 07/00.

NOTE 2: 01/11 is ESC.

The CMI consists of a PM octet, followed by a PI. For VEMMI the PM is 02/05 as registered by ISO 9281 [14]. The PI octet has the value 04/00.

The LI specifies the number of bytes in the PDE Its encoding is described in the following (except from ISO/IEC 9281-1 [14]):

b8	b7	b6	b5	b4	b3	b2	b1
X	ONE	ZERO or ONE					

Bit b8 of each byte shall be ignored. Bit b7 of each byte shall be set to ONE. Bit b6 of each byte shall be the Extension Flag. The LI value is specified in binary notation as an unsigned number using bits b5 to b1 with the weights  $2^4$ ,  $2^3$ ,  $2^2$ ,  $2^1$  and  $2^0$ , respectively. If the value of LI is less than or equal to 31 it shall be represented by one byte and the Extension Flag shall be set to ZERO. If the value of LI is larger than 31 it shall be represented by more than one byte. The most significant part of this value shall be recorded in the first byte. The Extension Flag shall be set to ONE in all bytes except the last where it shall be set to ZERO.

Values of LI:

1 byte:  $X111\ 1111$   
2 byte:  $0 \leq LI \leq 2^5 - 1$   
3 bytes:  $2^5 \leq LI \leq 2^{10} - 1$   
.  
.  
n bytes:  $2^{5(n-1)} \leq LI \leq 2^{5n} - 1$

Examples (the first byte shall always be encoded as 7FHex):

LI = 7FHex+31Dec

X	1	0	1	1	1	1	1
---	---	---	---	---	---	---	---

Second byte LI = 7FHex+33Dec

X	1	1	0	0	0	0	1
---	---	---	---	---	---	---	---

Second byte

X	1	0	0	0	0	0	1
---	---	---	---	---	---	---	---

Third byte

The VEMMI command structure is specified in subclause 9.1.

#### 8.4 VEMMI Command Syntax

Host Command	Command Syntax
VEMMI_Open	<open opc> <setup entity>
VEMMI_Close	<close opc>
VEMMI_Resume	<resume opc> <mode>o
VEMMI_Suspend	<suspend opc>
VEMMI_Close_All	<close all opc>
VEMMI_Identify_Term_Cap	<term cap request opc> <cap request>
VEMMI_Set_Options	<set options opc> <text type list>o
VEMMI_Reset_Col_Table	<reset col opc>
VEMMI_User_Lock	<user lock opc>
VEMMI_User_Unlock	<user unlock opc>
VEMMI_Open_Object	<open object opc> <oin spec>
VEMMI_Close_Object	<close object opc> <oin spec>
VEMMI_Destroy_Object	<destroy object opc> <oin spec>
VEMMI_Obj_Access_Disable	<access disable opc> <oin spec>
VEMMI_Obj_Access_Enable	<access enable opc> <oin spec>
VEMMI_Create_Object	<create object opc> <entity create object>
VEMMI_Delete_Outdated_Objects	<delete outdated objects opc> <entity versionlist>+
VEMMI_Modify_Component (see also 9.6)	<modify comp opc> <entity modify comp>
VEMMI_Obj_Location_Change	<object loc change opc> <oin> <xloc> <yloc>
VEMMI_Load_Col_Table	<load coltable opc> <entity load coltable>
VEMMI_Open_Application	<open appl opc> <entity open appl>
VEMMI_Store_Objects	<store objects opc> <oin spec>o
VEMMI_Erase_Objects	<erase objects opc> <oin spec>
VEMMI_Open_Blocking_Object	<open block object opc> <oin>
VEMMI_Resource-file_Transfer	<resource-file transfer opc> <transfer entity>

Terminal Command	Command Syntax
VEMMI_Store_Objects_Response	<store objects resp opc> <store result>
VEMMI_Object_Retransmission	<object retrans opc> <oin> <request type>o
VEMMI_Identify_Term_Cap_Resp	<term cap response opc> <cap response>+
VEMMI_Open_Application_Response	<open appl resp opc> <entity open appl resp>
VEMMI_User_Data	<user data opc> <entity user data>
VEMMI_Error	<error opc> <entity errorreport>
VEMMI_Transfer_Acknowledge	< resource-file transfer opc> <transfer id> <acknowledge>

NOTE: All values not assigned in the encoding of VEMMI now are reserved for future use.

<setup entity>:=	<version> <mode> o
<version>:=	<INTEGER> { Version }
<mode>:=	<mode opc> <native>   <private 1> { <private n> left for future extension }
<native>:=	<INTEGER: 0> { VEMMI as described in this document }
<private 1>:=	<page based>
<page based>:=	<INTEGER: 1> <basic page #> <input timeout>o
<basic page #>:=	<basic page # opc> <string>

<input timeout>:=	<input timeout opc> <INTEGER> { in seconds }
<cap request>:=	<local storage opc> o <data types opc> o <user language opc> <system information opc> o <file transfer opc> o
<cap response>:=	<version><term cap list>
<term cap list>:=	<local storage info> o <data type info> o <user language>o <system information> o <file transfer information> o
<local storage info>:=	<local storage opc> <true>   <false>
<data type info>:=	<data type opc> <text type> <still picture> <audio> o <graphic> o <audio visual> o <videotex syntax> o <multimedia type> o
<user language>:=	<user language opc> <string> { list of alphabetical language codes encoded according to ISO 639 Annex B (example: Dutch language: "NI") }
<system information>:=	<system information opc> <string>
<file transfer information>:=	<file transfer opc> <file transfer list>+ <end of list>
<file transfer list>:=	<INTEGER: 0>   { unknown } <INTEGER: 1>   { X-modem } <INTEGER: 2>   { Y-modem } <INTEGER: 3>   { Z-modem } <INTEGER: 4>   { Kermit } <INTEGER: 5>   { ETS 300 383, ISDN Eurofile } <INTEGER: 6>   { ETS 300 388 ISDN Simple FTAM } <INTEGER: 7>   { ETS 300 075 Processable Data File Transfer }
<text type>:=	<text type opc> <text type list> + <end of list>
<still picture>:=	<still picture opc> <still picture list> + <end of list>
<graphic>:=	<graphic opc> <graphic list> + <end of list>.
<audio>:=	<audio opc> <audio type list> + <end of list>
<audio visual>:=	<audio visual opc> <audio visual list> + <end of list>
<videotex syntax>:=	<videotex opc> <data syntax type> + <end of list>
<multimedia type>:=	<multimedia type opc> <multimedia type list> + <end of list>
<text type list>:=	<INTEGER: 0>   { T.51String } <INTEGER: 1>   { reserved } <INTEGER: 2>   { ISO 8859-1 } <INTEGER: 3>   { ISO 10646-1 } <INTEGER: 4>   { Shift JIS code } <INTEGER: 5>   { ISO 8859-2 } <INTEGER: 6>   { ISO 8859-3 } <INTEGER: 7>   { ISO 8859-4 } <INTEGER: 8>   { ISO 8859-5 } <INTEGER: 9>   { ISO 8859-6 } <INTEGER: 10>   { ISO 8859-7 } <INTEGER: 11>   { ISO 8859-8 } <INTEGER: 12>   { ISO 8859-9 } <INTEGER: 13>   { ISO 8859-10 }
{the values not used are reserved for	extensions of the ISO 8859 series extensions of the ISO 10646 series national variations of character sets }
<still picture list>:=	<INTEGER: 0>   { T.101 Annex F } <INTEGER: 1>   { JPEG } <INTEGER: 2>   { VEMMI DIB } <INTEGER: 3>   { GIF } <INTEGER: 4>   { MS DIB }
<graphic list>:=	<INTEGER: 0>   { ETS 300 073 (Geometric Display, Videotex) } <INTEGER: 1>   { CGM }
<audio type list>:=	<INTEGER: 0>   { ETS 300 149 } <INTEGER: 1>   { WAVE } <INTEGER: 2>   { MIDI }

