



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

**ETS 300 382**

February 1995

---

Source: ETSI TC-TE

Reference: DE/TE-01016

ICS: 33.020, 33.040.40

**Key words:** Videotex, Man Machine Interface, VEMMI

**Terminal Equipment (TE);  
Videotex Enhanced Man Machine Interface service (VEMMI)**

**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 1995. All rights reserved.



## Contents

Foreword .....	9
1 Scope .....	11
2 Normative references .....	11
3 Definitions and abbreviations .....	12
3.1 Definitions .....	12
3.2 Abbreviations .....	13
4 General model .....	14
4.1 Introduction .....	14
4.2 Definition of the VEMMI elements .....	14
4.2.1 VEMMI object definition .....	14
4.2.2 VEMMI component definition .....	15
4.2.3 VEMMI component item definition .....	15
4.3 VEMMI logical plane structure model .....	15
4.3.1 The standard Videotex logical plane .....	16
4.3.2 The VEMMI objects logical plane .....	16
4.4 Operation modes for VEMMI terminals .....	16
4.4.1 The standard Videotex mode .....	16
4.4.2 The VEMMI mode .....	16
4.4.3 Switching between standard Videotex mode and VEMMI mode .....	17
4.5 VEMMI elements data content .....	17
4.5.1 Text data definition .....	17
4.5.2 Videotex data definition .....	18
4.6 VEMMI objects positioning and dimensioning .....	18
4.6.1 Positioning .....	18
4.6.2 Dimensioning .....	19
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 .....	24
4.7.2.1 Definition of component states .....	24
4.7.2.2 Definition of component state parameters .....	25
4.8 Local action management .....	25
4.9 Storage considerations .....	26
4.10 Common rules for object handling .....	27
4.10.1 Active state and focus management .....	27
4.10.2 Behaviour of the modal mode .....	27
4.10.3 Size considerations and clipping .....	27
5 Service definition .....	28
5.1 Service elements initiated by the VEMMI application .....	29
5.1.1 VEMMI_On .....	29
5.1.2 VEMMI_Off .....	30
5.1.3 VEMMI_Create_Object .....	30
5.1.4 VEMMI_Open_Object .....	30
5.1.5 VEMMI_Close_Object .....	31
5.1.6 VEMMI_Close_All .....	31
5.1.7 VEMMI_Destroy_Object .....	31
5.1.8 VEMMI_Obj_Access_Disable .....	32
5.1.9 VEMMI_Obj_Access_Enable .....	32
5.1.10 VEMMI_Additional_Data .....	32

5.1.11	VEMMI_Modify_Component.....	33
5.1.12	VEMMI_Obj_Location_Change .....	33
5.1.13	VEMMI_User_Lock.....	33
5.1.14	VEMMI_User_Unlock .....	34
5.1.15	VEMMI_Reset.....	34
5.2	Service elements initiated by the terminal.....	34
5.2.1	VEMMI_Object_Retransmission.....	34
5.2.2	VEMMI_More_Data .....	35
5.2.3	VEMMI_User_Data.....	35
5.2.4	VEMMI_Error .....	36
6	VEMMI objects introduction.....	37
6.1	The Application Bar .....	37
6.1.1	Composition.....	37
6.2	The Button Bar .....	37
6.2.1	Composition.....	37
6.3	The Pop-Up Menu.....	38
6.3.1	Composition.....	38
6.4	The Dialogue Box.....	38
6.4.1	Composition.....	38
6.4.1.1	The Separator component .....	38
6.4.1.2	The Frame component .....	39
6.4.1.3	The Text Presentation Area component.....	39
6.4.1.4	The Videotex Presentation Area component .....	39
6.4.1.5	The Push Button component .....	39
6.4.1.6	The Text Input Field component.....	39
6.4.1.7	The Check Box component .....	39
6.4.1.8	The Radio Button component .....	39
6.4.1.9	The List Box component .....	39
6.4.1.10	The Combination Box component.....	39
6.4.1.11	The Sensitive Area component.....	40
6.4.1.12	The Locator component .....	40
6.5	The Presentation Box.....	40
6.5.1	Composition.....	40
6.5.1.1	The Text-Videotex Output Field component .....	40
6.5.1.2	The Push Button component .....	40
6.5.1.3	The Text Input Area .....	40
6.5.1.4	The Sensitive Area component.....	40
6.5.1.5	The Locator component.....	41
6.6	The Message Box .....	41
7	Functional description.....	41
7.1	General rules for the behaviour of elements.....	41
7.1.1	User Interaction .....	41
7.1.2	Local actions and reports.....	41
7.1.3	Relationship between objects and components .....	42
7.1.4	VEMMI elements with audio data content .....	42
7.2	The Application Bar .....	42
7.2.1	Composition.....	46
7.2.1.1	Menu Choice Bar components.....	46
7.2.1.2	Menu Choice Pull-Down components.....	48
7.2.1.3	Menu Choice Cascading components .....	50
7.2.1.4	Menu Choice Separator components .....	51
7.3	The Button Bar .....	52
7.3.1	Composition.....	54
7.3.1.1	The Button component.....	54
7.4	The Pop-Up Menu.....	55
7.4.1	Composition.....	57
7.4.1.1	Menu Choice Pop-Up components.....	57
7.4.1.2	Menu Choice Cascading components .....	59
7.4.1.3	Menu Choice Separator components .....	59
7.5	The Dialogue Box.....	60

7.5.1	Composition .....	62
7.5.1.1	The Separator component.....	62
7.5.1.2	The Frame component .....	64
7.5.1.3	The Text Presentation Area component.....	65
7.5.1.4	The Videotex Presentation Area component.....	66
7.5.1.5	The Push Button component.....	67
7.5.1.6	The Text Input Field component.....	68
7.5.1.7	The Check Box component.....	70
7.5.1.8	The Radio Button component.....	71
7.5.1.9	The List Box component.....	73
7.5.1.10	The Combination Box component .....	75
7.5.1.11	The Sensitive Area component .....	78
7.5.1.12	The Locator component.....	80
7.6	The Presentation Box .....	81
7.6.1	Composition .....	84
7.6.1.1	The Text-Videotex Output Field component.....	84
7.6.1.2	The Push Button component.....	88
7.6.1.3	The Text Input Area component.....	88
7.6.1.4	The Sensitive Area component .....	90
7.6.1.5	The Locator component.....	91
7.7	The Message Box.....	91
8	Coding of the service elements.....	94
8.1	Overall switching of coding environment .....	94
8.2	Switching into the VEMMI mode.....	96
8.3	ISO/IEC 9281 syntax structure .....	96
8.4	Coding of the Picture Data Entity (PDE).....	97
8.5	Object specific commands.....	98
8.6	General commands .....	98
8.7	Terminal commands .....	99
8.8	Error Message .....	99
8.9	Coding of the VEMMI command fields .....	100
8.9.1	Command Code .....	100
8.9.2	Object Identification Number (OIN).....	101
8.9.3	More Data Indicator (MDI).....	101
8.9.4	User data.....	101
9	Coding of the VEMMI data .....	102
9.1	Structure of the VEMMI data of a VEMMI_Create_Object command .....	102
9.2	Structure of the VEMMI data of a VEMMI_Modify_Component command .....	103
9.3	Structure of the VEMMI data of a VEMMI_Additional_Data command .....	104
9.4	Structure of the VEMMI data of a VEMMI_Obj_Location_Change command.....	105
9.5	Structure of the VEMMI data of a VEMMI_More_Data command.....	105
9.6	Structure of the VEMMI data of a VEMMI_User_Data command .....	106
9.6.1	Coding of the report values for VEMMI components .....	106
9.6.1.1	Menu Choice Bar, Menu Choice Pull-Down, Menu Choice Cascading, Menu Choice Pop-Up, Push Button, Button Sensitive Area.....	106
9.6.1.2	Text Input Field, List Box, Combination Box.....	107
9.6.1.3	Check Box, Radio Button .....	107
9.6.1.3.1	Locator.....	107
9.7	Structure of the VEMMI data of a VEMMI_Error command.....	108
9.7.1	Coding of the error message.....	108
9.8	General rules for coding VEMMI data.....	109
9.9	Code assignments for VEMMI objects and components.....	113
9.10	Coding of VEMMI elements .....	113
9.10.1	Application Bar .....	113
9.10.1.1	Menu Choice Bar, Menu Choice Pull-Down, Menu Choice Cascading, Menu Choice Pop-Up components.....	114
9.10.1.2	Menu Choice Separator component.....	115
9.10.2	Button Bar .....	115
9.10.2.1	Button .....	116

9.10.3	Pop-Up Menu.....	117
9.10.3.1	Menu Choice Pop-Up, Menu Choice Cascading components .....	118
9.10.4	Dialogue Box .....	118
9.10.4.1	Separator .....	119
9.10.4.2	Frame.....	120
9.10.4.3	Text Presentation Area .....	121
9.10.4.4	Videotex Presentation Area .....	122
9.10.4.5	Push Button.....	123
9.10.4.6	Text Input Field .....	124
9.10.4.7	Check Box.....	126
9.10.4.8	Radio Button .....	127
9.10.4.9	List Box .....	128
9.10.4.10	Combination Box.....	129
9.10.4.11	Sensitive Area .....	131
9.10.4.12	Locator .....	132
9.10.5	Presentation Box .....	133
9.10.5.1	Text-Videotex Output Field .....	135
9.10.5.2	Push Button.....	135
9.10.5.3	Text Input Area .....	136
9.10.5.4	Sensitive Area .....	137
9.10.5.5	Locator .....	138
9.10.6	Message Box .....	139
9.10.7	Coding of local actions.....	141
9.11	Attribute field type codes .....	143
10	Introduction of the VEMMI service into existing Videotex ETSs.....	144
10.1	Introduction of the VEMMI to ETS 300 072.....	144
10.2	Introduction of the VEMMI to ETS 300 223 and ETS 300 079.....	144
Annex A (normative):	T.51String .....	145
A.1	Introduction .....	145
A.2	Graphic character sets .....	145
A.3	Code extension technique .....	147
A.4	Repertoire of the latin based character set.....	147
A.5	Control functions.....	147
Annex B (informative):	Future VEMMI concepts .....	147
B.1	Local object storage.....	147
B.1.1	VEMMI_OpenApplication .....	148
B.1.2	VEMMI_OpenApplicationResponse .....	148
B.1.3	VEMMI_StoreObjects.....	148
B.1.4	VEMMI_StoreObjectsResponse .....	149
B.2	Operative objects.....	149
B.2.1	VEMMI_ExecuteProgram.....	149
B.3	Colour table .....	149
B.3.1	VEMMI_LoadColTable .....	150
B.3.2	VEMMI_ResetColTable.....	150
B.4	Set of Objects concept .....	150
B.4.1	VEMMI_CreateSetofObjects .....	151
B.5	Coding .....	151

B.6 Provisional command codes ..... 151  
History..... 152

Blank page



## Foreword

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

Annex A is normative to this ETS while annex B is informative.

<b>Proposed transposition dates</b>	
Date of latest announcement of this ETS (doa):	31 May 1995
Date of latest publication of new National Standard or endorsement of this ETS (dop/e):	30 November 1995
Date of withdrawal of any conflicting National Standard (dow):	30 November 1995

Blank page

## 1 Scope

This ETS specifies the data syntax to be used by Videotex services for implementation of the Videotex Enhanced Man Machine Interface (VEMMI).

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).

This ETS also applies to any equipment (e.g. another Videotex service) which acts as a Videotex terminal.

## 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 last edition of the publication referred to applies.

- [1] ETS 300 072: "Terminal Equipment (TE); Videotex presentation layer protocol, Videotex presentation layer data syntax".
- [2] ETS 300 073: "Videotex presentation layer protocol; Geometric Display (CEPT Recommendation T/TE 06-02, Edinburgh 1988)".
- [3] ETS 300 076 (1992): "Terminal Equipment (TE); Videotex, Terminal Facility Identifier (TFI)".
- [4] ETS 300 079: "Integrated Services Digital Network (ISDN); Syntax-based Videotex, End-to-end protocols, circuit mode DTE-DTE".
- [5] ETS 300 149: "Terminal Equipment (TE); Videotex, Audio syntax".
- [6] ETS 300 177: "Terminal Equipment (TE); Videotex, Photographic syntax".
- [7] ETS 300 223: "Terminal Equipment (TE); Syntax-based Videotex, Common end-to-end protocols".
- [8] ITU-T Recommendation T.50: "International Reference Alphabet (IRA) (Formerly International Alphabet No.5 or IA5) - Information technology - 7 bit coded character set for information interchange".
- [9] ITU-T Recommendation T.51: "Latin based coded character sets for telematic services".
- [10] ITU-T Recommendation T.101 (1993): "International interworking for videotex services".
- [11] ITU-T Recommendation F.300: "Videotex service".
- [12] ISO/IEC 9281 (1990): "Information technology - Picture coding methods".
- [13] ITU-T Recommendation T.52: "Non-latin coded character sets for telematic services".
- [14] ISO 2022 (1986): "Information Processing - ISO 7-bit and 8-bit coded character sets - Code extension techniques".

- [15] ISO 2375 (1991): "Data Processing - Procedure for registration of escape sequences".
- [16] ISO 10918-1: "Digital compression and coding of continuous-tone still images - Part 1: Requirements and guidelines".

### 3 Definitions and abbreviations

#### 3.1 Definitions

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

**anticipation:** Optional facility for a VEMMI application to send closed objects or objects with closed components to a VEMMI terminal which supports this option.

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

**Defined Display Area (DDA):** The rectangular part of the screen that can be used by the Videotex service [ITU-T Recommendation F.300 [11]].

**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 emphasise 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 in a Dialogue Box and a Button in an active Button Bar.

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

**standard Videotex application:** Videotex application using encoded data, protocols and profiles, as defined in the Videotex ETSs referenced in clause 2. A standard Videotex application does not use a VEMMI service, data and protocols.

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

**validation:** User activation action followed by a confirmation of the choice with a keyboard or with a pointing device.

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

**Videotex application:** Part of a Videotex service which is under the responsibility of only one application provider. The Videotex service provider may also act as an application provider [ITU-T Recommendation F.300 [11]].

**Videotex Host Computer:** The computer (or network of computers provided by a single party) on which one or more applications are implemented and/or one or more other Videotex service facilities are provided [ITU-T Recommendation F.300 [11]].

**Videotex terminal:** The equipment by means of which the user interacts with the Videotex service. A typical Videotex terminal includes:

- 1) a numeric keypad and/or alphanumeric keyboard and/or other graphical input devices;
- 2) a visual display unit or a suitably modified television receiver;
- 3) electronic processing and storage devices required to interface these components to the telecommunications network and to generate the display.

### 3.2 Abbreviations

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

CIN	Component Identification Number
CMI	Coding Method Identifier
CR	Carriage Return
DDA	Defined Display Area
DRCS	Dynamically Redefinable Character Set
DS I	Data Syntax according to ITU-T Recommendation T.101 [10], annex B
DS II	Data Syntax according to ITU-T Recommendation T.101 [10], annex C
DS III	Data Syntax according to ITU-T Recommendation T.101 [10], annex D
ESC	Escape
IRV	International Reference Version
ISDN	Integrated Services Digital Network
LF	Line Feed
LI	Length indicator
MDI	More Data Indicator
NDC	Normalised Device Co-ordinate
OIN	Object Identification Number
PCD	Picture Coding Delimiter
PCE	Picture Control Entity
PDE	Picture Data Entity
PE	Picture Entity
PI	Picture Identifier
PM	Picture Mode
PSPDN	Packet Switched Public Data Network
PSTN	Public Switched Telephone Network
SBV	Syntax-Based Videotex
TE	Terminal Equipment
TFI	Terminal Facility Identifier
VPDE	Videotex Presentation Data Element
VEMMI	Videotex Enhanced Man Machine Interface
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;
- text and Videotex encoded data contents as described in the Videotex ETSs referenced in clause 2.

A VEMMI terminal shall also handle a standard 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 or objects;
- VEMMI components or components;
- VEMMI component item 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.

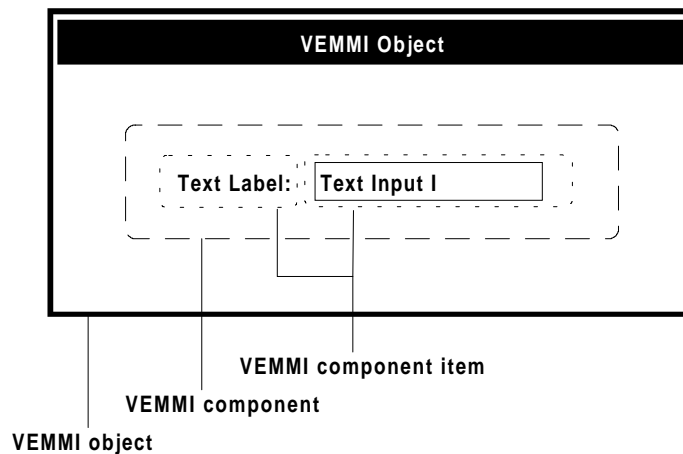


Figure 1: Example showing objects/components/items

#### 4.2.1 VEMMI object definition

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

VEMMI objects are 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 "parent object".

In order to transport information, components may carry a data content which can be text or Videotex data (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, and of their text content, is terminal dependent. The representation of Videotex content is defined in the corresponding Videotex ETSS referenced in clause 2.