<audio visual list>:=	<INTEGER: 0>   { ISO 11172-1, If Audio and Video is supported } <INTEGER: 1>   { ISO 11172-2, If only Video is supported } <INTEGER: 2>   { ISO 13818-1, If Audio and Video is supported } <INTEGER: 3>   { ISO 13818-2, If only Video is supported } <INTEGER: 4>   { H.261 } <INTEGER: 5>   { H.320 } <INTEGER: 6>   { AVI } <INTEGER: 7> { Quicktime }
<multimedia list>:=	<INTEGER: 0>   HTML } <INTEGER: 1>   { HTML 1 } <INTEGER: 2>   { HTML 2 } <INTEGER: 3> { HTML 3 }
<entity <b>create object</b> >:=	<oin> <save event> <object>
<save event>:=	<template> o   <autostore> o
<object>:=	<display object>   <resource object>   <metacode object>
<entity <b>versionlist</b> >:=	<timestamp> <oin spec>
<entity <b>load coltable</b> >:=	<colour entry> o <col rgb list>
<colour entry>:=	<colour entry opc> <INTEGER>
<col rgb list>:=	<list spec>
<entity <b>open appl</b> >:=	<applid> <appl add data> o <timestamp> <basic page #> o <input timeout> o
<applid>:=	<applid opc> <string>
<appl add data>:=	<appl add data opc> <string>
<entity <b>open appl resp</b> >:=	<result>
<entity <b>user data</b> >:=	<oin> <trigger> <component data> +
<trigger>:=	<INTEGER: 0>   { activation of a component } <INTEGER: 1> { validation of a component }
<component data>:=	<cin report>   { for menu choices, push buttons, pop-up menu, sensitive area, sensitive text } <text report>   { for text input fields, multimedia area } <list string report>   { for list box, combo box } <boolean report>   { for check box, radio button } <slider report>   { for slider } <locator report>   { for sensitive are with locator attribute set } <list index report> { for list box }
<cin report>:=	<INTEGER: 0> <cin>
<text report>:=	<INTEGER: 1> <cin> <string>
<list string report>:=	<INTEGER: 2> <cin> <string>
<boolean report>:=	<INTEGER: 3> <cin> <true>   <false>
<slider report>:=	<INTEGER: 4> <cin> <slider value> <negative>o
<locator report>:=	<INTEGER: 5> <cin> <xloc> <yloc>
<list index report>:=	<INTEGER: 6> <cin> <INTEGER>
<entity <b>errorreport</b> >:=	<error type> <error oin> o <error cin> o <error com code>o

<error type>:=	<INTEGER: 0>   { general error } <INTEGER: 1>   { unknown command } <INTEGER: 2>   { erroneous command } <INTEGER: 3>   { object syntax error } <INTEGER: 4>   { unexpected command } <INTEGER: 5>   { out of memory } <INTEGER: 6>   { cannot process audio objects } <INTEGER: 7>   { cannot process video objects } <INTEGER: 8>   { invalid colour index } <INTEGER: 9>   { file not found } <INTEGER: 10>   { conversion to bitmap failed } <INTEGER: 11>   { cannot process direct colour definition } <INTEGER: 12>   { operative object was rejected by the user } <INTEGER: 13>   { out of local storage space } <INTEGER: 14>   { a closed object has been destroyed } <INTEGER: 15>   { service not supported } <INTEGER: 16>   { object not supported } <INTEGER: 17>   { data content type not supported } <INTEGER: 18>   { VEMMI version not supported }
<error oin>:=	<error oin opc> <INTEGER>
<error cin>:=	<error cin opc> <INTEGER>
<error com code>:=	<error com code opc> <a/b> { a = 2 ..4, b = 0 .. 15 }
<transfer entity>:=	<data transfer>   <transfer abort>
<data transfer>:=	<INTEGER: 0> <transfer id> <current block number> <block>
<transfer id>:=	<INTEGER> { unique transfer identifier for each file }
<current block number>:=	<INTEGER> { from one to the total number of blocks }
<current block data length>:=	<INTEGER>
<block>:=	<file&block>   <current block>
<file&block>:=	<filespec> <current block>
<filespec>:=	<filename><creation date> <file length> <total number of blocks>
<creation date>:=	<string> { in the format YYYYMMDDHHMMSS }
<file length>:=	<INTEGER>
<total number of blocks>:=	<INTEGER>
<transfer abort>:=	<INTEGER: 1> <transfer id>
<current block>:=	<current block opc> <current block data length> <current block data>
<acknowledge>:=	<INTEGER: 0>   { transfer successful } <INTEGER: 1>   { transfer abort, no further information } <INTEGER: 2>   { transfer abort, invalid filename } <INTEGER: 3>   { transfer abort, directory not found } <INTEGER: 4>   { transfer abort, access not allowed } <INTEGER: 5>   { transfer abort, filesize exceeds allowed space } <INTEGER: 6>   { transfer abort, protocol error }
<current block data >:=	<a/b>+ { a,b = 0 to 15 }

## 8.5 Objects, components

<display object>:=	<application bar>   <button bar>   <pop-up menu>   <dialogue box>   <operative object>   <sound object>   <video object>   <message box>
--------------------	---

<application bar>:=	<appl bar opc> <application bar body> <menu item> +
<application bar body>:=	<ypos> o <closed> o <notaccessible> o <defactive> o <vertical> o
<menu item>:=	<bar menu choice> <pull down menu item>
<bar menu choice>:=	<bar menu choice opc> <menu choice>
<pull down menu item>:=	<pull down menu choice> <cascc menu choice>
<pull-down menu choice>:=	<pull down menu choice opc> <menu choice> <separated> o

<cascc menu choice>:=	<cascc menu choice opc> <menu choice> <separated> o
<menu choice>:=	<cin> <notaccessible> o <text> <locactact>o <locactval> o
<button bar>:=	<button bar opc> <button bar body> <push button comp> +
<button bar body>:=	<xpos> <ypos> <closed> o <notaccessible> o <vertical> o <defactive> o <modal> o <title> o
<push button comp>:=	<push button comp opc> <cin> <height> o <width> o <closed> o <notaccessible> o <button> <locactact> o <locactval> o
<pop-up menu>:=	<pop-up menu opc> <pop-up menu body> <pop-up menu choice> +
<pop-up menu body>:=	<xpos> <ypos> <closed> o <notaccessible> o <title> o <title font> o <defactive> o <modal> o
<pop-up menu choice>:=	<pop-up menu choice opc> <menu choice> <separated> o <cascc menu choice>
<dialogue box>:=	<dialogue box opc> <dialogue box body> < dialogue box components >*
<dialogue box components>:=	<separator>   <frame>   <graphic output area>   <text area>   <text input field>   <box push button>   <check box>   <radio button><list box>   <combo box>   <slider>   <sensitive area>   <text component>   <sensitive text>   <multimedia area>
<dialogue box body>:=	<xpos> o <ypos> o <width> o <height> o <closed> o <notaccessible> o <no border> o <title> o <defactive> o <modal> o <maximizable> o <background> o <store creation values> o
<background>:=	<colour>   <bitmapped>
<bitmapped>:=	<bin reference> <bitmap disptype> o <colour> o
<separator>:=	<separator opc> <cin> <xpos> o <ypos> o <width> o <height> o <closed> o <vertical> o <colour> o
<frame>:=	<frame opc> <cin> <xpos> o <ypos> o <width> o <height> o <closed> o <label> o <label font> o <colour> o
<graphic output area>:=	<graphic output area opc> <cin> <xpos> o <ypos> o <width> o <height> o <closed> o <graphic type>
<graphic type>:=	<bitmap>   <videotex>
<bitmap>:=	<output disp type> o <bin reference>
<output disp type>:=	<bitmap disp type>   <mapped>
< mapped>:=	<mapped opc> <xpos> o <ypos> o
<videotex>:=	<data syntax type> <direct videotex content>   <bin reference>
<data syntax type>:=	<INTEGER: 0>   { T.101 Annex B } <INTEGER: 1>   { T.101 Annex C, Profile 1 }  <INTEGER: 2>   { T.101 Annex C, Profile 2 } <INTEGER: 3>   { T.101 Annex C, Profile 3 } <INTEGER: 4>   { T.101 Annex C, Profile 4 } <INTEGER: 5>   { T.101 Annex C, Profile 5 } <INTEGER: 6> { T.101 Annex D }
<direct videotex content>:=	<videotex content opc> <nr of vtx bytes> <vtx bytes> +
<nr of vtx bytes>:=	<INTEGER> { number of Videotex bytes }
<text area>:=	<text area opc> <cin> <xpos> o <ypos> o <width> o <height> o <closed> o <initial font> o <maxim text> o <no scrolling tools> o <no format> o <no border> o <text area content> o
<text area content>:=	<direct text comp reference>   { cin of a text component } <in-text> +
<direct text comp reference>:=	<direct text comp reference opc> <cin reference>
<text component>:=	<text component opc> <cin> <prev text> o <current text> <next text> o
<prev text>:=	<prev text opc> <cin reference> { cin of a text component }
<next text>:=	<next text opc> <cin reference> { cin of a text component }

<current text>:=	<current text opc> <tin reference>   { tin of text resource object } <in-text> +
<in-text>:=	<text>   <in-text string>
<in-text string>:=	<fin reference>   { fin of a font resource object } <sensitive text reference>
<sensitive text reference>:=	<cin reference> <text> { cin of a sensitive text component }
<sensitive text>:=	<sensitive text opc> <cin> <notaccessible> o <locactact> o <locactval> o
<text input field>:=	<text input field opc> <cin> <xpos> o <ypos> o <width> o <height> o <closed> o <notaccessible> o <maxim text> o <default text> o <label> o <label font> o <input type> o <echo type> o <echo char> o <max char> o <max line> o <multiline> o <input transform> o <locactact> o <locactval> o
<box push button>:=	<box push button opc> <cin> <xpos> o <ypos> o <width> o <height> o <closed> o <notaccessible> o <button> <locactact> o <locactval> o
<check box>:=	<check box opc> <cin> <xpos> o <ypos> o <width> o <height> o <closed> o <notaccessible> o <label> o <label font> o <marked> o <locactact> o <locactval> o
<radio button>:=	<radio button opc> <cin> <xpos> o <ypos> o <width> o <height> o <closed> o <notaccessible> o <label> o <label font> o <group> o <marked> o <locactact> o <locactval> o
<list box>:=	<list box opc> <cin> <xpos> o <ypos> o <width> o <height> o <closed> o <notaccessible> o <default item> o <sorted> o <multiple choice> o <locactact> o <locactval> o <report index> o <arrayed> o <list text unit>
<arrayed>:=	<arrayed opc> <def column> + <end of list>
<def column>:=	<width> <alignment> o
<list text unit>:=	<list text unit opc> <list index> o <cell> + <item locact> o <item locval> o
<cell>:=	<text> <icon reference> o { one cell for each cell of a list item }
<combo box>:=	<combo box opc> <cin> <xpos> o <ypos> o <width> o <height> o <closed> o <notaccessible> o <default item> o <sorted> o <no drop> o <no consistency> o <no edit> o <max char> o <locactact> o <locactval> o <combo list text unit>
<combo list text unit>:=	<list index> o <text> <icon reference> o <item locact> o <item locval> o
<icon reference>:=	<bin reference>
<slider>:=	<slider opc> <cin> <xpos> o <ypos> o <height> o <width> o <closed> o <notaccessible> o <negative> o <min value> <negative> o <max value> <increment> o <initial info> o <direct in> o <vertical> o <locactact> o <locactval> o
<initial info>:=	<negative> o <initial value>
<sensitive area>:=	<sensitive area opc> <cin> <content associated> o <xpos> o <ypos> o <width> o <height> o <closed> o <notaccessible> o <locator> o <locactact> o <locactval> o
<content associated>:=	<content associated opc> <associated graphic area reference>
<multimedia area>:=	<multimedia area opc> <cin> <xpos> o <ypos> o <width> o <height> o <closed> o <notaccessible> o <maximizable> o <no border> o <multimedia content type> o <multimedia content> o <direct navigation> o <non-controlled mode> o
<multimedia content type>:=	<multimedia content type opc> <multimedia content id>
<multimedia content id>:=	<INTEGER: 0> { HTML }
<multimedia content>:=	<direct data>  <min reference>
<direct data>:=	<direct data opc> <data length> <data>

**<operative object>:=** <operative object opc> <closed> o <program filename> <program name> <program description> <program about> <program parameter>o <program type> <default directory>o

**<sound object>:=** <sound object opc> <closed> o <filename> <sound format>

**<sound format>:=** <INTEGER: 0> | { wave format }  
<INTEGER: 1> | { MIDI format }  
<INTEGER: 2> { ETS 300 149 }

**<video object>:=** <video object opc> <closed> o <filename> <audio visual list>

**<message box>:=** <message box opc> <xpos> o <ypos> o <width> o <height> o <closed> o <no border> o <title> o <modal> o <message type> o <max time> o <destroy event> o <no sound> o <attributed text> +

**<message type>:=** <message type opc> <message>

**<message>:=** <INTEGER: 0> | { general message }  
<INTEGER: 1> | { information message }  
<INTEGER: 2> | { warning message }  
<INTEGER: 3> { action message }

**<destroy event>:=** <destroy event opc> <event>

**<event>:=** <INTEGER: 0> | { destroy by any user action }  
<INTEGER: 1> | { destroy by validation of implicit defined button }  
<INTEGER: 2> | { destroy by a command from the host }  
<INTEGER: 3> | { close by validation of implicit defined button }  
<INTEGER: 4> | { close by any user interaction }  
<INTEGER: 5> { no implicit destroy event defined }

**<max time>:=** <max time opc> <INTEGER> { time period in seconds }

<b>&lt;resource object&gt;:=</b>	<bitmap object>   <font object>   <text object>   <videotex object>   <multimedia object>
----------------------------------	--