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

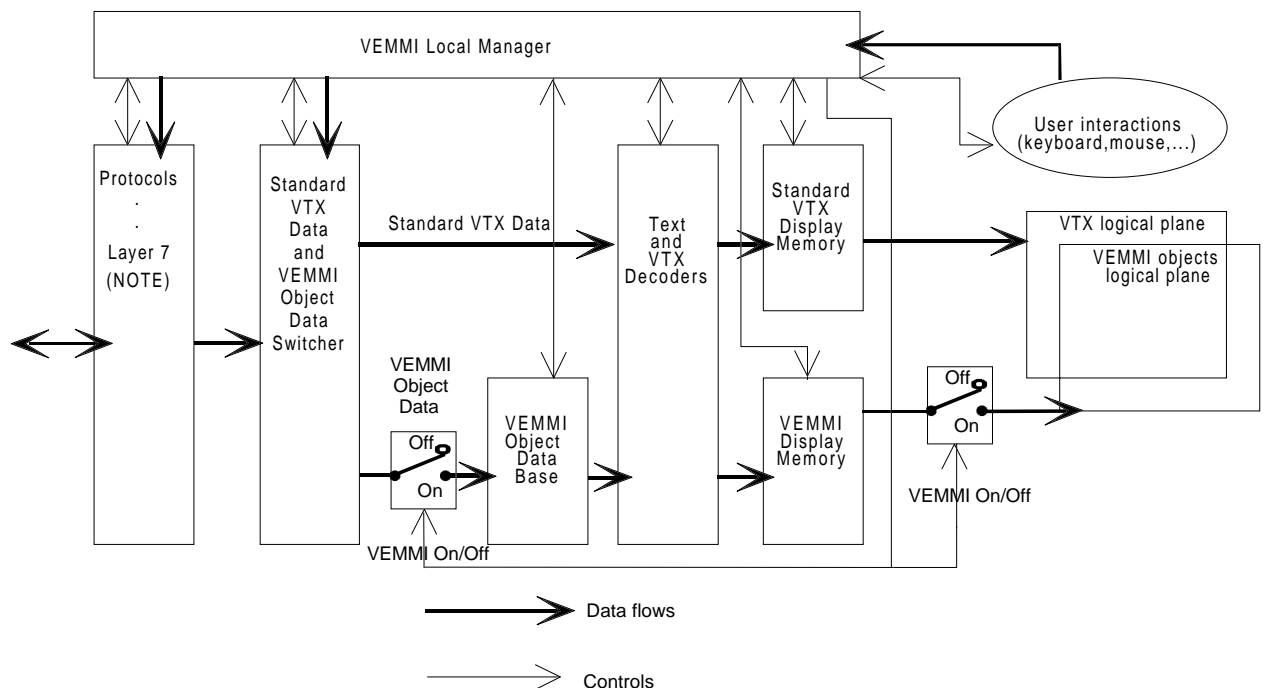
#### 4.2.3 VEMMI component item definition

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.

#### 4.3 VEMMI logical plane structure model

A VEMMI logical plane structure model is used to handle VEMMI objects and their contents, as well as standard Videotex data. The VEMMI logical plane structure model is described below and is presented in figure 2, with an example of a possible VEMMI terminal structure.

A VEMMI terminal shall implement the behaviour of this VEMMI logical plane structure model. However, no assumption is made on the real physical plane structure of the terminal and how the terminal implements that plane structure model. The possible VEMMI terminal structure drawing as shown in figure 2 is for explanatory purposes only.



NOTE: Standard Videotex 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 VEMMI logical plane structure model is based on two logical planes:

- a standard Videotex logical plane;
- a VEMMI objects logical plane.

These two logical planes are fully superimposed (same DDA dimensions and positions).

#### **4.3.1 The standard Videotex logical plane**

This logical plane receives standard Videotex data in a background plane. Standard Videotex data are received from standard Videotex applications or from VEMMI applications using standard Videotex data. This standard Videotex logical plane 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 logical plane**

This logical plane receives VEMMI objects described in this ETS, and their text and Videotex encoded data contents. This VEMMI objects logical plane is a foreground plane, with the highest visual priority over the standard Videotex logical plane. When empty, this plane is fully transparent. When not empty, DDA areas used by VEMMI objects hide the corresponding parts of the standard Videotex logical plane. Full transparency for this VEMMI objects logical plane is also possible within a VEMMI application through a VEMMI command. The VEMMI objects logical plane shall be able to handle object overlapping and restoring mechanisms as referenced in subclauses 4.6 and 4.11.1. Within this plane, and for a given VEMMI element Videotex content, the Videotex data rules and priorities defined in the Videotex ETSs referenced in clause 2 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 Videotex specific functions (disconnect, etc..).

#### **4.4.1 The standard Videotex mode**

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

In standard Videotex mode, the VEMMI terminal displays standard Videotex data in the standard Videotex logical plane. Standard Videotex data are fully visible; the VEMMI object logical plane is transparent. User inputs are not controlled by the VEMMI local manager.

VEMMI On/Off switches are set to the Off position. The terminal shall ignore all VEMMI commands except VEMMI\_On, VEMMI\_Create\_Object and VEMMI\_Open\_Object. These last two commands shall perform, in addition to their normal behaviour, an implicit switch to the VEMMI mode; VEMMI On/Off switches are then switched to the On position. A VEMMI\_Open\_Object referring to a non-existing object shall only induce an object request but shall not switch the terminal to the VEMMI operation mode.

#### **4.4.2 The VEMMI mode**

In the VEMMI mode, a VEMMI terminal can receive standard Videotex data and VEMMI objects data in two parallel paths, as shown in figure 2. The VEMMI On/Off switches are set to the On position.

VEMMI object data are displayed in the VEMMI objects logical plane. Standard Videotex data are displayed in the Standard Videotex logical plane. VEMMI object displayed in the VEMMI objects logical plane hide the corresponding parts of the standard Videotex logical plane. The display order for objects and for components within objects, shall correspond to their transmission order (last received, last displayed).

User inputs are controlled by the VEMMI local manager.

The display priority rules between the two planes, as defined in subclause 4.3, apply.



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

A standard Videotex Application shall be performed on a VEMMI terminal without any specific command, only if the application is compatible with the capabilities of the terminal identified with its Terminal Facility Identifier (TFI). A VEMMI terminal shall always be set in the standard Videotex mode, or Videotex mode, when powered on, or when physically reset.

From the standard Videotex mode, a VEMMI terminal shall be switched to the VEMMI operation mode, setting the logical VEMMI On/Off switches to the On position, if one of the following VEMMI commands is received:

- VEMMI\_On;
- VEMMI\_Create\_Object;
- VEMMI\_Open\_Object referring to an existing object in the terminal.

If a VEMMI\_Open\_Object referring to a non-existing object is received, the VEMMI terminal shall request it from the VEMMI application without switching to the VEMMI mode.

When switched to the VEMMI operation mode the VEMMI terminal shall display standard Videotex data in the Videotex logical background plane. Additionally, it shall display in the VEMMI objects logical plane:

- VEMMI Open objects still present in the Object data base before the VEMMI operation mode switching and coming from a previous VEMMI operation mode;
- VEMMI data received since this VEMMI operation mode switching.

Switching to the VEMMI operation mode or switching back does not reset:

- any data within the terminal standard Videotex data path;
- standard Videotex data reception and handling continues;
- any data within the terminal VEMMI object data base.

When a VEMMI\_Off command is received, the VEMMI terminal shall switch in the standard Videotex operation mode and shall display only the standard Videotex logical plane. The VEMMI On/Off switches are set by the VEMMI\_Off command to the Off position; the VEMMI objects logical plane then becomes fully transparent. The VEMMI\_Off command does not reset any data within the VEMMI object data base. VEMMI object reception and handling in the VEMMI data base then stops, except for the VEMMI\_Reset command and the three commands to switch back to the VEMMI operation mode.

#### 4.5 VEMMI elements data content

For the time being VEMMI elements are able to carry two different types of data content:

- a) text data;
- b) Videotex data.

For future versions of VEMMI some other types of data content may be available (bitmaps, fonts, etc.).

In table 61 (see subclause 9.11, "Attribute field type codes") the command attribute codes 5/7-6/4 are reserved for the definition of new types of data contents.

##### 4.5.1 Text data definition

Data encoded in accordance with the rules given in annex A (normative). The list of characters of ITU-T Recommendation T.52 [13] which can be used to encode text data is for further study.

The representation of text data is terminal dependent. The font and emphasised presentation of text data are also terminal dependent (underlining, reverse video, different colour). The maximum size of a text character shall be equal to the size of a Videotex character in the selected mode (40/80 columns).

To provide the "line feed" functionality within the text content of a component or a component item the CR+LF control characters given in ITU-T Recommendation T.50 [8] shall be used.

#### 4.5.2 Videotex data definition

Data encoded in accordance with ETS 300 072 [1], ETS 300 073 [2], ETS 300 076 [3], ETS 300 149 [5] and ETS 300 177 [6].

VEMMI can be used with 80 character mode making use of the appropriate SWITCHING SEQUENCE or the DEFINE FORMAT VPDE to switch to the desired Data Syntax. 80 character mode is not mandatory for a VEMMI terminal.

#### 4.6 VEMMI objects positioning and dimensioning

##### 4.6.1 Positioning

The standard Videotex logical plane shall support the co-ordinate system used for positioning the different types of Videotex encoded data (alphamosaic, Dynamically Redefinable Character Set (DRCS), geometric, photographic...) as defined in the relevant ETSs referenced in clause 2.

The VEMMI objects logical plane shall support a Normalised Device Co-ordinate (NDC) system for positioning VEMMI objects and components within the DDA (see figure 3). The normalised 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.

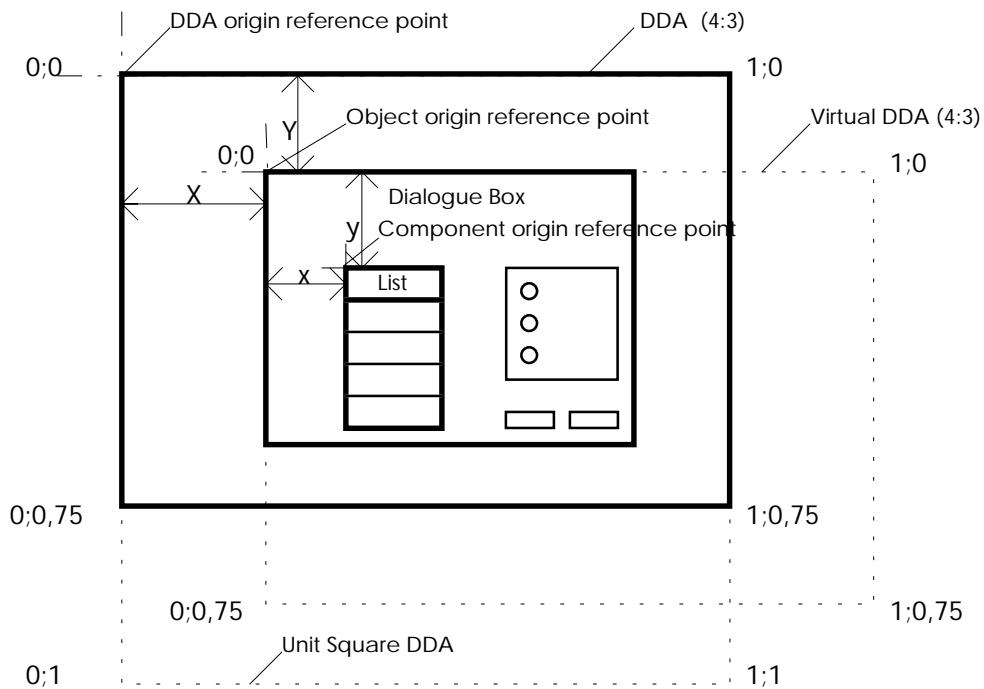
For typical Videotex display, using common 4:3 aspect ratio, the horizontal positioning is in the range 0 to 1 corresponding to the whole width of the DDA, the vertical positioning is in the range 0 to 0,75 corresponding to the whole height of the DDA.

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 origin reference point. Consequently, there is a virtual DDA for each VEMMI object for positioning its own components.

For objects with a border requested by the VEMMI application, the object reference point, to express the position of the object in the DDA and the positions of the components within the object, shall be equal to the object reference point of the same object without a border. The border is only a simple frame with no impact on the positioning of the object or its components.

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

For VEMMI terminals only able to handle Videotex data contents on a character basis (alphamosaic, DRCS, Photographic Profile 1), VEMMI objects and components can be physically positioned 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 VEMMI application using only this type of Videotex data contents.



**Figure 3: VEMMI positioning NDC space**

#### 4.6.2 Dimensioning

The standard Videotex logical plane shall support the co-ordinate system used for dimensioning the different types of Videotex encoded data (alphamosaic, DRCS, geometric, photographic...) as defined in the relevant ETSS referenced in clause 2.

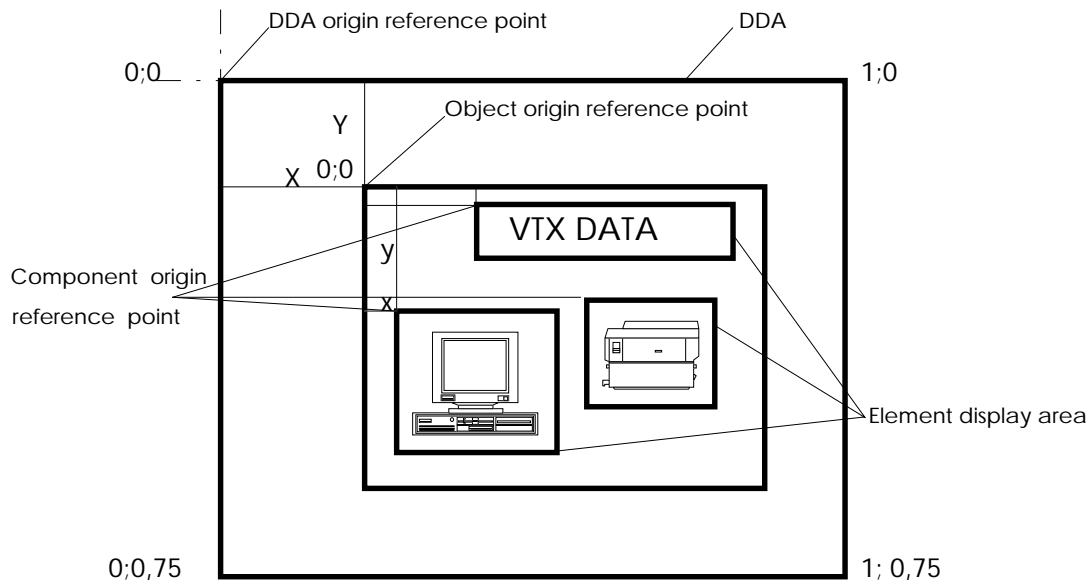
The VEMMI objects logical plane and VEMMI objects and components shall support a dimensioning system based on alphamosaic character positions. By default, the full DDA is supposed to support 24 rows of 40 character positions, as defined in ETS 300 072 [1] (annex A, Parts 0 and 1).

When a different number of character positions, or of rows, for the DDA is requested by the VEMMI application, the "Define Format" Videotex Presentation Data Element (VPDE) specified in ETS 300 072 [1] (annex A, Part 6) shall be used in accordance with the VEMMI terminal capabilities expressed in its TFI.

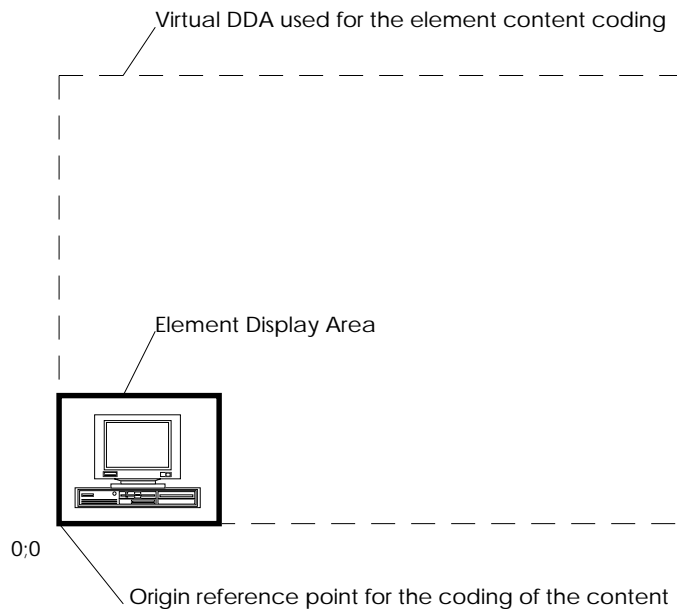
The width and the height of a VEMMI object, or of a VEMMI component, expressed in character positions, are referred with respect to their positioning reference point (upper left-hand corner). This reference point becomes the standard Videotex home position (character position 1/1).

In a VEMMI element, a Videotex content follows the general rules for the coding of its relevant ETS referenced in clause 2, including a possible picture placement or positioning.

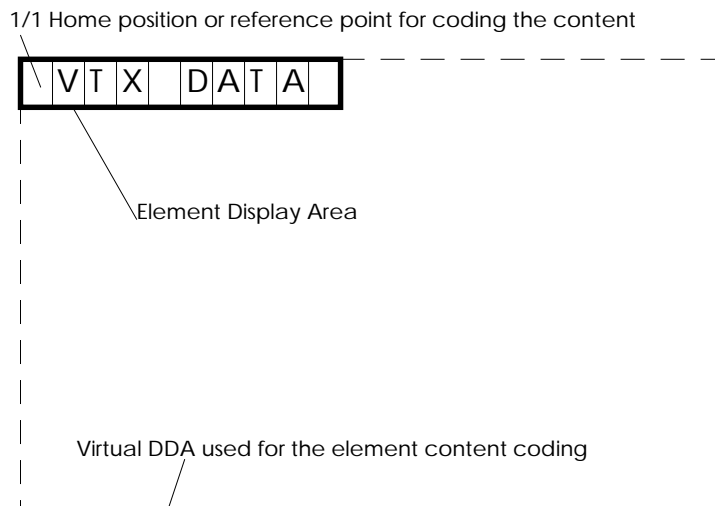
However, the origin of each content is no longer referred to the standard DDA reference point (0;0) used for Videotex data using a NDC system, or to the standard character home position (1/1) for Videotex data based on character positions. The origin is now the reference point, or the home position, of a virtual DDA mapped on the corresponding corner of the small display area, or element display area, which results of the dimensions of the VEMMI element (see figures 4, 5 and 6).



**Figure 4: VEMMI positioning system**



**Figure 5: Example for of a Videotex content using NDC positioning**



**Figure 6: Example for of a Videotex content using character positioning**

For objects with a border requested by the VEMMI application, the border shall be included in the dimensions defined by the VEMMI application.

#### **Other recommendations**

However, constraints for decoding, for positioning and dimensioning lead to the following recommendations:

- the number of rows and character positions for alphamosaic display;
- the DRCS parameters;
- the Videotex profile;
- the Geometric profile;
- the Videotex photographic syntax profile;

used by a VEMMI application, should be the same in the two logical planes used for standard Videotex data and VEMMI objects.

When the Videotex data content is greater than the space allocated for it (a part of the content is outside of the dimensions of the element display area), the terminal may clip the data by the overall dimensions of the element display area.