**<bitmap object>:=** <bitmap object opc> <bitmap data> | <bitmap file>

**<bitmap data>:=** <INTEGER: 0> <bmwidth> <bmheight> <bmcompr> o <pixel definition>

**<bmwidth>:=** <INTEGER>

**<bmheight>:=** <INTEGER>

**<bmcompr>:=** <bmcompr opc> <INTEGER: 0> { no compression }

**<pixel definition>:=** <index def> | <rgb def>

**<index def>:=** <index def opc> <bits per pixel> o <colour entry>o <colour list>

**<bits per pixel>:=** <bits per unit>

**<colour entry>:=** <colour entry opc> <INTEGER>

**<colour list>:=** <list spec> { bmwidth \* bmheight integers }

**<rgb def>:=** <rgb def opc> <bits per component> o <component list>

**<bits per component>:=** <bits per unit>

**<component list>:=** <list spec> { 3 \* bmwidth \* bmheight integers }

**<bits per unit>:=** <bits per unit opc> <INTEGER>

**<bitmap file>:=** <INTEGER: 1> <filename> <pict file type>

**<pict file type>:=** <INTEGER: 0> | { JPEG }  
<INTEGER: 1> | { GIF }  
<INTEGER: 2> { BMP }

**<font object>:=** <font object opc> <fn family> o <fn height> o <fn bold> o <fn underline> o <fn italic> o <fn colour> o

**<fn family>:=** <fn family opc> <family>

**<family>:=** <INTEGER: 0> | { SWISS }  
<INTEGER: 1> | { ROMAN }  
<INTEGER: 2> { FIXFONT }

**<fn height>:=** <fn height opc> <INTEGER>

**<fn bold>:=** <fn bold opc>

**<fn underline>:=** <fn underline opc>



<fn italic>:=	<fn italic opc>
<fn colour>:=	<fn colour opc> <INTEGER>
<text object>:=	<text object opc> <in-text> +   <text file>
<text file>:=	<file name> <text file type>
<text file type>:=	<INTEGER: 0>   { "in-text" format } <INTEGER: 1> { plain text, incl. CR, LF }
<videotex object>:=	<videotex object opc> <data syntax type> <direct videotex content>   <filename>
<multimedia object>:=	<multimedia object opc> <multimedia content type> o <filename>
<metacode object>:=	<metacode object opc> <command> + <command end>

## 8.6 Local actions

<locactact>:=	<locactact opc> <local action> +
<locactval>:=	<locactval opc> <local action> +
<local action>:=	<report command>   <general command>   <specific command>
<report command>:=	<loc command opc> <report type>
<report type>:=	<INTEGER: 0>   { report OIN, CIN } <INTEGER: 1>   { report current value } <INTEGER: 2>   { report all values } <INTEGER: 3> { report all changed values }
<general command>:=	<loc command opc> <general command type>
<general command type>:=	<INTEGER: 50>   { user lock } <INTEGER: 51>   { set initial values and states in the parent obj. } <INTEGER: 52> { No operation }
<specific command>:=	<loc command opc> <specific command type> <oin spec>   <cin spec>
<specific command type>:=	<INTEGER: 100>   { open components of the parent object } <INTEGER: 101>   { close components of the parent object } <INTEGER: 102>   { open objects } <INTEGER: 103>   { close objects } <INTEGER: 104>   { change comp. of the parent obj. to inacc. } <INTEGER: 105>   { change comp. of the parent obj. to access. } <INTEGER: 106>   { destroy objects } <INTEGER: 107> { open blocking object (only one) }

## 9 Encoding

### 9.1 Command structure

The PDE of one PE carries one or more VEMMI commands. The last VEMMI command of one PE can be split on more than one PE. This is indicated with the first byte of the PDE, called "more data indicator". The syntax of the PDE is as follows:

<PDE>:=	<mdi><command> +
<mdi>:=	<2/14>   {; MORE, no translation } <3/14>   {; MORE, 3 in 4 encoding } <4/14>   {; MORE, 7 shift encoding } <2/15>   {; LAST, no translation } <3/15>   {; LAST, 3 in 4 encoding } <4/15>   {; LAST, 7 shift encoding }

- "MORE": the last command is only partly contained in this PE, it continues in the next PE;
- "LAST": the last command is completely contained in the PE.

The value of the MDI is used to identify the transcoding algorithm used to transcode the VEMMI code in "<command>+". The following three transcoding algorithms are defined:

- no translation;
- 3 in 4 encoding (see ITU-T Recommendation T.101 [4]);
- 7 shift encoding (see ITU-T Recommendation T.101 [4]).

The translation mode can be different in both directions.

NOTE: Splitting commands over more than one PE is not reflected in the grammar specification.

One or several VEMMI commands are preceded by the VEMMI-Header. This header is used to identify VEMMI data in the data stream of the standard application. Such a sequence of VEMMI commands is called the Data Entity (DE). The last command of a data entity can be split across more than one data entity. This is indicated by the MDI. Examples:

Three commands contained in one data entity:

VEMMI Header	MDI (last)	Data Entity		
		Command 1	Command 2	Command 3

Three commands in two data entities, one is split:

VEMMI Header	MDI (more)	Data Entity		
		Command 1	Command 2 (first part)	

VEMMI Header	MDI (last)	Data Entity		
		Command 2 (last part)	Command 3	

A VEMMI\_Create\_Object command creating a Metacode object shall always be transmitted in a complete sequence of PEs.

## 9.2 Object, component and attribute structure

Object structure:

The objects have one of the following structures:

- <object code> <object body> <component>\*;
- <object code> <attribute>+;
- <object code> <command>+.

Component structure: <component code> <cin> <attribute>\*

Attribute structure:

The attributes have one of the following structures:

- <attribute code> { the attribute is defined only by the code };
- <attribute code, value> { the attribute is defined by the code and the value }
- <attribute> { the attribute is defined by another attribute };
- <attribute code, attribute> { the attribute is defined by the code and another attribute }.

### 9.3 Terminal symbols encoding

#### 9.3.1 Opcodes

Opcodes are used to identify the parts of the VEMMI commands. They are used to encode:

- command codes;
- object codes;
- component codes;
- the MDI code;
- attribute codes.

The terminal symbol notation is: <a/b> with a, b = 0 to 15 (hexadecimal notation).

#### 9.3.2 Integers

This format is used to encode non-negative values with no fractional part. An integer consists of a byte sequence of one or more bytes.

Coding rules:

- bit 8 set to 1 means no extension, it is the last byte of the sequence;
- bit 8 set to 0 and bit 7 set to 1 indicates an extension, it is not the last byte;
- the encoded value is specified:
  - a) for one byte integers by bits 7 to 1;
  - b) for multi-byte integers by bits 6 to 1 from all bytes except the last byte, and bits 7 to 1 from the last byte.
- the most significant part of the value is coded in the first byte;
- the least significant part of the value is coded in the last byte.

Symbols using integers:

<INTEGER>

<false>:= <INTEGER: 0>

<true>:= <INTEGER: 1>

EXAMPLE 1: Coded representation of the number 37.

b8	b7	b6	b5	b4	b3	b2	b1
1	0	1	0	0	1	0	1
No ext	integer value bits						

EXAMPLE 2: Coded representation of the number 1741:

Byte 1	b8	b7	b6	b5	b4	b3	b2	b1
	0	1	0	0	1	1	0	1
	Ext		Most significant bits					

Byte 2	b8	b7	b6	b5	b4	b3	b2	b1
	1	1	0	0	1	1	0	1
	No ext	Least significant bits						

#### 9.3.3 Enumerated

This data type denotes a value from a set of standardized values.

They are encoded as integers with the notation: <INTEGER: n> with n as a decimal value.