### **4.7 VEMMI elements states and state parameters**

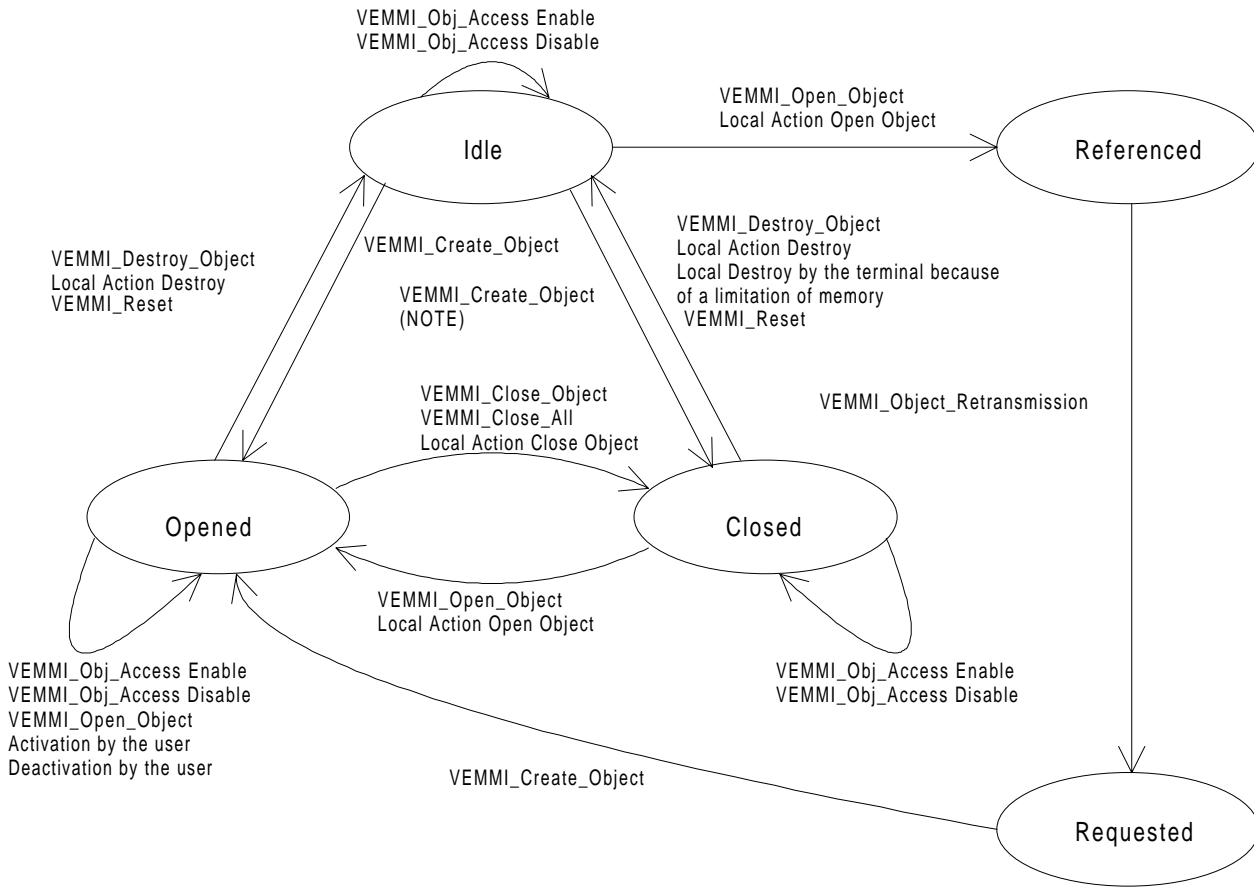
#### **4.7.1 Object**

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.

4.7.1.1 Definition of object states

Figure 7 shows the state diagram for objects as viewed by the terminal:



NOTE: Objects can be created in the closed state only if the optional anticipation facility is supported.

Figure 7: 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.
  - 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.
  - Closing an object is initiated by the VEMMI application, either as initial state at the time of the object creation (if the optional anticipation facility is supported), 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 on the terminal was referenced by an 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.

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 limitation of memory). The VEMMI application is not informed about this local destroy. 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 shall not be possible for the user to interact with it.  
  
The active state management is described in subclause 4.11.1.
  
- 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.

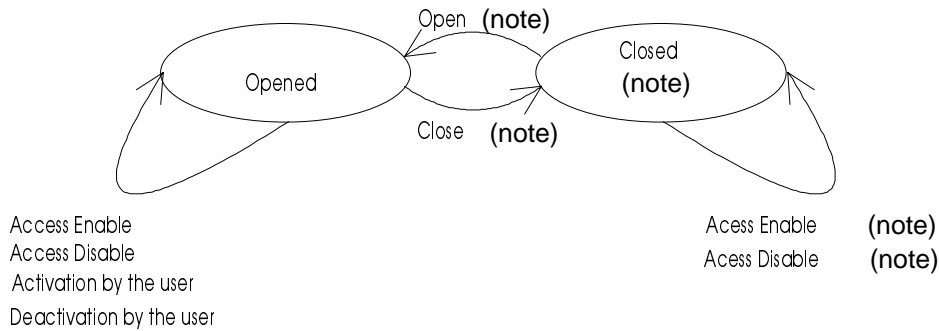
## 4.7.2 Component

### 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 8 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. There shall be no closed state for components, if the optional anticipation facility is not supported.



NOTE: There shall be no closed state for components if the optional anticipation facility is not supported.

**Figure 8: State diagram for components**

In general, the following logical states for components are defined, with certain restrictions 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 but is not displayed.

Closing a component is initiated by the VEMMI application, either as initial state at the time of the object creation (if the optional anticipation facility is supported), 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.



#### 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 has currently the input focus. The user inputs always refer to the active component within the active object. If the active component is inaccessible it is not possible for the user to interact with it.  
  
The active state management is described in subclause 4.11.1.
- 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 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.

#### 4.8 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, can be delayed, because the user inputs were sent to the VEMMI application first, and then the application can react with a VEMMI command to change the state. This procedure may 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", one "Report command" and one "General command". The following commands are defined:

a) Element specific commands:

- open component of the parent object (Parameter: CIN) see statement below;
- close component of the parent object (Parameter: CIN) see statement below;
- open object (Parameter: OIN);
- close object (Parameter: OIN);
- change component state parameter of a component in the parent object to inaccessible (Parameter: CIN);
- change component state parameter of a component in the parent object to accessible (Parameter: CIN);
- destroy object (Parameter: OIN).

There shall be no closed state for components if the optional anticipation facility is not supported.

b) Report commands:

- report CIN of the component;
- report the current values of the component;
- report the values of all components in the parent object.

c) General commands:

- user lock;
- restore initial values of all input components in the parent object. This command should only be applied if the object was created with the "StoreInitialValue" attribute set true.

A list of "Element specific commands", one "Report command" and one "General command" can be used to specify one local action for a component. No specific command 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. When a list of local actions is executed each action shall be completed before the execution of the next action in the list.

#### 4.9 Storage considerations

VEMMI terminals should have storage capabilities correctly dimensioned to provide a satisfactory VEMMI service, in close relationship with their profile and their Videotex 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 optional and terminal dependent, the VEMMI application is never informed.

However, the VEMMI terminal shall send a VEMMI\_Error (Out of memory) message to the VEMMI application when it does not have possibility to release enough memory to execute a VEMMI command or a local action. In that case the terminal may ignore further VEMMI commands, as long as the VEMMI application has not released a significant part of storage 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 storage 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:

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

The last opened VEMMI object shall be set active by the VEMMI local manager up 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 state parameter "active" 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, the 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 active, the VEMMI object and/or one of its components shall receive its focus from the local manager. The active state parameter of the VEMMI object and/or of the VEMMI components shall be clearly, visually indicated to the user.

Within the active VEMMI object, not more than one VEMMI component shall be active and have the input 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. The other times the manner of giving the focus on activation of an object to one of its component 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 same VEMMI application is forbidden by the terminal, any tentative attempt is 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 are executed.

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

The text or Videotex 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 may decide to clip non-consistent data content to the overall dimensions of the container.

## 5 Service definition

This clause describes the service characteristics offered by the VEMMI.

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

All VEMMI service elements, or VEMMI commands, shall only be issued on an established network connection. Before sending the first VEMMI command the VEMMI application should issue a Terminal Facility Identifier (TFI) request to ensure that the Videotex 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. Regarding the execution of the VEMMI service elements, VEMMI terminals can be divided into two different classes:

a) terminals which support object anticipation:

these terminals shall execute all VEMMI service elements including their attributes.

b) terminals which do not support object anticipation:

these terminals shall execute all VEMMI service elements including their attributes except:

- 1) A VEMMI\_Create\_Object service element which contains any Opened attribute carrying the value false. Such an attempt to create an object or one of its components in the closed state shall cause the VEMMI error "Service not supported".
- 2) A VEMMI\_Modify\_Component service element which contains any Opened attribute carrying the value false. Such an attempt to change the state of one component to closed shall cause the VEMMI error "Service not supported".

The VEMMI services are divided into two groups:

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

Table 1: VEMMI services

VEMMI Services	Initiated Appl/Term	Function
VEMMI_On	Application	switch to VEMMI mode
VEMMI_Off	Application	switch to standard Videotex mode
VEMMI_Create_Object	Application	object definition
VEMMI_Open_Object	Application	display 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_Additional_Data	Application	more data belonging to an object
VEMMI_Modify_Component	Application	modify components characteristics
VEMMI_Obj_Location_Change	Application	defines a new object position
VEMMI_User_Lock	Application	user inputs are not permitted
VEMMI_User_Unlock	Application	user inputs are permitted
VEMMI_Reset	Application	remove all the existing objects
VEMMI_Object_Retransmission	Terminal	request for an object retransmission
VEMMI_More_Data	Terminal	request for more data blocks
VEMMI_User_Data	Terminal	user data in VEMMI mode
VEMMI_Error	Terminal	error condition or situation
Abbreviations: Appl = Application; Term = Terminal.		

There is no explicit confirmation in any 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.

All local actions, except "Open object", which refer to a non-existing object shall be ignored by the VEMMI terminal. If a local action "Open object" refers to a non-existing object the terminal shall request its retransmission.

## 5.1 Service elements initiated by the VEMMI application

### 5.1.1 VEMMI\_On

This service element can be used to switch the VEMMI terminal to the VEMMI operation mode. The switching mechanism is defined in subclause 4.4.3. In the VEMMI operation mode, the mechanism described in subclause 4.4.2 shall apply. The primitives and parameters of the VEMMI\_On service are described below.

Table 2

Parameters	Request	Indication
None		

**5.1.2 VEMMI\_Off**

This service element shall be used to switch the VEMMI terminal to the standard Videotex mode. The switching mechanism is defined in subclause 4.4.3. In standard Videotex operation mode, the mechanism described in subclause 4.4.1 shall apply. The primitives and parameters of the VEMMI\_Off service are described below.

**Table 3**

Parameters	Request	Indication
None		

**5.1.3 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 store 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 destruction of the previously defined 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.

The primitives and parameters of the VEMMI\_Create\_Object service are described below.

**Table 4**

Parameters	Request	Indication
Object Identification Number	Mandatory	Mandatory(=)
Object Description coded according to subclause 9.1	Mandatory	Mandatory(=)
(=): the value of this parameter is identical to the one of the corresponding parameter in the preceding Request primitive.		

**5.1.4 VEMMI\_Open\_Object**

This service element shall be used to set an object to the opened state in the VEMMI terminal. The terminal shall display the specified object immediately. If the object is not stored in the terminal, it shall request the object by sending a VEMMI\_Object\_Retransmission Request to the VEMMI application with the specified OIN. 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 this object.

The primitives and parameters of the VEMMI\_Open\_Object service are described below.

**Table 5**

Parameters	Request	Indication
Object Identification Number	Mandatory	Mandatory(=)
(=): the value of this parameter is identical to the one of the corresponding parameter in the preceding Request primitive.		

### 5.1.5 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 its 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.

The primitives and parameters of the VEMMI\_Close\_Object service are given in table 6.

**Table 6**

Parameters	Request	Indication
Object Identification Number	Mandatory	Mandatory(=)
(=): the value of this parameter is identical to the one of the corresponding parameter in the preceding Request primitive.		

### 5.1.6 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 operation mode of the terminal. The primitives and parameters of the VEMMI\_Close\_All service are given in table 7.

**Table 7**

Parameters	Request	Indication
None		

### 5.1.7 VEMMI\_Destroy\_Object

This service element shall be used to remove a VEMMI object from the object data base in the VEMMI terminal. 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 primitives and parameters of the VEMMI\_Destroy\_Object service are given in table 8.

**Table 8**

Parameters	Request	Indication
Object Identification Number	Mandatory	Mandatory(=)
(=): the value of this parameter is identical to the one of the corresponding parameter in the preceding Request primitive.		

### 5.1.8 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 shall 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.

The primitives and parameters of the VEMMI\_Obj\_Access\_Disable service are given in table 9.

Table 9

Parameters	Request	Indication
Object Identification Number	Mandatory	Mandatory(=)
(=): the value of this parameter is identical to the one of the corresponding parameter in the preceding Request primitive.		

### 5.1.9 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.

The primitives and parameters of the VEMMI\_Obj\_Access\_Enable service are shown in table 10.

Table 10

Parameters	Request	Indication
Object Identification Number	Mandatory	Mandatory(=)
(=): the value of this parameter is identical to the one of the corresponding parameter in the preceding Request primitive.		

### 5.1.10 VEMMI\_Additional\_Data

This service element shall be used to send a data block of a previously created Presentation Box to the VEMMI terminal.

A VEMMI application shall use this service only after the reception of a VEMMI\_More\_Data indication.

The primitives and parameters of the VEMMI\_Additional\_Data service are shown in table 11.

Table 11

Parameters	Request	Indication
Object Identification Number	Mandatory	Mandatory(=)
Object Description coded according subclause 9.3.	Mandatory	Mandatory(=)
(=): the value of this parameter is identical to the one of the corresponding parameter in the preceding Request primitive.		



**5.1.11 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 9.10.

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.

The primitives and parameters of the VEMMI\_Modify\_Component service are given in table 12.

**Table 12**

Parameters	Request	Indication
Object Identification Number	Mandatory	Mandatory(=)
Modification description coded according to subclause 9.2.	Mandatory	Mandatory
(=): the value of this parameter is identical to the one of the corresponding parameter in the preceding Request primitive.		

**5.1.12 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.

The primitives and parameters of the VEMMI\_Obj\_Location\_Change service are shown in table 13.

**Table 13**

Parameters	Request	Indication
Object Identification Number	Mandatory	Mandatory(=)
New Location coded according to subclause 9.4	Mandatory	Mandatory(=)
(=): the value of this parameter is identical to the one of the corresponding parameter in the preceding Request primitive.		

**5.1.13 VEMMI\_User\_Lock**

This service element shall be used to restrict any user input in the terminal until a VEMMI\_User\_Unlock is sent. The primitives and parameters of the VEMMI\_User\_Lock service are given in table 14.

**Table 14**

Parameters	Request	Indication
None		

#### 5.1.14 VEMMI\_User\_Unlock

This service element shall be used to permit user inputs in the terminal. The primitives and parameters of the VEMMI\_User\_Unlock service are shown in table 15.

Table 15

Parameters	Request	Indication
None		

#### 5.1.15 VEMMI\_Reset

This service element shall be used to destroy all objects stored in the objects data base of the VEMMI terminal. All opened objects shall be cleared from the terminal screen. A VEMMI\_Reset shall not change the current operation mode of the terminal. The VEMMI\_Reset command shall be executed even if the terminal is waiting for the retransmission of a requested object or data block. The primitives and parameters of the VEMMI\_Reset service are shown in table 16.

Table 16

Parameters	Request	Indication
None		

### 5.2 Service elements initiated by the terminal

#### 5.2.1 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 or a local action "Open object" referring to an object that does not exist on the terminal.

If a VEMMI\_Open\_Object command referring to a non-existing object is received, or a local action "Open object" referring to a non-existing object is executed the terminal shall:

- suspend any execution of VEMMI commands or local actions, except the VEMMI\_Reset command;
- lock the user;
- send a VEMMI\_Object\_Retransmission request;
- memorise all VEMMI commands received until it receives a VEMMI\_Create\_Object with the same OIN or a VEMMI\_Reset command.

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.

After the reception of a VEMMI\_Reset, the terminal shall:

- cancel the execution of the possibly suspended local action;
- cancel the execution of the possibly memorised VEMMI commands;
- execute the VEMMI\_Reset command;
- unlock the user.

The primitives and parameters of the VEMMI\_Object\_Retransmission service are shown in table 17.

**Table 17**

Parameters	Request	Indication
Object Identification Number	Mandatory	Mandatory(=)
(=): the value of this parameter is identical to the one of the corresponding parameter in the preceding Request primitive.		

### 5.2.2 VEMMI\_More\_Data

This service element shall be used to request to the VEMMI application one data block belonging to an existing Presentation Box.

When the terminal issues one command VEMMI\_More\_Data after a user interaction on scrolling tools, the terminal shall:

- suspend any execution of VEMMI commands or local actions, except the VEMMI\_Reset command;
- lock the user;
- memorise all VEMMI commands received until it receives a VEMMI\_Additional\_Data of the requested data block or a VEMMI\_Reset command.

After the reception of the requested data block, the terminal shall:

- resume the execution of the possibly memorised VEMMI commands received in the order of their reception.

After the reception of a VEMMI\_Reset, the terminal shall:

- cancel the execution of the possibly memorised VEMMI commands;
- execute the VEMMI\_Reset command;
- unlock the user.

The primitives and parameters of the VEMMI\_More\_Data service are shown in table 18.

**Table 18**

Parameters	Request	Indication
Object Identification Number	Mandatory	Mandatory(=)
Block Identification Number coded according to subclause 9.5	Mandatory	Mandatory(=)
(=): the value of this parameter is identical to the one of the corresponding parameter in the preceding Request primitive.		

### 5.2.3 VEMMI\_User\_Data

This service element shall be used to send user data corresponding to one object to the VEMMI application. The primitives and parameters of the VEMMI\_User\_Data service are shown in table 19.

**Table 19**

Parameters	Request	Indication
Object Identification Number	Mandatory	Mandatory(=)
User Data coded according to subclause 9.6	Mandatory	Mandatory(=)
(=): the value of this parameter is identical to the one of the corresponding parameter in the preceding Request primitive.		

#### 5.2.4 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.

The primitives and parameters of the VEMMI\_Error service are shown in table 20.

Table 20

Parameters	Request	Indication
Object Identification Number	Mandatory	Mandatory(=)
Command Code	Mandatory	Mandatory(=)
User Data coded according subclause 9.7	Mandatory	Mandatory(=)
(=): the value of this parameter is identical to the one of the corresponding parameter in the preceding Request primitive.		

**Command Code:** This parameter specifies the VEMMI command that caused the error. The one byte command codes are defined in subclause 8.9.1, table 27.

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.
- Service not supported:
  - a terminal which is not supporting anticipation received:
    - 1) a VEMMI\_Create\_Object service element which contains any Opened attribute carrying the value false;
    - 2) a VEMMI\_Modify\_Component service element which contains any Opened attribute carrying the value false.

## 6 VEMMI objects introduction

Six VEMMI objects are defined in this ETS. They offer a wide choice of dialogue elements needed by VEMMI applications.

The following VEMMI objects are defined:

- Application Bar;
- Button Bar;
- Pop-Up Menu;
- Dialogue Box;
- Presentation Box;
- Message Box.

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 with 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 functionalities. The three different groups are named:

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

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

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

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

The menu choice components can have either text or Videotex content.

A Menu Choice Separator component can be used to visually separate menu choice components in a logical group.

### 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, and with either text data or Videotex data content.

### 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 any where 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 functionalities. The two different groups are named:

- Primary Pop-Up Menu;
- Cascading Menu.

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

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

The menu choice components can have either text or Videotex content.

A Menu Choice Separator component can be used to visually separate menu choice components in a logical group.

### 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.

Four presentation components are defined:

- the Separator;
- the Frame;
- the Text Presentation Area;
- the Videotex Presentation Area.

Eight dialogue components are defined:

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

#### 6.4.1 Composition

##### 6.4.1.1 The Separator component

A Separator is a horizontal or vertical solid line. It's 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 goals of this component are:

- to present text information;
- to title or to label dialogue components;
- to present results from the VEMMI application execution;
- to offer choices on text contents using the Sensitive Area and the Locator.

#### **6.4.1.4 The Videotex Presentation Area component**

The Videotex Presentation Area is a rectangular area in which Videotex data is displayed. The goals of this component are:

- to present Videotex information;
- to title or to label dialogue components;
- to embellish the presentation;
- to offer choices on Videotex contents using the Sensitive Area and the Locator.

#### **6.4.1.5 The Push Button component**

The purpose of the Push Button is to trigger a local action to be immediately performed. The content of a Push Button is either text data or Videotex data.

#### **6.4.1.6 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.

#### **6.4.1.7 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 marked or unmarked independently of any other Check Boxes.

#### **6.4.1.8 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 marking of one Radio Button leads to the unmarking of the other Radio Buttons belonging to the same Radio Button group.

#### **6.4.1.9 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, so different controls are offered to scroll the list up and down.

#### **6.4.1.10 The Combination Box component**

The purpose of the Combination Box is to combine the functionality of a single choice List Box with the functionality of a Text Input Field. It contains a list of text items that 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.

#### **6.4.1.11 The Sensitive Area component**

The purpose of the Sensitive Area in the Dialogue Box is to offer a selection area associated to a Text or Videotex Presentation Area.

#### **6.4.1.12 The Locator component**

The purpose of the Locator is to permit the choice of a specific co-ordinate position in a Text or Videotex Presentation Area within a Dialogue Box.

### **6.5 The Presentation Box**

The Presentation Box is the main area of the DDA where the VEMMI application presents text or Videotex data to the user in a Text-Videotex Output Field.

As the space requirement for text and Videotex data presentation may be higher than the size of the Text-Videotex Output Field component, the Text-Videotex Output Field component offers, when necessary, vertical scrolling capabilities over the data. To satisfy the scrolling needs and user interactions, the Presentation Box includes service areas to present the possible scrolling tools and Push Buttons.

The following components are defined in the Presentation Box:

- Text-Videotex Output Field;
- Push Button;
- Text Input Area;
- Sensitive Area;
- Locator.

#### **6.5.1 Composition**

##### **6.5.1.1 The Text-Videotex Output Field component**

The Text-Videotex Output Field is the output area of the Presentation Box. In this area, text or Videotex data content is displayed to the user with possible scrolling capabilities. For a given Presentation Box this Output Field is either a Text Output Field or a Videotex Output Field. Text and Videotex data are never mixed in a Text-Videotex Output Field component. A Presentation Box always has one and only one Text-VTX Output Field, with a content possibly split in data blocks transmitted on terminal requests.

##### **6.5.1.2 The Push Button component**

The purpose of the Push Button is to associate some controls to the presentation of text and Videotex data in a Presentation Box (e.g. Help, Cancel, OK,..). The Push Button can have text or Videotex data content. Push Buttons are grouped in a button service area on the bottom of the Presentation Box just under the Text-Videotex Output Field component.

##### **6.5.1.3 The Text Input Area**

The purpose of the Text Input Area is to offer character input in direct relation with the text and Videotex output data in the Text-Videotex Output Field.

##### **6.5.1.4 The Sensitive Area component**

The purpose of the Sensitive Area is to permit the user selection of a rectangular area transparently overlapping the Text-Videotex Output Field.



### 6.5.1.5 The Locator component

The purpose of the Locator is to permit the choice of a specific co-ordinate position into the Text-Videotex Output Field.

### 6.6 The Message Box

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

## 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 CIN of the component;
- report the current values of the component;
- report the values of all components in this object.

Tables 21 and 22 show the possible trigger events and the possible reports for each component.

**Table 21: 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		
VTX Presentation Area		
Push Button	✓	✓
Text Input Field	✓	✓
Check Box	✓	✓
Radio Button	✓	✓
List Box	✓	✓
Combination Box	✓	✓
Sensitive Area	✓	✓
Locator	✓	✓
Text-VTX Output Field		
Text Input Area	✓	✓

Table 22: Reports

Component	CIN	Value of all Cmp	Value (note)
Menu Choice Bar	✓		
Menu Choice Pull-Down	✓		
Menu Choice Cascading	✓		
Menu Choice Pop-Up	✓		
Menu Choice Separator			
Button	✓		
Separator			
Frame			
Text Presentation Area			
VTX Presentation Area			
Push Button	✓	✓	
Text Input Field	✓	✓	string
Check Box	✓	✓	Boolean
Radio Button	✓	✓	Boolean
List Box	✓	✓	string list
Combination Box	✓	✓	string
Sensitive Area	✓	✓	
Locator	✓	✓	integer
Text-VTX Output Field			
Text Input Area	✓	✓	string
NOTE: The entries in this column specify the data type of the values that are reported.			
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 gets closed, the entire object including all its components is removed from the screen, regardless if 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 to inaccessible, the entire object including all its components becomes inaccessible, regardless if 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 VEMMI elements with audio data content

If the content of a VEMMI element is audio data, the sound shall be performed at the time the component is opened.

## 7.2 The Application Bar

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

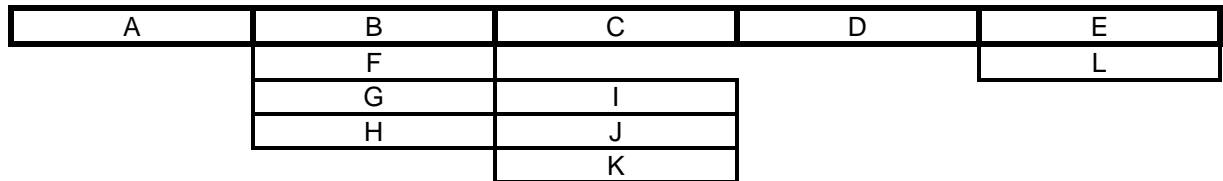
- Bar;
- Pull-Down Menu;
- Cascading Menu.

The Bar is a horizontal or vertical list of Menu Choice Bar components. The Pull-Down Menus are vertical lists of Menu Choice Pull-Down components which are associated to the same Menu Choice Bar component. The Cascading Menus are vertical lists of Menu Choice Cascading components which are associated to the same Menu Choice Pull-Down component.

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).

A Menu Choice Separator component can be used to visually separate menu choice components in a logical group.

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



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 of 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

**Interactive functionalities:**

- shifting.

General visual aspect (see figures 9 and 10):

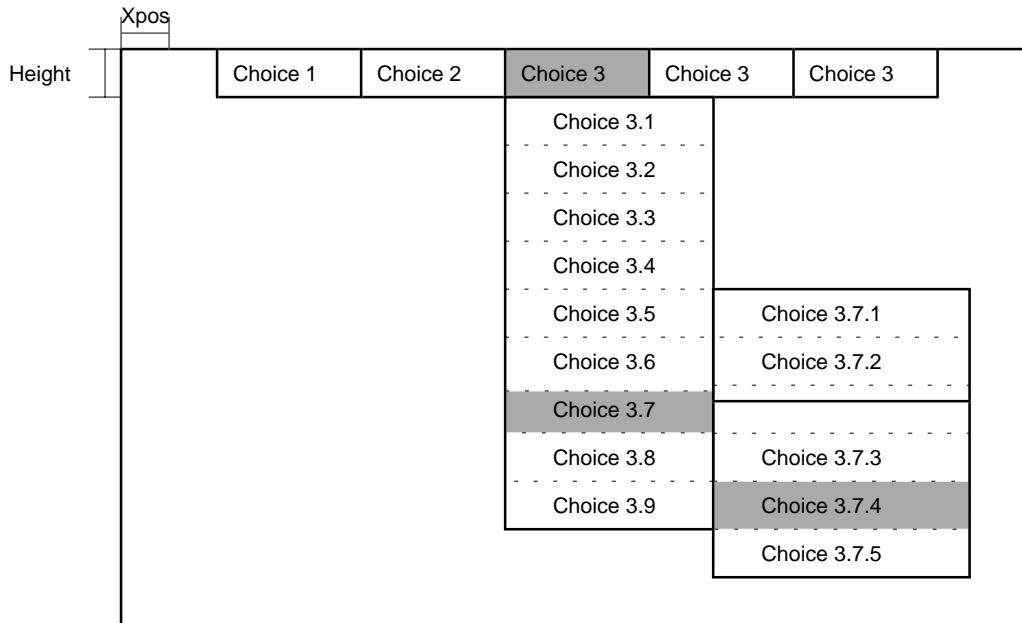


Figure 9

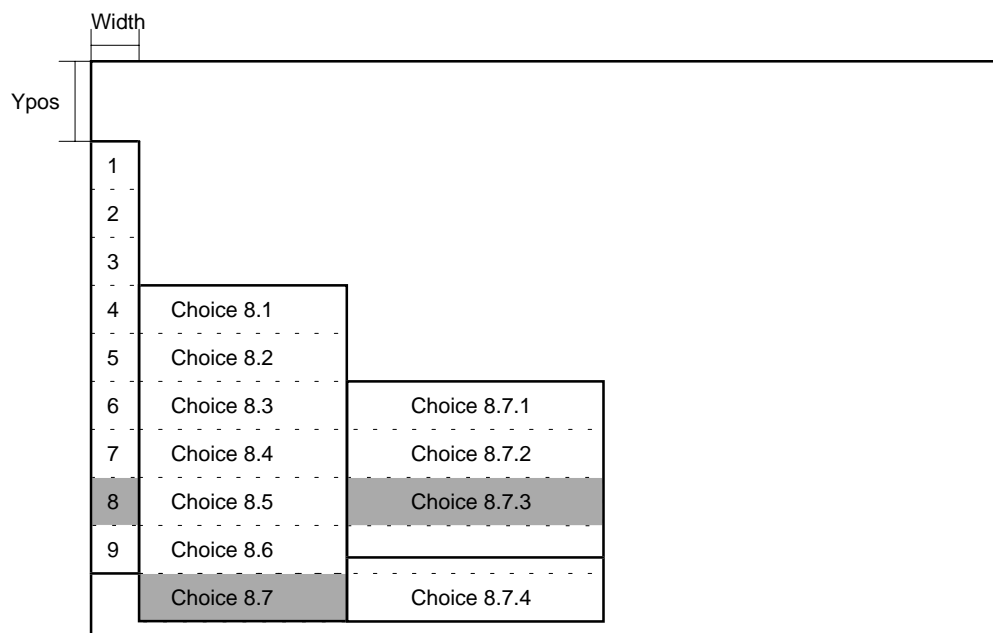


Figure 10

**Attributes:**

**Horizontal:** The values of this Boolean attribute carry following meanings:

true: horizontal presentation of the element;

false: vertical presentation of the element.

**XPos:** This attribute carries the horizontal position of the element in NDC. It is only applicable to horizontal Application Bars.

**YPos:** This attribute carries the vertical position of the element in NDC. It is only applicable to vertical Application Bars.

**Height:** This attribute carries the height of the Bar. It is only applicable to horizontal Application Bars.

**Width:** This attribute carries the width of the Bar. It is only applicable to vertical Application Bars.

**FirstActive:** This attribute carries the CIN of the Menu Choice which is active by default, the first time the object is opened.

**Opened:** The values of this Boolean attribute carry following meanings:

true: the element shall be in the opened state;

false: the element shall be in the closed state.

**Accessible:** The values of this Boolean attribute carry following meanings:

true: the object can be accessed;

false: the object cannot be accessed.

## 7.2.1 Composition

### 7.2.1.1 Menu Choice Bar components

**Description:** The Bar is a consecutive list of Menu Choice Bar components positioned side by side horizontally, or vertically. All Menu Choice Bar components shall have the same type of data content (either text data or Videotex data).

Horizontal representation:

- the Application Bar shall be presented at the top of the DDA;
- the number of rows allocated for each Menu Choice Bar component shall be the same.

Vertical representation:

- the Application Bar shall be presented at the leftmost part of the DDA;
- the number of columns allocated for each Menu Choice Bar component shall be the same.

A Menu Choice Separator component can be used to visually separate Menu Choice Bar components.

**Behaviour:** When the Application Bar is active, the terminal shall emphasise one Menu Choice Bar component, and offer shifting and validation facilities to the user. Menu Choice Bar components may be validated by a specified mnemonic key which should be visually indicated. The key is defined by the VEMMI application and transmitted within the Menu Choice Bar component description.

If a Menu Choice Bar component is validated, if existing, the associated local action shall be performed. Then, if existing, the associated Pull-Down Menu shall be opened (in that case the Menu Choice Bar component remains emphasised).

#### Interactive functionalities:

- activation;
- validation.

#### Visual aspect (see figures 11 and 12):

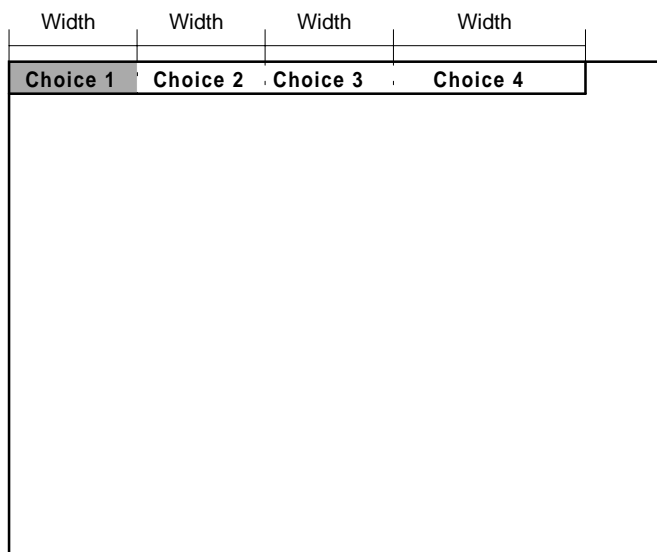
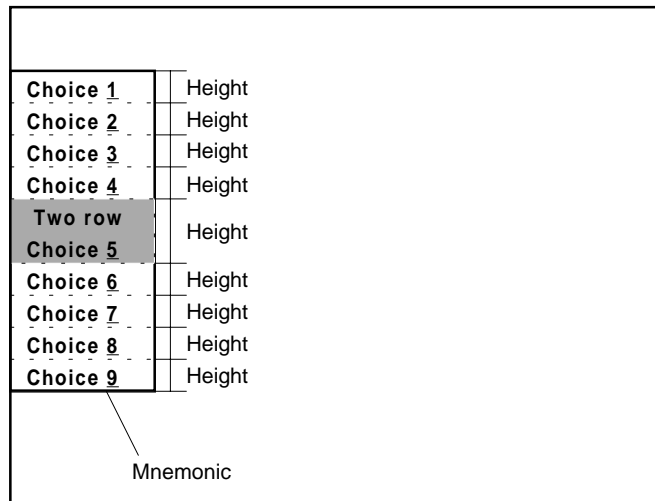


Figure 11: Horizontal Menu Bar



**Figure 12: Vertical Menu Bar**

**Attributes:**

- CIN:** This attribute carries the Component Identification Number (CIN).
- Height:** This attribute carries the height of the component. It is only applicable to vertical Bars.
- Width:** This attribute carries the width of the component. It is only applicable to horizontal Bars.
- Accessible:** The values of this Boolean attribute carry following meanings:  
 true: the element can be accessed;  
 false: the element cannot be accessed.
- Text:** This attribute carries the text content of the component.
- Mnemonic:** This attribute carries the character that may be entered by the user to validate the component if the Bar is active.
- VTX:** This attribute carries the Videotex content of a component.
- LocActAct:** This attribute carries the code for the local action which is associated to the component and is triggered by its activation.
- LocActVal:** This attribute carries the code for the local action which is associated to the component and is triggered by its validation.

### 7.2.1.2 Menu Choice Pull-Down components

**Description:** The Pull-Down Menu is a consecutive list of Menu Choice Pull-Down components associated to the same Menu Choice Bar component and presented vertically on several rows.

The number of columns allocated for each Menu Choice Pull-Down component shall be the same.

The Pull-Down Menu shall be positioned next to the associated Menu Choice Bar component.

A Menu Choice Separator component can be used to visually separate Menu Choice Pull-Down components.

All Menu Choice Pull-Down components shall have the same type of data content (either text data or Videotex data).

**Behaviour:** When the Pull-Down Menu is active, the terminal shall emphasise one Menu Choice Pull-Down component, and offer to the user shifting and validation facilities. Menu Choice Pull-Down components may be validated by a specified mnemonic key which should be visually indicated. The key is defined by the VEMMI application and transmitted within the Menu Choice Pull-Down component description.

If a Menu Choice Pull-Down component is validated, if existing, the associated local action shall be performed. Then, if existing, the associated Cascading Menu shall be opened (in that case the Menu Choice Pull-Down component remains emphasised). If no Cascading Menu is associated to the Menu Choice Pull-Down component, after a user validation the associated local action shall be performed, then the Pull-Down Menu shall be closed and the Menu Choice Bar component shall be de-emphasised.

#### Interactive functionalities:

- activation;
- validation.