### 9.3.4 Strings

This data type denotes a sequence of characters with the following structure:

<string>:=	<latin string>   <non latin string>   <unicode string>
<latin string>:=	<character> * <end of string>
<character>:=	<a/b>   <c/b>   <CR>   <LF>   <ESC>   <SS2> { a = 2 .. 7, b = 0 to 15, c = 10 to 15 characters coded according ISO.8859 [13] or CCITT Recommendation T.51 [2] String }
<end of string>:=	<9/12> { ST (string terminator) from ISO 6429 [x] }
<CR>:=	<0/13>
<LF>:=	<0/10>
<ESC>:=	<1/11>
<SS2>:=	<1/9>
<non latin string>:=	<shift JIS string>   <T.52 string>
<shift JIS string>:=	<shift JIS character> * <end of JIS string>
<shift JIS character>:=	<a/b>   <c/d> <e/f>   <CR>   <LF> { a = 2 to 7, 10 to 13; b = 0 to 15, c = 8,9,14,15; d = 0 to 15; e = 4 to 15; f = 0 to 15 }
<end of JIS string>:=	<1/11> <5/12> { ST (string terminator) from ISO 6429 [x] }
<T.52 string>:=	for further study
<unicode string>:=	<uni char> * <end of unicode string> { string according ISO 10646 [15]-1 (BMP) }
<uni char>:=	<a/b a/b> { a = 0 to 15, b = 0 to 15 }
<end of unicode>:=	<ST version>   <FF version>
<ST version>:=	<0/0> <9/12>
<FF version>:=	<F/F> <F/F>

### 9.3.5 NDC

This data type denotes horizontal or vertical scalar values in the NDC space. A NDC is a fixed point real number from the range 0.0 (inclusive) to 1.0 (exclusive).

Coding rules:

- a NDC value is coded in a sequence of one or more bytes;
- bit 8 set to 1 means no extension, it is the last byte of the sequence;
- bit 8 set to 0 and bit 7 set to 1 indicates an extension, it is not the last byte;
- the encoded value is specified:
  - a) for one byte NDC by bits 7 to 1;
  - b) for multi-byte NDC by bits 6 to 1 from all bytes except the last byte and bits 7 to 1 from the last byte.
- the binary radix point is not coded, but assumed to be:
  - a) for one byte NDC next left to the bit 7;
  - b) for multi-byte NDC next left to bit 6 in the first byte.
- the most significant part of the value is coded in the first byte;
- the least significant part of the value is coded in the last byte.

The terminal symbol notation is: <NDC>.

EXAMPLE 1: Coded representation of the number  $0,3125_{10} = 0,0101_2$

b8	b7	b6	b5	b4	b3	b2	b1
1	0	1	0	1	0	0	0
No ext	NDC value bits						

EXAMPLE 2: Coded representation of the NDC  $0,205078125_{10} = 0,001000101_2$

Byte 1	b8	b7	b6	b5	b4	b3	b2	b1
	0	1	0	0	1	1	0	1
	Ext.		Most significant bits					

Byte 2	b8	b7	b6	b5	b4	b3	b2	b1
	1	0	0	1	0	0	0	0
	No ext		Least significant bits					

#### 9.4 Attributes and lower level symbols

<alignment>:=	<alignment opc> <INTEGER: 0>   { left } <INTEGER: 1>   { right } <INTEGER: 2> { centred }
<associated graphic area reference>:=	<cin reference>
<attributed text>:=	<text>   <cin reference>
<autostore>:=	<7/8>
<bin>:=	<INTEGER>
<bin reference>:=	<11/8> <retransmission rule B> o <bin>
<bitmap disp type>:=	<11/11> <INTEGER: 0>   { stretched } <INTEGER: 1>   { centred } <INTEGER: 2> { tiled }
<block index>:=	<12/4> <INTEGER>
<button>:=	<button text>   <bin reference>
<button text>:=	<label font> o <text>
<cin>:=	<INTEGER>
<cin reference>:=	<12/7> <INTEGER>{ reference to a component }
<cin spec>:=	<in spec>+
<closed>:=	<6/8>
<colour>:=	<11/10> <INTEGER>{ colour index }
<command end>:=	<2/15>
<data>:=	<a a> {a=0 to 15}
<data length>:=	<INTEGER>
<defactive>:=	<11/7> <INTEGER>{ the referenced cin }
<default directory>:=	<14/2> <string>
<default item>:=	<12/3> <INTEGER>
<default text>:=	<text>
<delete list item>:=	<6/12>
<direct navigation>:=	<6/3>
<direct in>:=	<7/10>
<display character>:=	<a/b>{ a = 2 to 15, b = 0 to 15}
<echo char>:=	<14/6> <display character>
<echo type>:=	<12/0><INTEGER: 0>   { local echo } <INTEGER: 1>   { no echo } <INTEGER: 2> { echo defined character }

<end of list>:=	<3/15>
<filename>:=	<14/8> <string>
<fin>:=	<INTEGER>
<fin reference>:=	<10/13> <INTEGER>{ reference to a font resource object }
<group>:=	<12/2> <INTEGER>
<height>:=	<9/2> <NDC>
<in spec>:=	<list spec>   <range spec>   <one in> { defines a list, a range or a single identification number }
<list spec>:=	<11/0> <INTEGER> + <end of list>
<range spec>:=	<13/0> <from> <to>
<from>:=	<INTEGER>
<to>:=	<INTEGER>
<one in>:=	<13/5> <INTEGER>
<increment>:=	<12/10> <INTEGER>
<initial font>:=	<11/12> <INTEGER>{ the referenced fin }
<initial value>:=	<12/11> <INTEGER>
<input type>:=	<11/14> <INTEGER: 0>   { any character } <INTEGER: 1>   { alphabetic } <INTEGER: 2>   { numeric } <INTEGER: 3> { alphanumeric }
<input transform>:=	<11/5><INTEGER: 0>   { no transformation } <INTEGER: 1>   { to lower } <INTEGER: 2> { to upper }
<item locact>:=	<7/13> <local action> +
<item locval>:=	<7/14> <local action> +
<label font>:=	<11/13> <INTEGER>{ the referenced fin }
<label>:=	<14/5> <string>
<list index>:=	<12/4> <INTEGER>
<locator>:=	<6/0>
<marked>:=	<7/1>
<max char>:=	<12/1> <INTEGER>
<max line>:=	<12/5> <INTEGER>
<max value>:=	<12/9> <INTEGER>
<maxim text>:=	<7/0>
<maximizable>:=	<6/14>
<min value>:=	<12/8> <INTEGER>
<min reference>:=	<10/14> <retransmission rule B> o <INTEGER>
<modal>:=	<6/12>
<multiline>:=	<6/15>
<multiple choice>:=	<7/2>
<negative>:=	<7/9>
<no border>:=	<6/13>
<no consistency>:=	<7/15>
<no drop>:=	<7/3>
<no edit>:=	<7/4>
<no format>:=	<7/6>
<no scrolling tools>:=	<7/5>
<no sound>:=	<7/11>
<non-controlled mode>:=	<7/4>
<notaccessible>:=	<6/11>
<num blocks>:=	<10/8> <INTEGER>
<oin>:=	<INTEGER>
<oin spec>:=	<in spec> + { defines object identifiers }
<output height>:=	<9/4> <NDC>