#### Visual aspect (see figures 13 and 14):

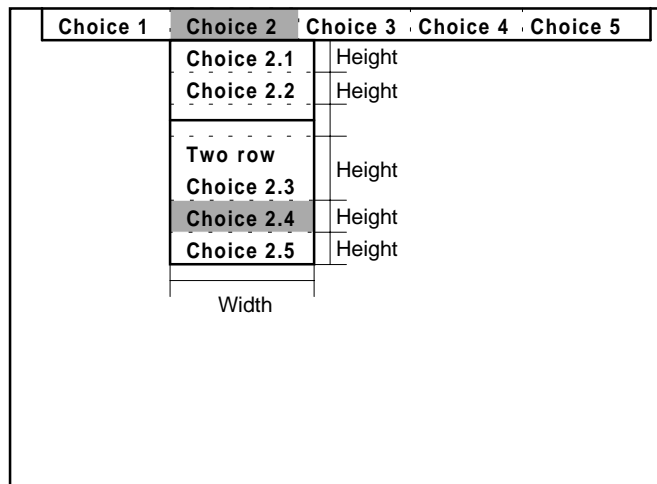
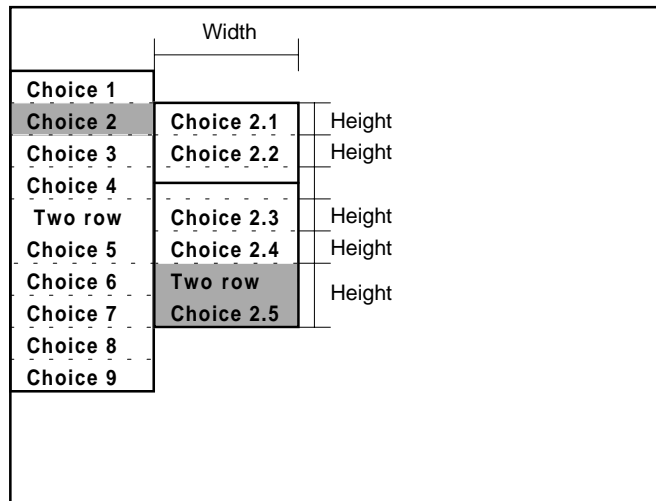


Figure 13: Pull-Down Menu





**Figure 14: Pull-Down Menu**

**Attributes:**

**CIN:** This attribute carries the Component Identification Number.

**Height:** This attribute carries the height of the component.

**Width:** This attribute carries the width of the component. It shall be equal to all Menu Choice Pull-Down components of one Pull-Down Menu.

**Accessible:** The values of this Boolean attribute carry following meanings:

true: the element can be accessed;

false: the element cannot be accessed.

**Text:** This attribute carries the text content of the component.

**Mnemonic:** This attribute carries the character that may be entered by the user to validate the component if the Pull-Down Menu is active.

**VTX:** This attribute carries the Videotex content of a component.

**LocActAct:** This attribute carries the code for the local action which is associated to the component and is triggered by its activation.

**LocActVal:** This attribute carries the code for the local action which is associated to the component and is triggered by its validation.

**7.2.1.3 Menu Choice Cascading components**

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

**Description:** The Cascading Menu is a consecutive list of Menu Choice Cascading components associated to the same Menu Choice Pull-Down component of the Pull-Down Menu or Pop-Up Menu and presented vertically on several rows.

The number of columns allocated for each Menu Choice Cascading component shall be the same.

The Cascading Menu shall be positioned next to the associated Pull-Down Menu or Pop-Up Menu.

A Menu Choice Separator component can be used to visually separate Menu Choice Cascading components.

All Menu Choice Cascading components shall have the same type of data content (either text data or Videotex data).

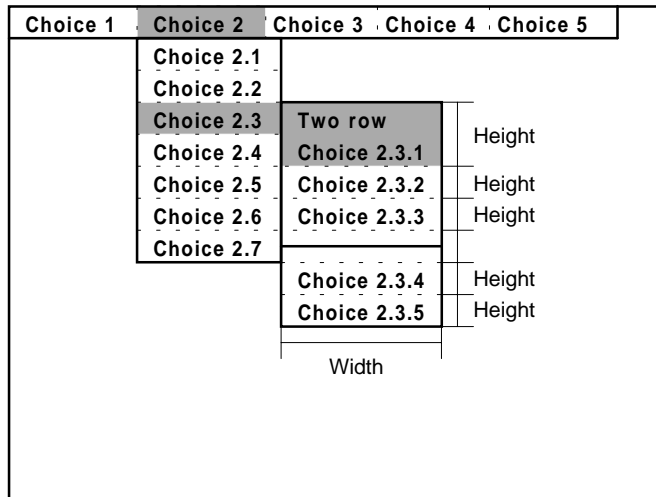
**Behaviour:** When the Cascading Menu is active, the terminal shall emphasise one Menu Choice Cascading component, and offer to the user shifting and validation facilities. Menu Choice Cascading components may be validated by a specified mnemonic key which should be visually indicated. The key is defined by the VEMMI application and transmitted within the Menu Choice Cascading component description.

After the validation of a Menu Choice Cascading component, if existing, the associated local action shall be performed, then the Cascading Menu and the Pull-Down Menu (Pop-Up Menu) shall be closed and the Menu Choice Bar component shall be de-emphasised.

**Interactive functionalities:**

- activation;
- validation.

**Visual aspect (see figure 15):**



**Figure 15: Cascading Menu**

**Attributes:**

<b>CIN:</b>	This attribute carries the Component Identification Number.
<b>Height:</b>	This attribute carries the height of the component.
<b>Width:</b>	This attribute carries the width of the component. It shall be equal to all Menu Choice Cascading components of one Cascading Menu.
<b>Accessible:</b>	The values of this Boolean attribute carry following meanings:  true: the element can be accessed;  false: the element cannot be accessed.
<b>Text:</b>	This attribute carries the text content of the component.
<b>Mnemonic:</b>	This attribute carries the character that may be entered by the user to validate the component if the Cascading Menu is active.
<b>VTX:</b>	This attribute carries the Videotex content of a component.
<b>LocActAct:</b>	This attribute carries the code for the local action which is associated to the component and is triggered by its activation.
<b>LocActVal:</b>	This attribute carries the code for the local action which is associated to the component and is triggered by its validation.

**7.2.1.4 Menu Choice Separator components**

A Menu Choice Separator component can be used to visually separate menu choice components in the Application Bar or the Pop-Up Menu.

**Description:** A Menu Choice Separator is a line of one character. This line is vertical or horizontal with a width/height equal to all components of the logical group.

In a horizontal Bar, the Separator has a height equal to all components of the logical group and the maximum space reserved for its width shall be one character position.

In a vertical Bar, a Pull-Down Menu, a Cascading Menu and a Primary Pop-Up Menu the Separator has a width equal to all components of the logical group and the maximum space reserved for its height shall be one character position.

**Behaviour:** A Menu Choice Separator is not accessible by the user.

**Interactive functionalities:**

- none.

**Visual aspect:** (see figure 15, subclause 7.2.1.3).

**Attributes:**

<b>CIN:</b>	This attribute carries the Component Identification Number.
-------------	---

### 7.3 The Button Bar

**Description:** The Button Bar is a consecutive list of Buttons positioned side by side horizontally, or vertically. All Buttons of the Button Bar shall have the same type of data content (either text data or Videotex data).

Horizontal representation:

- the number of rows allocated for each Button shall be the same.

Vertical representation:

- the number of columns allocated for each Button shall be the same.

**Behaviour:** When the Button Bar is active, the terminal shall emphasise one Button, and offer to the user shifting and validation facilities.

**Interactive functionalities:**

- shifting;
- moving.

**Visual aspect (see figure 16):**

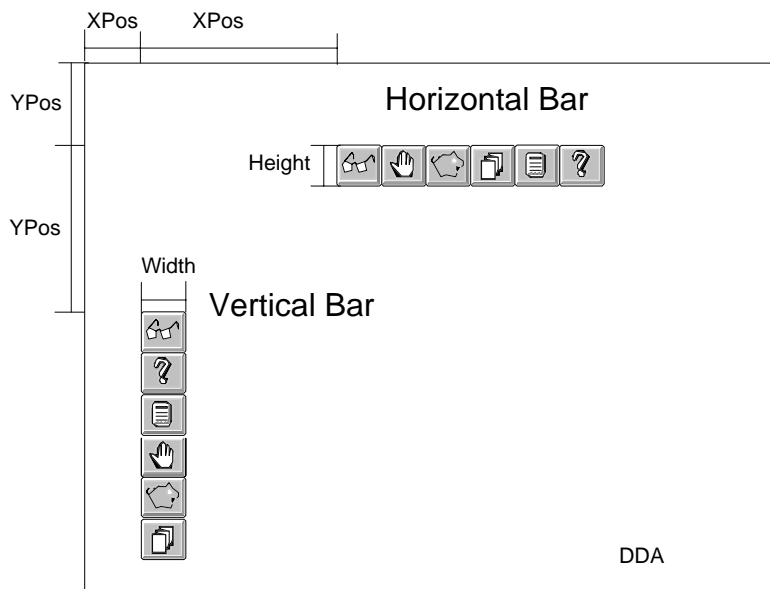


Figure 16: Button Bar

**Attributes:**

**XPos:** This attribute carries the horizontal position of the element in Normalised Device Co-ordinate (NDC).

**YPos:** This attribute carries the vertical position of the element in NDC.

**Horizontal:** The values of this Boolean attribute carry following meanings:

true: horizontal presentation of the element;

false: vertical presentation of the element.

**Height:** This attribute carries the height of the Button Bar. It is only applicable to horizontal Button Bars.

**Width:** This attribute carries the width of the Button Bar. It is only applicable to vertical Button Bars.

**FirstActive:** This attribute carries the CIN of the Button which is active by default, the first time the object is opened.

**Modal:** The values of this Boolean attribute carry following meanings:

true: the object shall be modal;

false: the object shall not be modal.

**Opened:** The values of this Boolean attribute carry following meanings:

true: the element shall be in the opened state;

false: the element shall be in the closed state.

**Accessible:** The values of this Boolean attribute carry following meanings:

true: the element can be accessed;

false: the element cannot be accessed.

### 7.3.1 Composition

#### 7.3.1.1 The Button component

**Description:** The Button is a rectangular area in the DDA. It can be formed either with text or Videotex data.

If formed with text data it should be represented by a text label, possibly associated with a 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. The terminal may centre the text data in the element display area reserved by the VEMMI application for the component.

The drawing of the Button and the possible visual trigger effect are terminal dependent.

If formed with Videotex data, the entire element display area should be reserved for the display of the Videotex data.

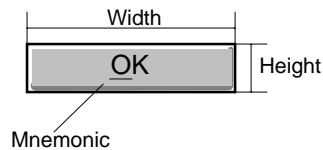
**Behaviour:** When a Button is active, the terminal shall emphasise it. Buttons may be validated by a specified mnemonic key which should be visually indicated. The key is defined by the VEMMI application and transmitted within the Button description.

When a trigger effect is implemented the Button shall go back to the initial display state after user validation.

#### Interactive functionalities:

- activation;
- validation.

#### Visual aspect (see figure 17):



**Figure 17: Button**

**Attributes:**

- CIN:** This attribute carries the Component Identification Number.
- Height:** This attribute carries the height of the component. It is only applicable to Buttons of vertical Button Bars.
- Width:** This attribute carries the width of the component. It is only applicable to Buttons of horizontal Button Bars.
- Accessible:** The values of this Boolean attribute carry following meanings:  
 true: the element can be accessed;  
 false: the element cannot be accessed.
- Text:** This attribute carries the text content of the component.
- Mnemonic:** This attribute carries the character that may be entered by the user to validate the component if the Button Bar is active.
- VTX:** This attribute carries the Videotex content of a component.
- LocActAct:** This attribute carries the code for the local action which is associated to the component and is triggered by its activation.
- LocActVal:** This attribute carries the code for the local action which is associated to the component and is triggered by its validation.

**7.4 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 functionalities. The two different groups are named:

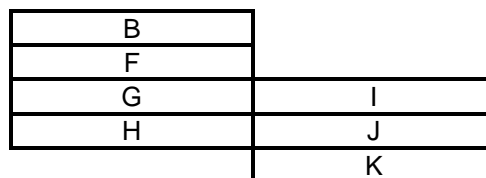
- Primary Pop-Up Menu;
- Cascading Menu.

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

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).

A Menu Choice Separator component can be used to visually separate menu choice components in a logical group.

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



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

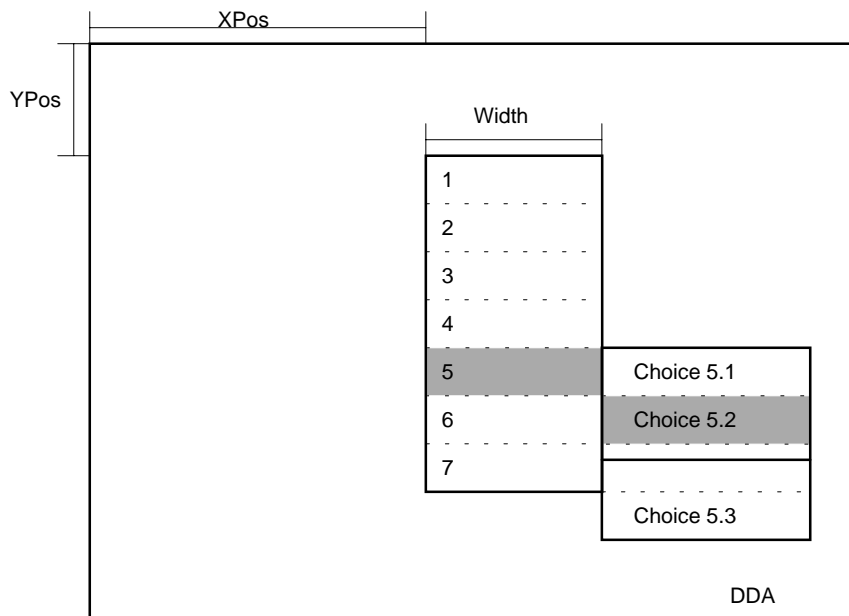
The description of the structure of the object is then:

- 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 functionalities:**

- shifting.

**General visual aspect (see figure 18):**



**Figure 18**

A text title may be given to the Pop-Up Menu which shall be displayed in the 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.
- Width:** This attribute carries the width of the Pop-Up Menu.
- 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 values of this Boolean attribute carry following meanings:  
  
true: the object shall be modal;  
  
false: the object shall not be modal.
- Opened:** The values of this Boolean attribute carry following meanings:  
  
true: the element shall be in the opened state;  
  
false: the element shall be in the closed state.
- Accessible:** The values of this Boolean attribute carry following meanings:  
  
true: the element can be accessed;  
  
false: the element cannot be accessed.

#### 7.4.1 Composition

##### 7.4.1.1 Menu Choice Pop-Up components

**Description:** The Primary Pop-Up Menu is a consecutive list of Menu Choice Pop-Up components presented vertically on several rows.

The number of columns allocated for each Menu Choice Pop-Up component shall be the same.

A Menu Choice Separator component can be used to visually separate Menu Choice Pop-Up components.

All Menu Choice Pop-Up components shall have the same type of data content (either text data or Videotex data).

**Behaviour:** When the Primary Pop-Up Menu is active, the terminal shall emphasise one Menu Choice Pop-Up component, and offer to the user shifting and validation facilities. Components may be validated by a specified mnemonic key which should be visually indicated. The key is defined by the VEMMI application and transmitted within the Menu Choice Pop-Up component description.

If a Menu Choice Pop-Up component is validated, if existing, the associated local action shall be performed. Then, if existing, the associated Cascading Menu shall be opened (in that case the Menu Choice Pop-Up component remains emphasised). If no Cascading Menu is associated to the Menu Choice Pop-Up component, after a user validation the associated local action shall be performed, then the Pull-Down Menu shall be closed.

**Interactive functionalities:**

- activation;
- validation.

**Visual aspect (see figure 19):**

Choice 1	Height
Choice 2	Height
Two row	
Choice 3	Height
Choice 4	Height
Choice 5	Height

**Figure 19: Pop-Up Menu**

**Attributes:**

- CIN:** This attribute carries the Component Identification Number.
- Height:** This attribute carries the height of the component.
- Accessible:** The values of this Boolean attribute carry following meanings:  
true: the element can be accessed;  
false: the element cannot be accessed.
- Text:** This attribute carries the text content of the component.
- Mnemonic:** This attribute carries the character that may be entered by the user to validate the component if the Primary Pop-Up Menu is active.
- VTX:** This attribute carries the Videotex content of a component.
- LocActAct:** This attribute carries the code for the local action which is associated to the component and is triggered by its activation.
- LocActVal:** This attribute carries the code for the local action which is associated to the component and is triggered by its validation.

### 7.4.1.2 Menu Choice Cascading components

Description, Behaviour, Interactive functionalities, Attributes:

see subclause 7.2.1.3.

Visual aspect (see figure 20):

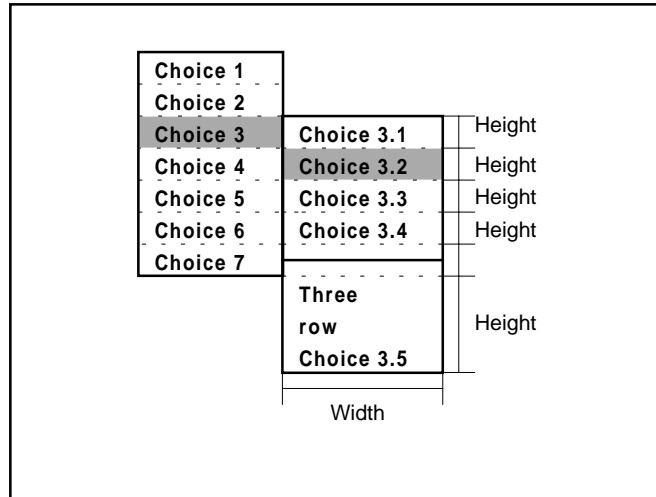


Figure 20: Pop-Up Menu with associated Cascading Menu

### 7.4.1.3 Menu Choice Separator components

See subclause 7.2.1.4

### 7.5 The Dialogue Box

**Description:** The Dialogue Box is a rectangular area in the DDA which contains VEMMI components in order to establish user interaction.

A text title may be given to the Dialogue Box which shall be displayed in it's first row.

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.

**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 reception order of the components of the Dialogue Box.

**Interactive functionalities:**

- moving.