<program about>:=	<string>
<program description>:=	<string>
<program filename>:=	<filename>
<program name>:=	<string>
<program parameter>:=	<program parameter opc> <string>
<program type>:=	<13/1> <INTEGER: 0>   { standalone program } <INTEGER: 1> { program with filter interface }
<report index>:=	<6/2>
<result>:=	<true>   <false>
<request type>:=	<11/6> <INTEGER: 0>   { object request } <INTEGER: 1>   { request for resource-file } <INTEGER: 2> { request for text component }
<retransmission rule B>:=	<6/15>
<scrolling tools>:=	<11/1> <INTEGER: 0>   { terminal dependent } <INTEGER: 1>   { two column scrolling } <INTEGER: 2> { two row scrolling }
<slider value>:=	<INTEGER>
<store result>:=	<true>   <false>
<separated>:=	<6/10>
<sorted>:=	<7/12>
<template>:=	<7/7>
<text>:=	<14/3> <string>
<timestamp>:=	<12/14> <INTEGER>
<tin reference>:=	<10/15> <INTEGER> { reference to a text resource object }
<title font>:=	<11/9> <INTEGER> { the referenced fin }
<title>:=	<14/4> <string>
<vertical>:=	<6/7>
<vin reference>:=	<10/12> <retransmission rule B> o <INTEGER>
<width>:=	<9/3> <NDC>
<xloc>:=	<NDC>
<xpos>:=	<9/0> <NDC>
<yes/no>:=	<true>   <false>
<ypos>:=	<9/1> <NDC>
<yloc>:=	<NDC>
<vtx byte>:=	<a/a> {a= 0 to 15}

## 9.5 Opcodes

In the following two tables, the suffix "opc" from the symbols of the command syntax specification is omitted.

The code positions currently not used are reserved for future extensions.

**Table 39: Host commands opcodes**

	2	3
0	open	create object
1	close	delete outdated objects
2	resume	modify component (see also 9.6.)
3	suspend	obj location change
4	close all	load coltable
5	reset col	open application
6	user lock	store objects
7	user unlock	erase objects
8	open object	open blocking object
9	close object	resource-file transfer
10	destroy object	term cap request
11	access disable	
12	access enable	set options
13		
14		
15		

**Table 40: Terminal commands opcodes**

	2	3
0	store objects resp	
1	object retrans	
2	open appl resp	
3	user data	
4	error	
5		
6	term cap resp	
7		
8		
9		resource-file transfer
10		
11		
12		
13		
14		
15		

NOTE: The following tables contain symbols which have the same code. Their names are separated with a "/".



Table 41: Opcodes (part 1)

	Objects	"End"-codes	Components	
	2	3	4	5
0	application bar		bar menu choice, menu choice	slider
1	button bar		pull-down choice	text component
2	pop-up menu		push button comp	sensitive text
3	dialogue box		pop-up choice	casc menu choice
4	multimedia object		separator	multimedia area
5	operative object		frame	
6	sound object		graphic output area	
7	video object		text area	
8	message box		text input field	
9	metacode object		box push button	
10	bitmap object		check box	
11	font object		radio button	
12	text object		list box	
13	videotex object		combo box	
14		command end	sensitive area	
15		end of list		

Table 42: Opcodes (part 2)

	Boolean attributes, higher layer symbols		Coordinates, dimensions
	6	7	9
0	fn bold/ locator	maxim text/ arrayed	xpos
1	fn underline/ local storage	marked/ list text unit	ypos
2	fn italic/ report index	multiple choice/ direct text comp reference	height
3	index def/ direct navigation	no drop/ content associated	width
4	rgb def/ alignment	no edit/non- controlled mode	output height
5	locactact	no scrolling tools	
6	locactval	no format	
7	vertical	template	
8	closed	autostore	
9	mode	negative	
10	separated	direct in/ more blocks	
11	notaccessible	no sound/ store creation values	
12	modal/delete list item	sorted/direct data	end of string
13	no border	item locact	
14	maximizable	item locval	
15	multiline/ retransmission rule B	no consistency	

Table 43: Opcodes (part 3)

	Integer attributes				String attrib.
	10	11	12	13	14
0	fn family	list spec	echo type	range spec	applid/ user language
1	fn height/ data types	error oin/ scrolling tools	max char	program type	appl add data/ system information
2	fn colour/ text types	error cin/ multimedia type	group	prev text	default directory
3	bmcompr/ still picture	error com code	default item	current text	text
4	bits per unit/ videotex	local commands	list index/ block index	next text	title
5	colour entry/ audio	input transform	max line	one in	label
6	graphic	request type	multimedia content type		echo char
7	audiovisual	defective	cin reference		vtx content
8	num blocks/ file transfer	bin reference	min value		filename
9	message type	title font	max value		program parameter
10	destroy event	colour	increment		current block
11	default vtx	bitmap disptype	initial value/mapped		
12	vin reference	initial font	max time		basic page #
13	fin reference	label font	input timeout		
14	min reference	input type	timestamp		
15	tin reference				

## 9.6 Syntax of the VEMMI\_Modify\_Component

VEMMI\_Modify\_Component is used to:

- modify existing components or attributes;
- add one or more new components to an existing object.

All attributes which are modifiable have respectively their higher symbol definitions summarised in the following:

attributed text	echo char	label	marked	text
bin reference	echo type	label font	message	tin reference
cin reference	group	destroy event	min value	title
closed	increment	list text unit	negative	graphic type
colour	initial font	locactval	next text	
current text	initial info	locactact	no sound	
default item	input type	max char	notaccessible	

The modification of attributes is achieved by replacing the old value with a new value. There is no partial modification of attributes. Only the transmitted attributes are applied; the attributes that are not transmitted remain unchanged.

The modification of boolean attributes is achieved by transmitting the same attribute again. The host application can either set or reset the value for a boolean attribute.

In order to set a boolean attribute, the host application shall transmit the attribute once (this is equivalent to the object creation syntax). In order to reset a boolean attribute, the host application shall transmit the attribute twice immediately following each other.

In the following example a pop-up menu offers four choices to the user, two accessible (CINs 35, 36) and two not accessible (CINs 37, 38):

Start New Game
Beginner Level
Intermediate Level
Advanced Level

Upon reception of VEMMI\_Modify\_Component with:

- <oin: 2>;
- <cin: 37> <NotAccessible> <NotAccessible>.

the menu is changed, one inaccessible component is made accessible.

Start New Game
Beginner Level
Intermediate Level
Advanced Level

If VEMMI\_Modify\_Component adds a new component, the command has the same syntax as VEMMI\_Create\_Object. In this case, for the optional attributes the default values are applied. That is the same rule as VEMMI\_Create\_Object uses.

Specific requirements for the modification of list items are described in subclause 7.6.1.11 and subclause 7.6.1.12.

If an object was created without a title, the title attribute shall be ignored in a modify command referencing this object. If a component was created without a label the label attribute shall be ignored in a modify command referencing this component.

Based on the above description, the VEMMI\_Modify\_Component command syntax is as follows:

```

<entity modify comp>:=
<modify comp>:=
    <oin> <modify comp> | <new comp> +
    <modify application bar> | <modify button bar> | <modify pop-up
    menu> | <modify dialogue box> <modify message box>
<modify application bar>:=
    <application bar opc> <modify menu item > *
<modify menu item >:=
    <menu choice opc> <cin> <notaccessible> 2* <text>o <locactact>
    o <locactval> o <separated> 2*
<modify button bar>:=
    <button bar opc> <title> o <modify push button > *
<modify push button >:=
    <push button opc> <cin> <closed> 2* <notaccessible>2*
    <button> o <locactact> o <locactval> o
<modify pop-up menu>:=
    <pop-up menu opc> <title> o <modify pop-up choice> *
<modify pop-up choice>:=
    <pop-up choice opc> <cin> <notaccessible> 2* <text> o
    <locactact> o <locactval> o <separated> 2*
<modify dialogue box>:=
    <dialogue box opc> <modify dialogue box body> o < modify
    dialogue box comp > *
<modify dialogue box body>:=
    <title> o
< modify dialogue box comp>:=
    <modify separator> | <modify frame> | <modify graphic output
    area> | <modify text area> | <modify text input field> | <modify box
    push button> | <modify check box> | <modify radio button> |
    <modify list box> | <modify combo box> | <modify slider> | <modify
    sensitive area> | <modify text component> | <modify sensitive text>
    | <modify multimedia area>
<modify separator>:=
    <separator opc> <cin> <closed> 2* <colour> o
<modify frame>:=
    <frame opc> <cin> <closed> 2* <label> o <label font> o <colour>o
<modify graphic output area>:=
    <graphic output area opc> <cin> <closed> 2* <graphic type> o

```

<modify text area>:=	<text area opc> <cin> <closed> 2* <initial font> o <text area content> o
<modify text component>:=	<text component opc> <cin> <prev text> o <current text> o <next text> o
<modify sensitive text>:=	<sensitive text opc> <cin> <notaccessible> 2* <locactact> o <locactval> o
<modify text input field>:=	<text input field opc> <cin> <closed> 2* <notaccessible> 2* <default text> o <label> o <label font> o <input type> o <echo type> o <echo char> o <max char> o <locactact> o <locactval> o
<modify box push button>:=	<box push button opc> <cin> <closed> 2* <notaccessible> 2* <button> o <locactact> o <locactval> o
<modify check box>:=	<check box opc> <cin> <closed> 2* <notaccessible> 2* <label> o <label font> o <marked> 2* <locactact> o <locactval> o
<modify radio button>:=	<radio button opc> <cin> <closed> 2* <notaccessible> 2* <label> o <label font> o <marked> 2* <locactact> o <locactval> o
<modify list box>:=	<list box opc> <cin> <closed> 2* <notaccessible> 2* <default item> o <locactact> o <locactval> o <modify list text unit> *
<modify list text unit>:=	<list text unit opc> <list index> <change or delete>
<change or delete>:=	<change>   <delete list item> o
<change>:=	<cell> + <item locact> o <item locval> o
<modify combo box>:=	<combo box opc> <cin> <closed> 2* <notaccessible> 2* <default item> o <locactact> o <locactval> o <modify combo list text unit> *
<modify combo list text unit>:=	<list index> <combo change or delete>
<combo change or delete>:=	<combo change>   <delete list item>
<combo change>:=	<text> o <icon reference> o <item locact> o <item locval> o
<modify slider>:=	<slider opc> <cin> <closed> 2* <notaccessible> 2* <modify min value> o <modify max value> o <increment> o <initial info> o <locactact> o <locactval> o
<modify min value>:=	<negative> o <min value>
<modify max value>:=	<negative> o <max value>
<initial info>:=	<negative> o <initial value>
<modify sensitive area>:=	<sensitive area opc> <cin> <closed> 2* <notaccessible> 2* <locactact> o <locactval> o
<modify multimedia area>:=	<multimedia area opc> <cin> <closed> 2* <notaccessible> 2* <maximizable> 2* <multimedia content> o <direct navigation> 2* <non-controlled mode> 2*
<modify message box>:=	<message box opc> <title> o <message type> o <destroy event> o <no sound> 2* <attributed text> *
<new comp>:=	<new menu choice>   <push button comp>   <separator>   <frame>   <graphic output area>   <text area>   <text component>   <sensitive text>   <text input field>   <box push button>   <check box>   <radio button>   <list box>   <combo box>   <slider>   <sensitive area>   <multimedia area>
<new menu choice>:=	<logical group> <cin reference>o <menu choice>{ see NOTE }
<logical group>:=	<INTEGER: 0>   { bar } <INTEGER: 1>   { pull-down, pop-up } <INTEGER: 2> { cascading }

If a new menu choice is added to an Application Bar or a Pop-Up menu the "logical group" attribute specifies to which logical group of the object the component shall be added. The "cin reference" attribute represents the CIN of an already defined menu choice of the object and indicates the position at which the menu choice shall be added. If the menu choice that is referenced with the CIN is in the same logical group identified with the "logical group" attribute, the new menu choice shall be inserted after the specified menu choice. If the menu choice that is referenced with the CIN is in a higher logical group than identified with the "logical group" attribute, the new menu choice shall be the first item of the associated group. If the "cin reference" attribute is not present the menu choice shall be inserted at the first position of the highest

logical group. It is not possible to associate a cascading menu directly to the Bar logical group of the Application Bar.

## 9.7 Defaults

The default values of attributes are applied when an optional attribute is not contained in the element definition. An exception makes VEMMI\_Modify\_Component applied on existing components, see subclause 9.6. The default values depend from the attribute data type and from the object/component to which they belong.

For the default value of a Boolean attribute (these are coded with the opcode only) the logical negative meaning compared with its definition shall be assumed. A default value of a Boolean attribute can not be coded and is never contained in the element definition.

**Table 44: Boolean attributes defaults**

Attribute	Default value	Attribute	Default value
fn bold	normal, non-bold	maxim text	no maximized text
fn underline	not underlined	marked	unmarked
fn italic	non-italic	multiple choice	single choice
vertical	horizontal	no drop	drop
closed	open	no edit	editable
notaccessible	accessible	no scrolling tools	with scrolling tools
modal	non-modal	no format	formatable
no border	with border		
locator	no locator	template	no "template" store
separated	not separated	autostore	no "autostore"
multiline	single line	no sound	with sound
no consistency	with consistency	sorted	not sorted
maximizable	not maximizable	negative	positive
retransmission rule B	retransmission rule A	direct in	no direct input
content associated	independent positioning	report index	report string
direct navigation	not allowed	non controlled mode	controlled mode

**Table 45: Integer attributes defaults**

Attribute	Default value	Attribute	Default value
fn family	SWISS	input type	0
fn height	10	echo type	0
fn colour	0 (black)	max char	1
bits per units (pixel)	1	group	1
bits per units (component)	8	default item	1
		list index	1 for the first, all other are numbered consecutively
message type	0	max line	terminal dependent
destroy event	1	sound format	0
colour entry	0	initial value	min value
request type	0	increment	1
defactive	1	max time	infinite
title font	default font	input timeout	infinite
colour	0 (black)	alignment	0, left aligned
bitmap disptype	stretched	multimedia content type	0, HTML
initial font	default font		
label font	default font		

**Table 46: NDC attributes defaults**

Attribute	Default value
xpos, ypos of the message box	terminal dependent
xpos	0.0
ypos	0.0

**Table 47: Width and height defaults**

Element	Width default value	Height default value
application bar	terminal dependent	terminal dependent
button: button bar	0,125	0,03
pop-up menu	terminal dependent	
dialogue box	1,0	0,75
separator	dialogue box width	dialogue box height
frame	dialogue box width	dialogue box height
text area	dialogue box width	dialogue box height
graphic output area	dialogue box width	dialogue box height
text input field	dialogue box width	0,03
push button	0,125	0,03
check box	0,125	0,03
radio button	0,125	0,03
list box	dialogue box width	dialogue box height
combination box	dialogue box width	dialogue box height
sensitive area	0,125	0,03
message box	terminal dependent	terminal dependent

**Table 48: String attributes defaults**

Attribute	Default value
echo char	2/13

## **10 Introduction of the VEMMI service into existing Videotex ETSS**

### **10.1 Introduction of the VEMMI to ETS 300 072**

The VEMMI Protocol Elements are mapped on the Videotex Presentation Data Elements (VPDEs).