**Visual aspect (see figure 21):**

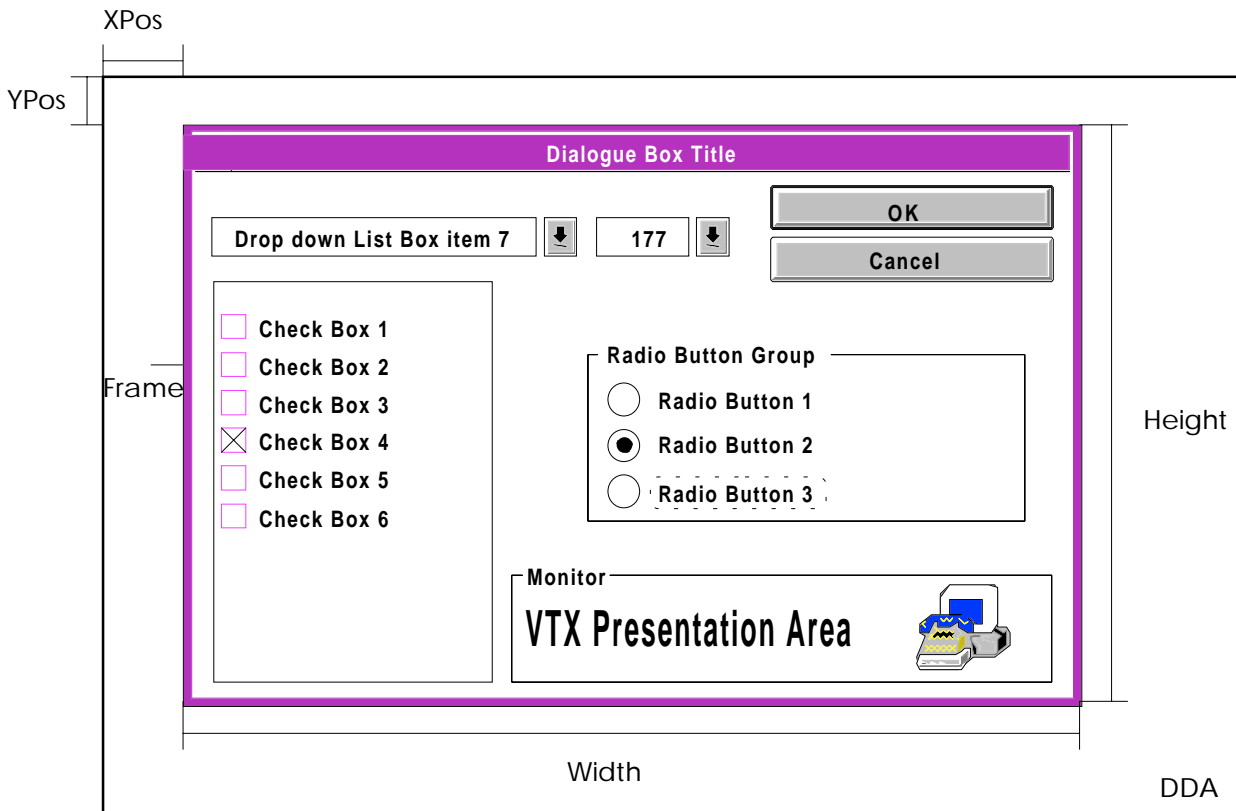


Figure 21: Dialogue Box

**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.

**Border:** The values of this Boolean attribute carry following meanings:

true: a border shall be drawn by the terminal to frame the object. One character position should be reserved for the drawing of the border;

false: 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 values of this Boolean attribute carry following meanings:

true: the object shall be modal;

false: the object shall not be modal.

**Opened:** The values of this Boolean attribute carry following meanings:

true: the element shall be in the opened state;

false: the element shall be in the closed state.

**Accessible:** The values of this Boolean attribute carry following meanings:

true: the element can be accessed;

false: the element cannot be accessed.

**StoreInitialValues:** The values of this Boolean attribute carry following meanings:

true: the terminal shall store the initial values of the component at the time of the object creation. If a local action "restore initial values" is applied these stored values shall be restored;

false: it shall not be mandatory for the terminal to store the initial values.

### 7.5.1 Composition

#### 7.5.1.1 The Separator component

**Description:** The Separator is either a horizontal or a vertical solid line on one character used to separate different areas within a Dialogue Box.

**Behaviour:** The Separator shall be inaccessible.

**Interactive functionalities:**

- none.

**Visual aspect (see figure 22):**

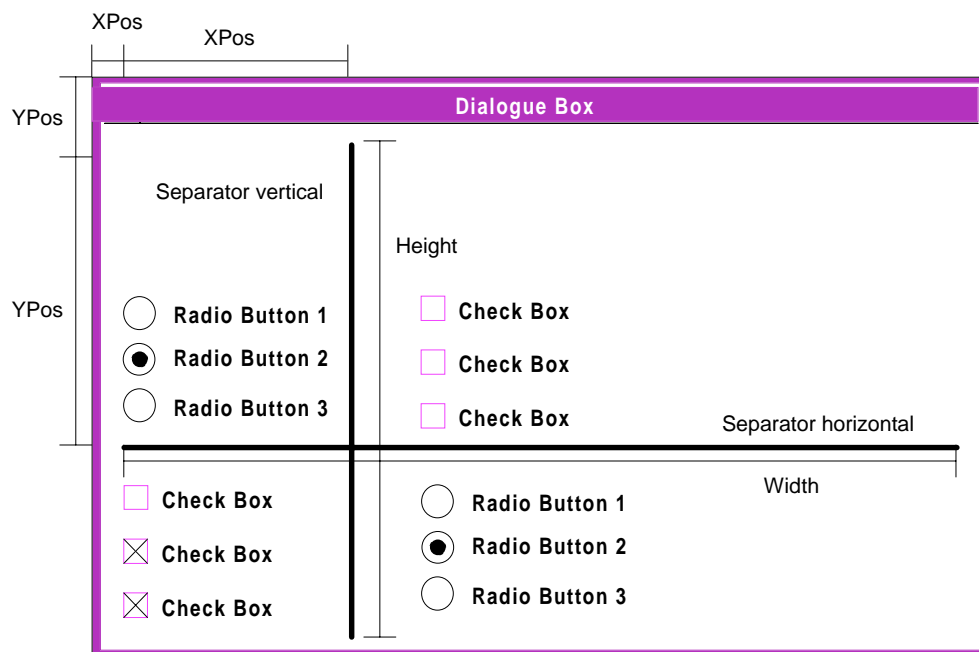


Figure 22: Separator

**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.

**Horizontal:** The values of this Boolean attribute carry following meanings:

true: horizontal presentation of the element;

false: vertical presentation of the element.

**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.

**Opened:** The values of this Boolean attribute carry following meanings:

true: the element shall be in the opened state;

false: the element shall be in the closed state.

### 7.5.1.2 The Frame component

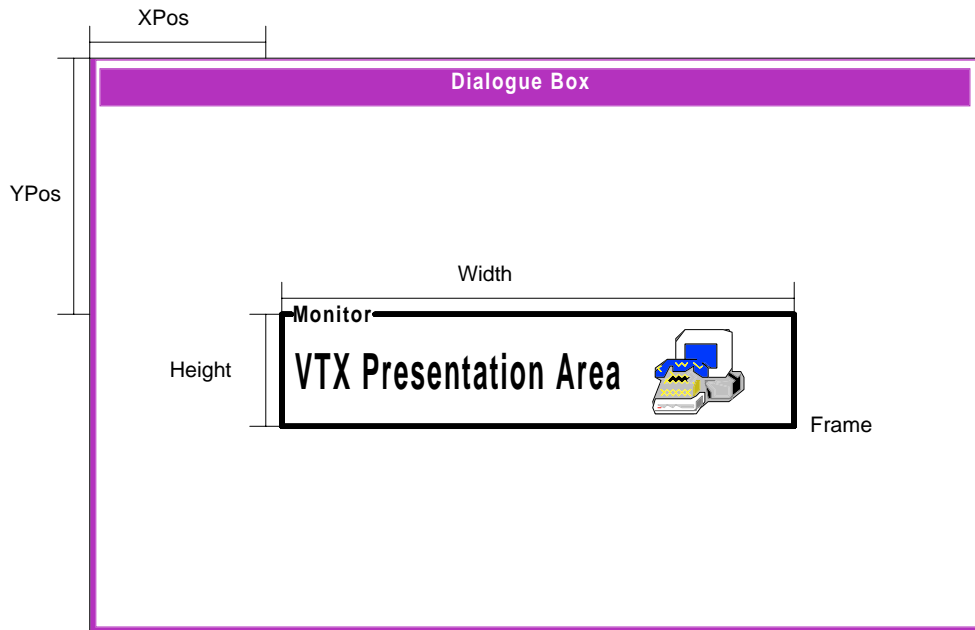
**Description:** A Frame consists of four solid lines on one character which visually separate a rectangular area of the Dialogue Box.

**Behaviour:** The Frame shall be inaccessible.

**Interactive functionalities:**

- none.

**Visual aspect (see figure 23):**



**Figure 23: Frame titled with 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.
- Opened:** The values of this Boolean attribute carry following meanings:  
true: the element shall be in the opened state;  
false: the element shall be in the closed state.



### 7.5.1.3 The Text Presentation Area component

**Description:** The Text Presentation Area is an area intended to present text data to the user. It is composed of two items:

- the text label;
- the text output field.

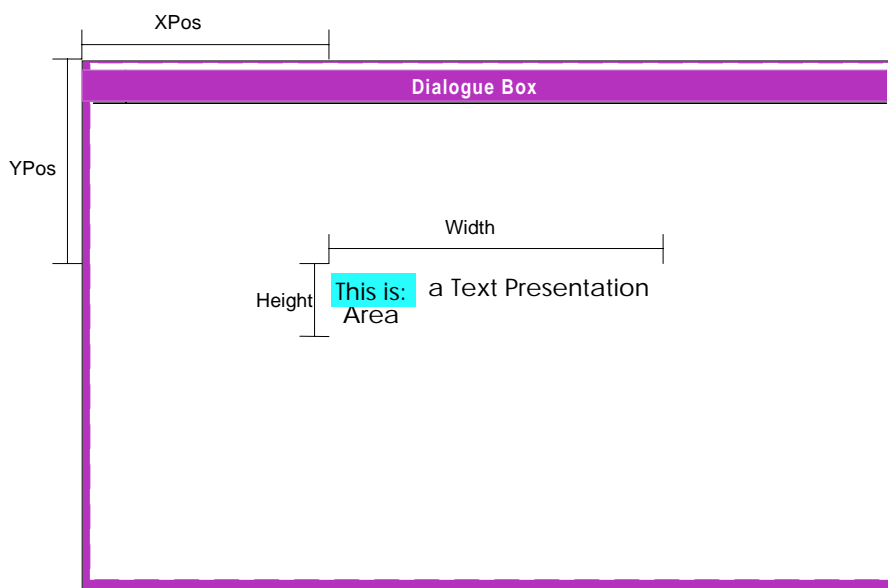
The text output field has its starting location immediately after the text label.

**Behaviour:** The text presentation area shall be inaccessible.

**Interactive functionalities:**

- none.

**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.
- Opened:** The values of this Boolean attribute carry following meanings:  
true: the element shall be in the opened state;  
false: the element shall be in the closed state.
- TextLabel:** This attribute carries the text label of the component.
- Text:** This attribute carries the text content of the component.

#### 7.5.1.4 The Videotex Presentation Area component

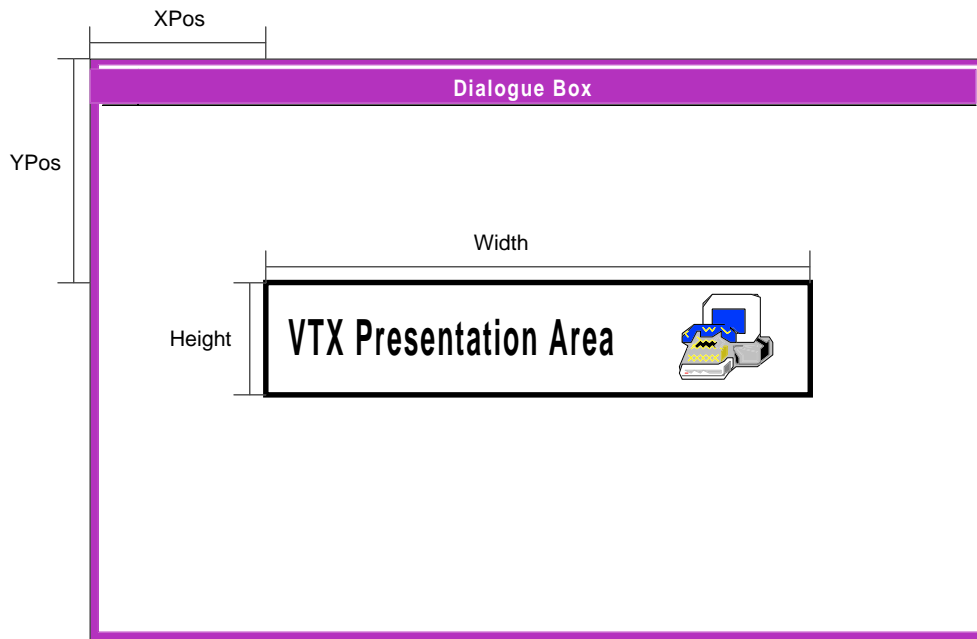
**Description:** The Videotex Presentation Area is an area intended to present Videotex data to the user.

**Behaviour:** The Videotex Presentation Area shall be inaccessible.

**Interactive functionalities:**

- none.

**Visual aspect (see figure 25):**



**Figure 25: Videotex 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.
- Opened:** The values of this Boolean attribute carry following meanings:  
true: the element shall be in the opened state;  
false: the element shall be in the closed state.
- VTX:** This attribute carries the Videotex content of a component.

### 7.5.1.5 The Push Button component

#### Description, Behaviour, Interactive functionalities

See subclause 7.3.1.1.

Visual aspect (see figure 26):

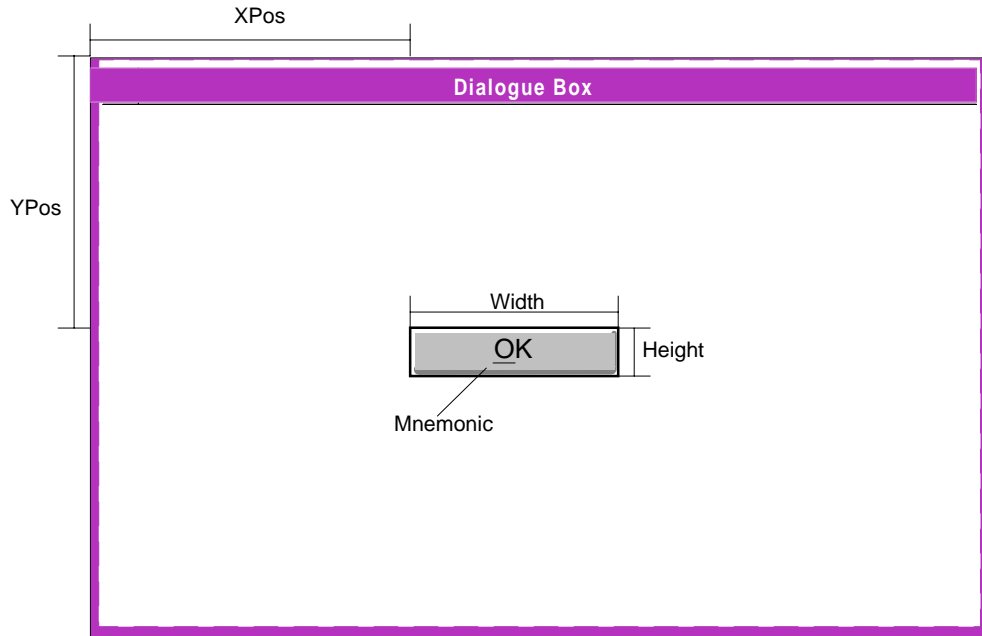


Figure 26: Push Button

#### 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.
- Opened:** The values of this Boolean attribute carry following meanings:  
true: the element shall be in the opened state;  
false: the element shall be in the closed state.
- Accessible:** The values of this Boolean attribute carry following meanings:  
true: the element can be accessed;  
false: the element cannot be accessed.
- Text:** This attribute carries the text content of the component.
- VTX:** This attribute carries the Videotex content of a component.

- Mnemonic:** This attribute carries the character that may be entered by the user to validate the component if the Push Button is active.
- LocActAct:** This attribute carries the code for the local action which is associated to the component and is triggered by its activation.
- LocActVal:** This attribute carries the code for the local action which is associated to the component and is triggered by its validation.

#### 7.5.1.6 The Text Input Field component

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

- the text label;
- the input area.

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 types are available:

- any text character;
- alphabetic (A..Z, a..z, diacritical characters);
- numeric (0..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.

**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. In order to display all possible characters, enough space should be given for user inputs. If the maximum number of the input characters is reached this should be indicated to the user. An empty string is a valid input for a text input field.

**Constraints:** The maximum length of the text label is limited to the first line of the Text Input Field component width.

**Interactive functionalities:**

- activation;
- validation;
- local editing functionalities of the input area (terminal dependent).

Visual aspect (see figure 27):

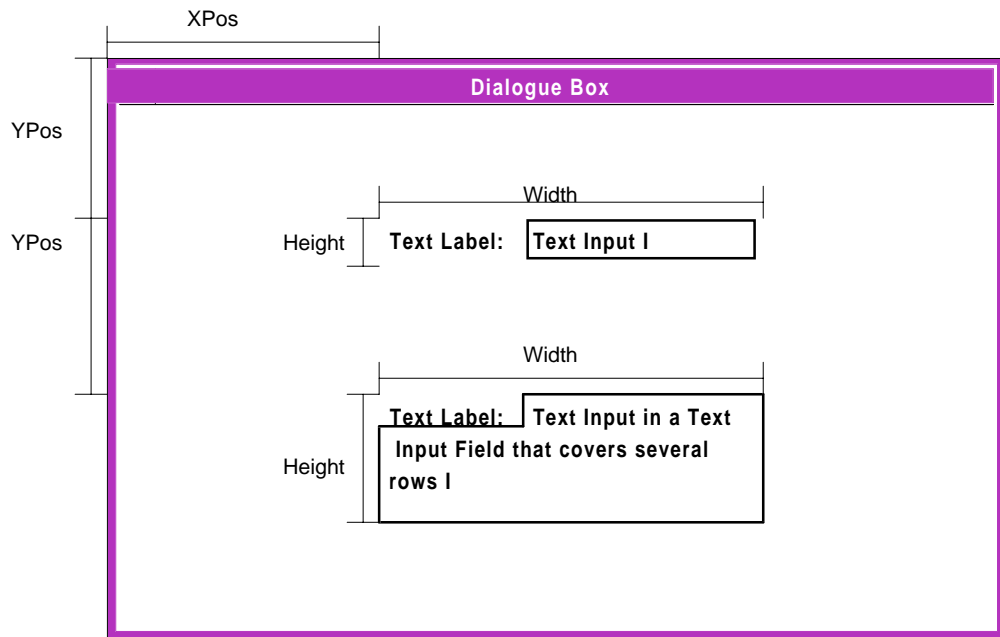


Figure 27: Text Input Field

**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.
- Opened:** The values of this Boolean attribute carry following meanings:  
true: the element shall be in the opened state;  
false: the element shall be in the closed state.
- Accessible:** The values of this Boolean attribute carry following meanings:  
true: the element can be accessed;  
false: the element cannot be accessed.
- DefValue:** This attribute carries the default value for the component.
- TextLabel:** This attribute carries the text label of the component.
- InputType:** This attribute specifies which type of user inputs shall be accepted by the component.
- Echo:** 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.

- MaxChar:** This attribute carries the maximum number of characters the user can enter in the component.
- InputTransformation:** This attribute specifies if the terminal shall convert the alphabetic characters entered by the user to upper case characters, to lower case characters or if no conversion should be done.
- LocActAct:** This attribute carries the code for the local action which is associated to the component and is triggered by its activation.
- LocActVal:** This attribute carries the code for the local action which is associated to the component and is triggered by its validation.

### 7.5.1.7 The Check Box component

**Description:** The Check Box component is composed of two items:

- the text label item;
- the check item.

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 functionalities:**

- activation;
- validation;
- switch between the values marked and unmarked.

**Visual aspect (see figure 28):**

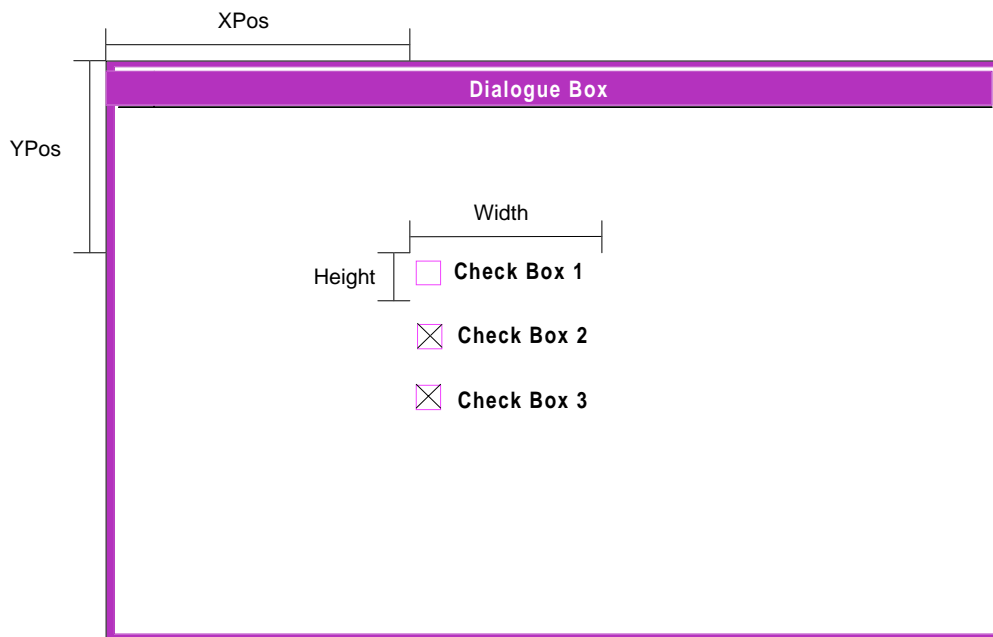


Figure 28: Check Box

**Attributes:**

<b>CIN:</b>	This attribute carries the Component Identification Number.
<b>XPos:</b>	This attribute carries the horizontal position of the element in NDC.
<b>YPos:</b>	This attribute carries the vertical position of the element in NDC.
<b>Width:</b>	This attribute carries the width of the component.
<b>Height:</b>	This attribute carries the height of the component.
<b>Opened:</b>	The values of this Boolean attribute carry following meanings:  true: the element shall be in the opened state;  false: the element shall be in the closed state.
<b>Accessible:</b>	The values of this Boolean attribute carry following meanings:  true: the element can be accessed;  false: the element cannot be accessed.
<b>TextLabel:</b>	This attribute carries the text label of the component.
<b>DefMarked:</b>	The values of this Boolean attribute carry following meanings:  true: the default value of the component shall be marked;  false: the default value of the component shall be unmarked.
<b>LocActAct:</b>	This attribute carries the code for the local action which is associated to the component and is triggered by its activation.
<b>LocActVal:</b>	This attribute carries the code for the local action which is associated to the component and is triggered by its validation.

#### 7.5.1.8 The Radio Button component

**Description:** The Radio Button component is composed of two items:

- the text label item;
- the check item.

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) may be.

**Interactive functionalities:**

- activation;
- validation;
- switch between the values marked and unmarked.

Visual aspect (see figure 29):

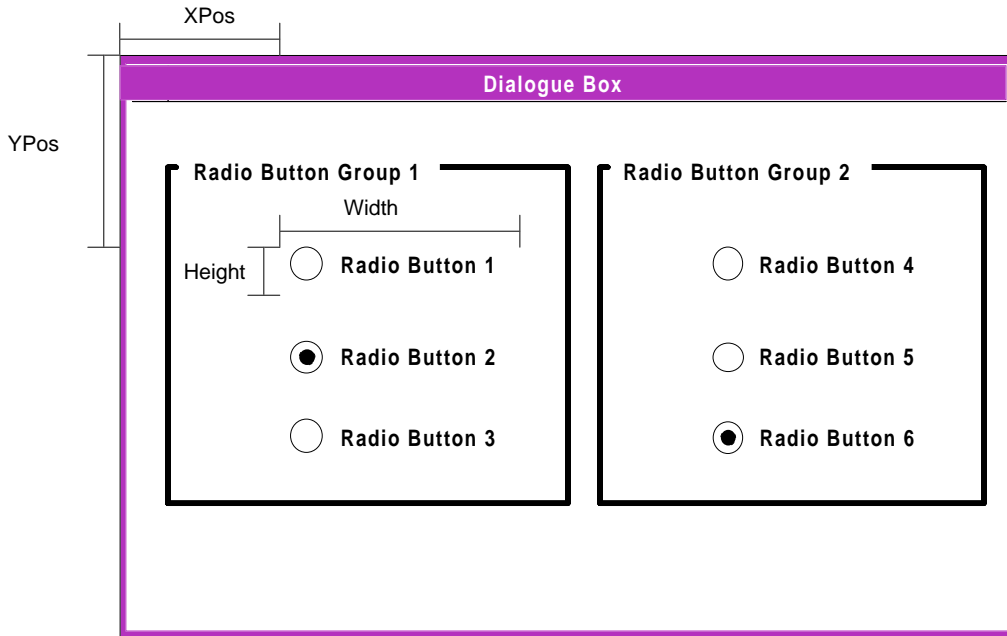


Figure 29: Radio Buttons with a Frame and a Text Presentation Area

**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.
- Opened:** The values of this Boolean attribute carry following meanings:  
true: the element shall be in the opened state;  
false: the element shall be in the closed state.
- Accessible:** The values of this Boolean attribute carry following meanings:  
true: the element can be accessed;  
false: the element cannot be accessed.
- TextLabel:** This attribute carries the text label of the component.
- 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 values of this Boolean attribute carry following meanings:  
true: the default value of the component shall be marked;  
false: the default value of the component shall be unmarked.



**LocActAct:** This attribute carries the code for the local action which is associated to the component and is triggered by its activation.

**LocActVal:** This attribute carries the code for the local action which is associated to the component and is triggered by its validation.

### 7.5.1.9 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. When the entire list is visible the scrolling controls may be invisible. 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 dependant (vertical scrollbar, buttons, etc.).

The list content (maximum 128 list items) is within the definition of the component. There shall be no CR+LF within the text content of a list item.

**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.

#### Interactive functionalities:

- activation;
- validation (single or multiple choice);
- scrolling;
- shifting.

#### Visual aspect (see figure 30):

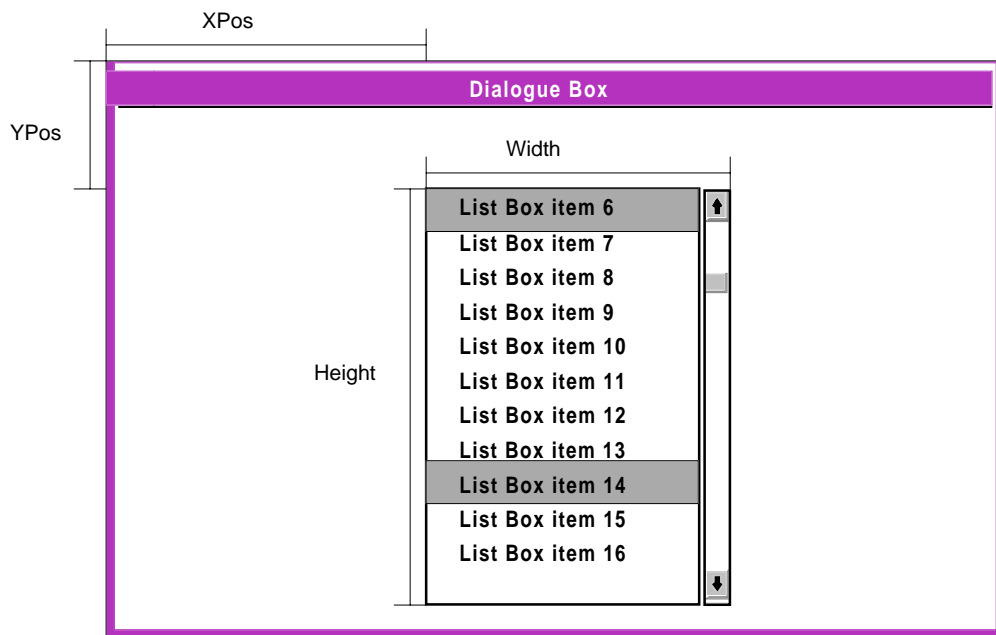


Figure 30: List Box

**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.
- Opened:** The values of this Boolean attribute carry following meanings:  
true: the element shall be in the opened state;  
false: the element shall be in the closed state.
- Accessible:** The values of this Boolean attribute carry following meanings:  
true: the element can be accessed;  
false: the element cannot be accessed.
- DefItem:** This attribute carries the index of the list item that is selected by default when the component is opened for the first time.
- ListIndex:** This attribute carries the index of the next 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 corresponding to the list text. If, in modify command, a list index for an already existing list item is used, the existing list item shall be 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 which is already existing but in the modify command the corresponding attribute "ListText" is missing, then the referenced list item on the terminal shall be deleted.
- ListText:** This attribute carries the text content of a list item.
- SingleChoice:** The values of this Boolean attribute carry following meanings:  
true: the component allows the user to choose one option out of a set of given options;  
false: the component allows the user to choose one or more options out of a set of given options.
- LocActAct:** This attribute carries the code for the local action which is associated to the component and is triggered by its activation.
- LocActVal:** This attribute carries the code for the local action which is associated to the component and is triggered by its validation.

### 7.5.1.10 The Combination Box component

**Description:** The Combination Box is composed of two parts:

- a single choice List Box;
- 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 facility 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 dependant (vertical scrollbar, buttons, etc.).

The Combination Box content (maximum 128 items) is within the definition of the component. There shall be no CR+LF within the text content of a list item.

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.

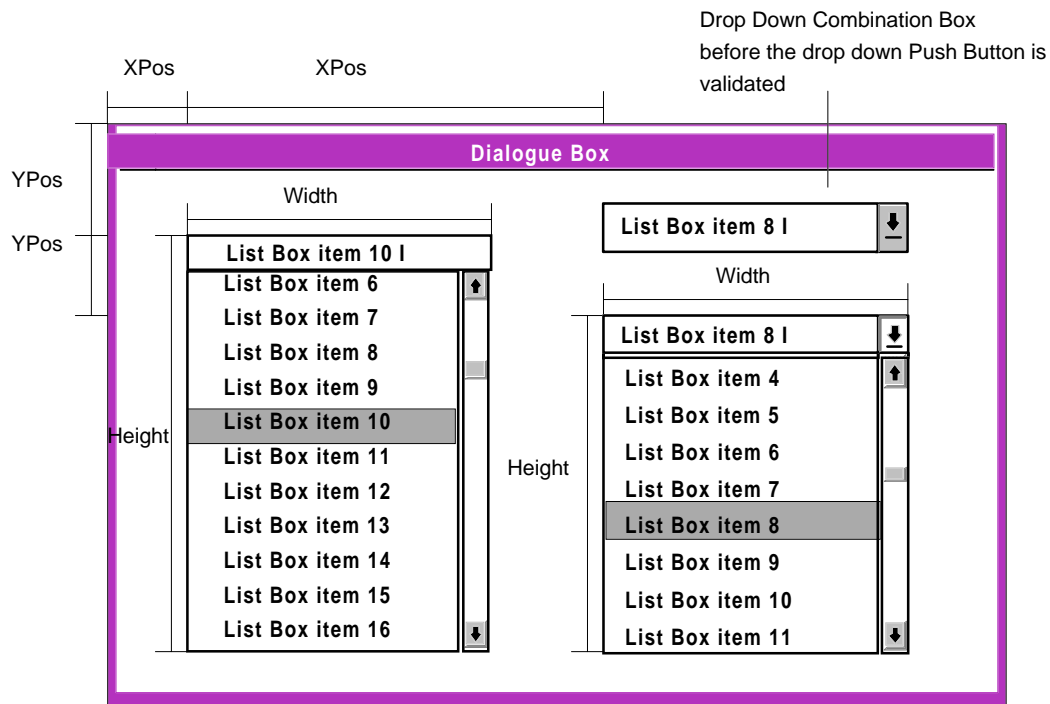
**Behaviour:** When the List Box is active the terminal shall offer shifting facilities to the user. A Combination Box contains a list of items that the user can scroll through and select from to complete the text field. A parameter of the Combination Box specifies if the text field is editable or not. 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.

If the Combination Box is designed to be a drop down Combination Box, the validation of the Push Button shall switch between the display and the hide of the List Box.

**Interactive functionalities:**

- activation;
- validation (single choice);
- scrolling;
- editing.

**Visual aspect (see figure 31):**



**Figure 31: 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.
- Opened:** The values of this Boolean attribute carry following meanings:  
 true: the element shall be in the opened state;  
 false: the element shall be in the closed state.
- Accessible:** The values of this Boolean attribute carry following meanings:  
 true: the element can be accessed;  
 false: the element cannot be accessed.
- DefItem:** This attribute carries the index of the list item that is selected by default when the component is opened for the first time.

- ListIndex:** This attribute carries the index of the next 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 corresponding to the list text. If, in a modify command, a list index for an already existing list item is used, the existing list item shall be 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 which is already existing but in the modify command the corresponding attribute "ListText" is missing then the referenced list item on the terminal shall be deleted.
- ListText:** This attribute carries the text content of a list item.
- DropDown:** The values of this Boolean attribute carry following meanings:
- true: the component shall support the drop down facility;
  - false: the component shall not support the drop down facility.
- EditableInput:** The values of this Boolean attribute carry following meanings:
- true: the text field of the component shall be editable;
  - false: the text field of the component shall not be editable.
- ConsistencyCheck:** This attribute shall only be present if the "EditableInput" Parameter is set true. The values of this Boolean attribute carry following meanings:
- true: the text entered by the user shall correspond to a list item;
  - false: the user may enter any text string.
- 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.
- LocActAct:** This attribute carries the code for the local action which is associated to the component and is triggered by its activation.
- LocActVal:** This attribute carries the code for the local action which is associated to the component and is triggered by its validation.

### 7.5.1.11 The Sensitive Area component

Sensitive Area components are common to the Dialogue Box and the Presentation Box.

**Description:** A Sensitive Area component is an area, non-materialised, in which the VEMMI application permits a validation operation. A Sensitive Area component shall be used locally associated with a Text or Videotex Presentation Area component of the Dialogue Box or with the Text-Videotex Output Field data blocks of the Presentation Box. In the Dialogue Box, the start position of a Sensitive Area shall always be referred to the origin of the component to which it is associated. The start position of a Sensitive Area is defined by the row and the character position co-ordinate. The horizontal dimension is defined by the number of character positions in a row. The vertical dimension is defined by the number of rows.

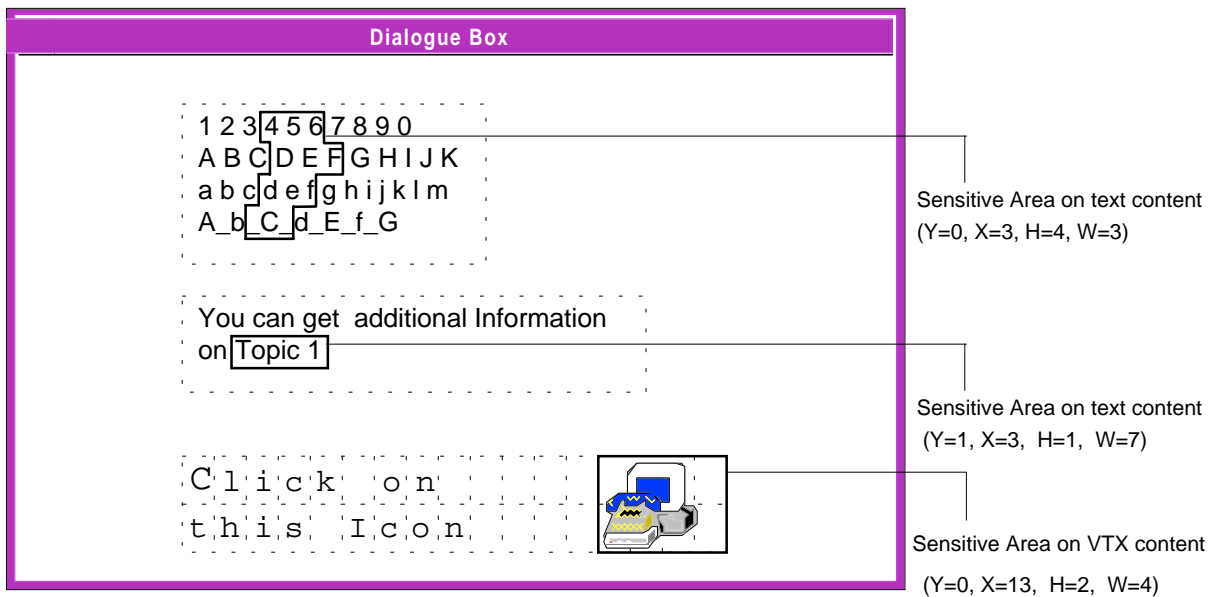
**Behaviour:** When active, a visible effect (e.g. thread, dotted lines) can be implemented by the local manager.

**Constraints:** A Sensitive Area shall not be covered by other components or items which are sensitive to user interaction, in order to avoid conflicts. The value of the "Opened" attribute of the Sensitive Area shall correspond to the value of the "Opened" attribute of the associated component.

#### Interactive functionalities:

- activation;
- validation.