In order to enable the data flow between Videotex terminal and VEMMI application, the logical channel shall be transparent for VEMMI data blocks.

### **10.2 Introduction of the VEMMI to ETS 300 223 and ETS 300 079**

The VEMMI Protocol Elements are mapped on one or more SBV\_VTX\_Data service element.

In order to enable the data flow between Videotex terminal and VEMMI application, the logical channel shall be transparent for VEMMI data blocks.

NOTE: ETS 300 079 [23] corresponds to ITU-T Recommendation T.102 [5] and ETS 300 223 [26] corresponds to ITU-T Recommendation T.105 [6].

## **Annex A (normative): CCITT Recommendation T.51 [2] String**

### **A.1 Scope**

This annex describes the rules to be applied when encoding names, strings, etc. using CCITT Recommendation T.51 [2]. The CCITT Recommendation T.51 [2] String defined by this annex serves as a reference model by restricting CCITT Recommendation T.51 [2] to those elements of the code extension mechanisms, of the character sets and repertoire which are necessary to ease the implementations.

This definition is not specific to an individual telematic service, but it can be referenced by telematic application standards.

### **A.2 Graphic character sets**

The primary set of graphic characters (see figure A1) is identical with the set of graphic characters of the International Reference Version (IRV) of the 7-bit coded character set of CCITT Recommendation T.50 [1].




				b7	0	0	0	0	1	1	1	1
				b6	0	0	1	1	0	0	1	1
				b5	0	1	0	1	0	1	0	1
					0	1	2	3	4	5	6	7
b4	b3	b2	b1									
0	0	0	0	0				0	@	P	`	p
0	0	0	1	1			!	1	A	Q	a	q
0	0	1	0	2			"	2	B	R	b	r
0	0	1	1	3			#	3	C	S	c	s
0	1	0	0	4			\$	4	D	T	d	t
0	1	0	1	5			%	5	E	U	e	u
0	1	1	0	6			&	6	F	V	f	v
0	1	1	1	7			'	7	G	W	g	w
1	0	0	0	8			(	8	H	X	h	x
1	0	0	1	9			)	9	I	Y	i	y
1	0	1	0	10			*	:	J	Z	j	z
1	0	1	1	11			+	;	K	[	k	{
1	1	0	0	12			,	<	L	\	l	
1	1	0	1	13			-	=	M	]	m	}
1	1	1	0	14			.	>	N	^	n	~
1	1	1	1	15			/	?	O	_	o	

**Figure A.1: Primary set of graphic characters for CCITT-Recommendation T.51 String**

The supplementary set of graphic characters is specified in figure A.2.

				b7	0	0	0	0	1	1	1	1
				b6	0	0	1	1	0	0	1	1
				b5	0	1	0	1	0	1	0	1
					0	1	2	3	4	5	6	7
b4	b3	b2	b1									
0	0	0	0	0			NBSP	°		—	Ω	K
0	0	0	1	1			ı	±	˘	¹	Æ	æ
0	0	1	0	2			ø	²	´	®	Ð	ð
0	0	1	1	3			£	³	^	©	ª	õ
0	1	0	0	4				x	~	™	℥	℥
0	1	0	1	5			¥	μ	-	♪		ı
0	1	1	0	6				¶	˘	¬	IJ	ij
0	1	1	1	7			§	·	▪	ı	Ł	ł
1	0	0	0	8			¤	÷	▪▪		ƒ	ƒ
1	0	0	1	9			¢	‚			Ø	ø
1	0	1	0	10			“	”	°		Œ	œ
1	0	1	1	11			«	»	¸		ŉ	ŉ
1	1	0	0	12			←	¼		⅛	Ɔ	Ɔ
1	1	0	1	13			↑	½	”	⅜	ƒ	t
1	1	1	0	14			→	¾	ı	⅝	ŋ	ŋ
1	1	1	1	15			↓	¿	ˆ	⅞	'n	SHY

 These codepositions shall not be used

**Figure A.2: Supplementary set of graphic characters for CCITT Recommendation T.51 String**

For the CCITT Recommendation T.51 [2] String the following applies:

- the primary set is designated as the G0 set and invoked in columns 2 to 7 of the code table;
- the supplementary set is designated as a G2 set. In the 8 bit environment the set is invoked in column 10 to 15;
- in the CCITT-Recommendation T.51 [2] String no designation and invocation sequences are allowed;
- all characters in column 4 of the supplementary set are non-spacing characters (diacritical marks);
- unallocated code positions are reserved and shall not be used;
- The compatibility provisions described in CCITT Recommendation T.51 [2] subclause 2.2.4 notes 3, 4, 6 are not applied.

### **A.3 Code extension technique**

In the 8 bit environment no code extension sequence is allowed. In the 7 bit environment the single shift function SS2 (code 1/9) is used to invoke one character from the G2 set. All other shift functions are not allowed.

### **A.4 Repertoire of the latin-based character set**

The CCITT Recommendation T.51 [2] String repertoire is identical with the superset of the repertoire of the Latin based character set specified in CCITT Recommendation T.51 [2] annex A. All combinations of diacritical marks with basic letters as specified in figure A2 of CCITT Recommendation T.51 [2] are supported.

The notes and remarks from CCITT Recommendation T.51 [2], annex A subclause A.4 shall not apply.

### **A.5 Control functions**

Invocation and designation sequences for control functions shall not be permitted. From columns 0 and 1 of the code table in use, only the control characters CR, Carriage Return (0/13) and LF, Line Feed (0/10) can be used. They are specified in CCITT Recommendation T.50 [1].

**Annex B (normative): Mandatory subset of ISO 8859**

	2	3	4	5	6	7		10	11	12	13	14	15
0	SP	0	@	P	`	p							
1	!	1	A	Q	a	q							
2	"	2	B	R	b	r							
3	#	3	C	S	c	s							
4	\$	4	D	T	d	t							
5	%	5	E	U	e	u							
6	&	6	F	V	f	v							
7	'	7	G	W	g	w							
8	(	8	H	X	h	x							
9	)	9	I	Y	i	y							
10	*	:	J	Z	j	z							
11	+	;	K	[	k	{							
12	,	<	L	\	l								
13	-	=	M	]	m	}							
14	.	>	N	^	n	~							
15	/	?	O	_	o								

Figure B.1: Mandatory subset of ISO 8859 [13] code table

## Annex C (normative): Minimum datatype kernel

A VEMMI terminal shall support the following minimum datatype kernel:

**Table C.1: Supported datatypes**

Class	Encoding standard	Remarks
For text data		
	VEMMI high quality text	
	ISO 8859 [13] (fully formed accented characters) from code position 2/0 to code position 7/15 as given in annex B	For latin based terminal
	Shift JIS Code (for Japanese characters)	For Japanese terminals
For still picture		
	ISO/IEC 10918 [16] (JPEG)	
	VEMMI device independent bitmap	
	Graphics Interchange Format (sm) Version 89a. CompuServe Incorporated Columbus, Ohio, USA (only the functions of Version 87 shall be supported)	
	Microsoft Windows device-independent bitmap (DIB) with RLE4, RLE8 and 1, 4, 8 or 24 bits per pixel	
For audio data		
	WAVE format	see note
	MIDI format	see note
NOTE: Mandatory in this case denotes that a terminal shall support all VEMMI services related to the functionality. It does not mean that a terminal shall be equipped with the necessary device to play this data.		

## History

Document history	
April 1996	UAP 46: 1996-04-08 to 1996-08-30
September 1996	First Edition