#### Visual aspect (see figure 32):



Abbreviations: Y: vertical starting position      H: Height  
X: horizontal starting position      W: Width

**Figure 32: Sensitive Area**

**Attributes:**

- CIN:** This attribute carries the Component Identification Number.
- AssCIN:** This attribute carries the CIN of the Text or VTX Presentation Area to which the component is associated.
- XCharPos:** This attribute carries the horizontal cursor position of the components start, referring to the origin of the associated component.
- YCharPos:** This attribute carries the vertical cursor position of the components start, referring to the origin of the associated component.
- Width:** This attribute carries the width of the component.
- Height:** This attribute carries the height of the component.
- Opened:** The values of this Boolean attribute carry following meanings:  
  
true: the element shall be in the opened state;  
  
false: the element shall be in the closed state.
- Accessible:** The values of this Boolean attribute carry following meanings:  
  
true: the element can be accessed;  
  
false: the element cannot be accessed.
- LocActAct:** This attribute carries the code for the local action which is associated to the component and is triggered by its activation.
- LocActVal:** This attribute carries the code for the local action which is associated to the component and is triggered by its validation.

### 7.5.1.12 The Locator component

**Description:** A Locator is used to get the current cursor position in a text or Videotex output component.

**Behaviour:** The purpose of this component is getting the co-ordinates, in a Videotex output component, of the cursor at the time of a validation event. The co-ordinates are always referred to the origin of the component to which they are associated. When the Locator is accessed, it becomes active; the borders should be indicated (e.g. by a dotted line) and a moveable prompt should appear. A Locator shall memorise the last value chosen by the user with the interaction tools.

**Constraints:** A Locator shall not be covered by other components or items which are sensitive to user interaction, in order to avoid conflicts. The value of the "Opened" attribute of the Locator shall correspond to the value of the "Opened" attribute of the associated component.

**Interactive functionalities:**

- activation;
- validation;
- moveable prompt.

**Visual aspect (see figure 33):**

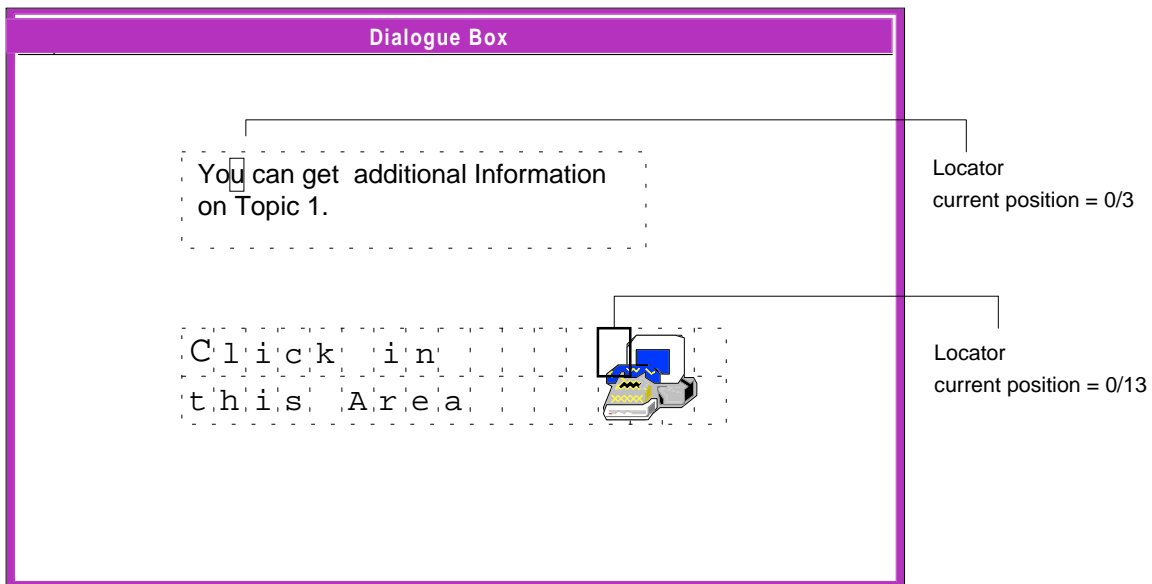


Figure 33: Locator



**Attributes:**

- CIN:** This attribute carries the Component Identification Number.
- AssCIN:** This attribute carries the CIN of the Text or VTX Presentation Area to which the component is associated.
- Opened:** The values of this Boolean attribute carry following meanings:  
true: the element shall be in the opened state;  
false: the element shall be in the closed state.
- Accessible:** The values of this Boolean attribute carry following meanings:  
true: the element can be accessed;  
false: the element cannot be accessed.
- LocActAct:** This attribute carries the code for the local action which is associated to the component and is triggered by its activation.
- LocActVal:** This attribute carries the code for the local action which is associated to the component and is triggered by its validation.

## 7.6 The Presentation Box

**Description:** The Presentation Box is a rectangular area in the DDA which presents Text or Videotex data in a Text-Videotex Output Field component and which contains VEMMI components in order to enable user interaction (Push Button, Text Input Area, Sensitive Area and Locator). In a Presentation Box the Output Field is either a Text Output Field or a Videotex Output Field. Text and VTX are never mixed in a Text-Videotex Output Field component. A Presentation Box has always one and only one Text-VTX Output Field, with a content possibly split in data blocks transmitted on terminal requests.

The Presentation Box may have to present a large amount of data which cannot be stored in the terminal. Therefore this data shall be split into several data blocks which are transferred to the terminal on its request. If the space needed for the presentation of the data is bigger than the space allocated for Text-Videotex Output Field, tools for vertical scrolling shall be provided by the VEMMI local manager.

An attribute of the Presentation Box specifies if the representation of scrolling tools is terminal dependent or not. If not, the application can reserve either two columns or two rows of the Text-Videotex Output Field for these scrolling tools. Preferably, the application should reserve the two rightmost columns or the two bottommost rows. The size of the Text-Videotex Output Field that can be used to display data to the user is then reduced by the area reserved for the scrolling tools.

If scrolling is needed, but no specific position and size reservation is requested for the scrolling tools by the VEMMI application, the local manager may present:

- scrolling tools preferably next to the right side of the Text-Videotex Output Field;
- reduced scrolling tools as buttons overlapping the Text-Videotex Output Field;
- keys which provide the scrolling functionality.

The scrolling tools shall be clearly identifiable by the user particularly when they are reported on keys.

Push Button can be used to associate some controls to the presentation of text and Videotex data in a Presentation Box (e.g. Help, Cancel, OK,..). The Push Button can have text or Videotex data content. When Push Buttons are defined they shall be positioned into a button service area located immediately under the Text-Videotex Output Field and dimensioned by the difference between the Presentation Box height, the Text-VTX Output Field height and, if requested, the border.

Text Input Area, Sensitive Area and Locator components should only be emphasised when activated by the user.

The Presentation Box can be provided with a border area. This area should be equal to one character position on the left side and on the right side of the Presentation Box, and to one row on the top and on the bottom. This border shall be included in the dimensions of the Presentation Box. When a border is requested, a frame shall be drawn by the terminal.

An example for the dimensioning of the Presentation Box is given in figure 34

If the Presentation Box has a Videotex content, a default Videotex content can be defined and used on request in every data block. An attribute of the Text-Videotex Output Field specifies for each data block if the data presented to the user shall be a combination of the default content and of the data block (DefVTX + VTX), or if only the content of the data block shall be presented.

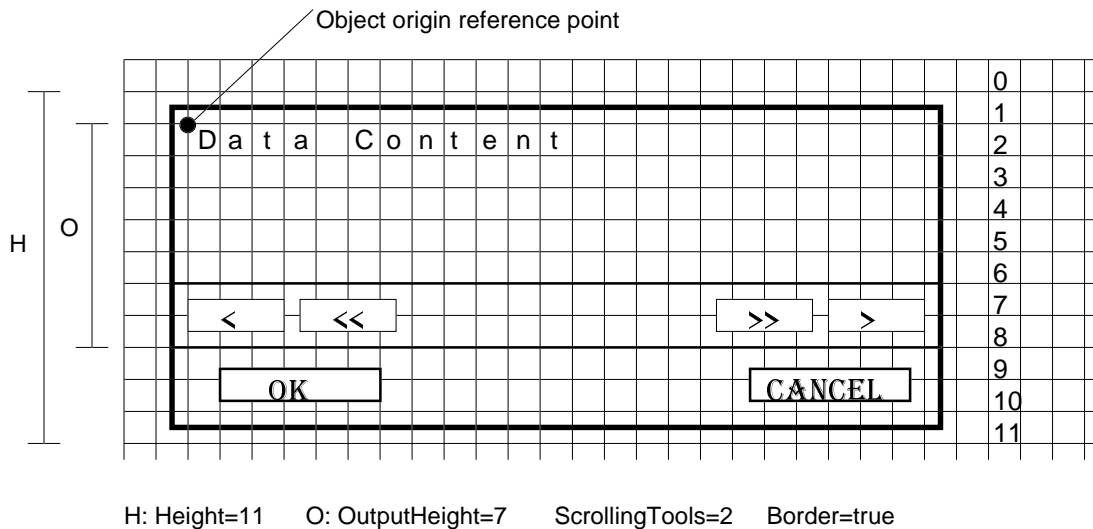
**Behaviour**

The general principles for active state and focus management expressed in subclause 4.11.1 apply.

The Presentation Box can be declared modal or non-modal by the VEMMI application.

**Interactive functionalities:**

- moving.



**Figure 34: Dimensioning of the Presentation Box**

Visual aspect (see figure 35):

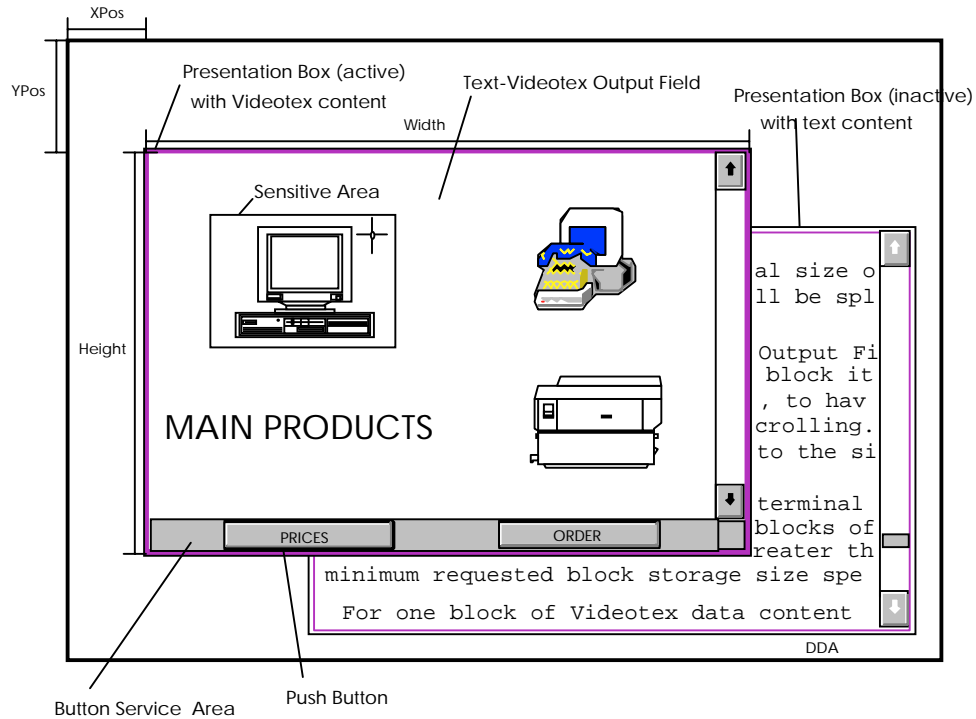


Figure 35: Presentation Box

**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, including possible requested scrolling tools and border.

**Height:** This attribute carries the height of the object, including possible requested scrolling tools and border.

**ContentType:** This attribute specifies which type of content is used in the object.

**ScrollingTools:** The values of this attribute carry following meanings:

- 0: scrolling tools are terminal dependant, but in conformance to the previously defined rules;
- 1: two columns of the Text-Videotex Output Field shall be reserved for the presentation of the tools for vertical scrolling;
- 2: two rows of the Text-Videotex Output Field shall be reserved for the presentation of the tools for vertical scrolling.

**OutputHeight:** This attribute carries the height of the Text-VTX Output Field.

**Border:** The values of this Boolean attribute carry following meanings:

true: a border shall be drawn by the terminal to frame the object. One character position should be reserved for the drawing of the border;

false: no border shall be drawn.

- FirstActive:** This attribute carries the CIN of the component which is active by default, the first time the object is opened.
- DefVTX:** This attribute carries Videotex data which can be used as a template for each data block. The UseDef attribute specifies for each data block if its data content shall be displayed alone or if a concatenation of the DefVTX data and the data of the data block (DefVTX+VTX) shall be displayed.
- Modal:** The values of this Boolean attribute carry following meanings:  
true: the object shall be modal;  
false: the object shall not be modal.
- Opened:** The values of this Boolean attribute carry following meanings:  
true: the element shall be in the opened state;  
false: the element shall be in the closed state.
- Accessible:** The values of this Boolean attribute carry following meanings:  
true: the element can be accessed;  
false: the element cannot be accessed.
- NumBlocks:** This attribute carries the total number of blocks of the Presentation Box. This can be useful for the local drawing of scrolling tools, for example to handle a local graphic index. When the number of blocks is unknown, NumBlocks is set to 0 by the VEMMI application.

## 7.6.1 Composition

### 7.6.1.1 The Text-Videotex Output Field component

#### Description

The Text-Videotex Output Field component is the output area of the Presentation Box where the VEMMI application displays text data content or Videotex data content to the user.

The size of the Text-Videotex Output Field is equal to the size of the Presentation Box possibly reduced by the space reserved for the scrolling tools, the button service area and for the border. The Text-Videotex Output Field and the Presentation Box have the same origin.

A Text-Videotex Output Field which covers the whole DDA may be used to present regular full screen Videotex frames in VEMMI Applications. In this case the following Presentation Box attributes shall take simultaneously the following values:

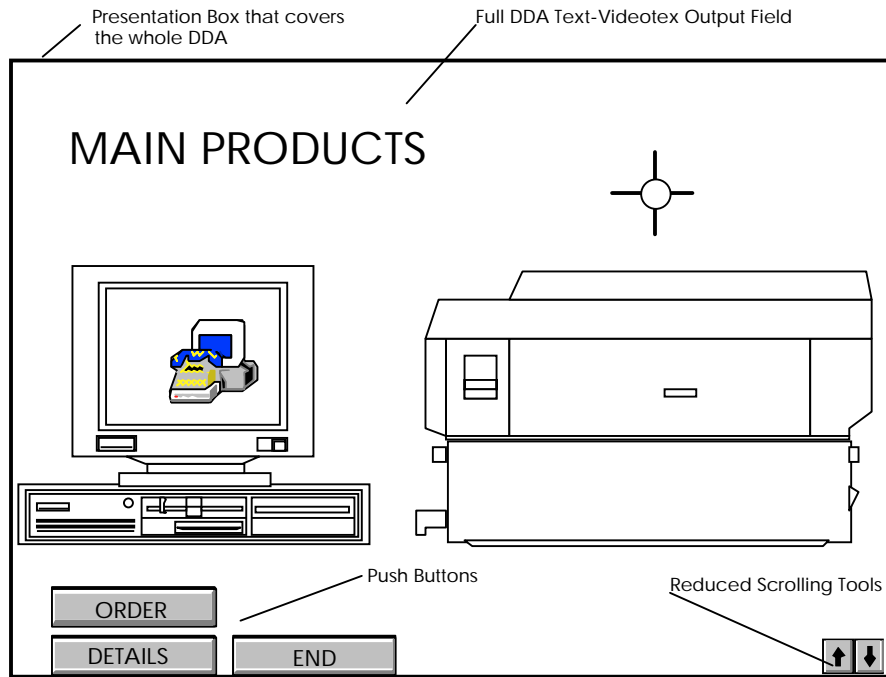
- XPos and YPos = 0;
- Height = the height of the DDA;
- Width = the width of the DDA;
- OutputHeight = the height of the DDA;
- ScrollingTools = 0.

Scrolling tools when needed are then presented locally in the most convenient way, either as reduced scrolling tools overlapping a small space on the Text-VTX Output Field, or reported on keys. Possible Buttons are also overlapping the Text-Videotex Output Field, in their positions and dimensions defined by the VEMMI Application.

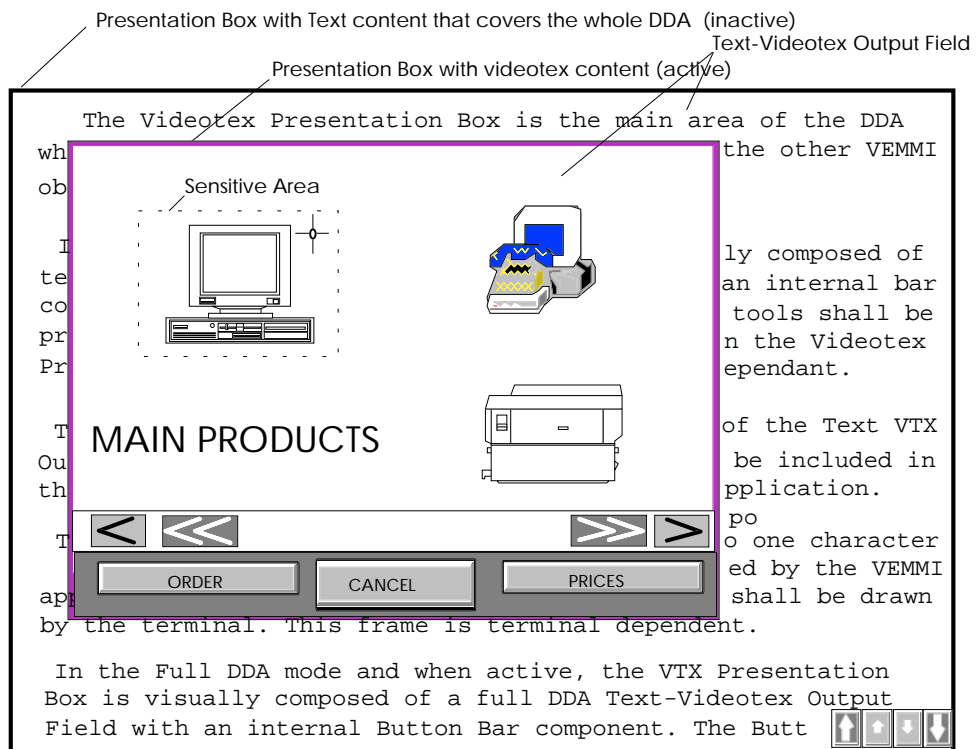
**Interactive functionalities:**

- scrolling.

**Visual aspect (see figures 36 and 37):**



**Figure 36: Text Videotex Output Field which covers the whole DDA**



**Figure 37: Text Presentation Box**

**Behaviour and scrolling management**

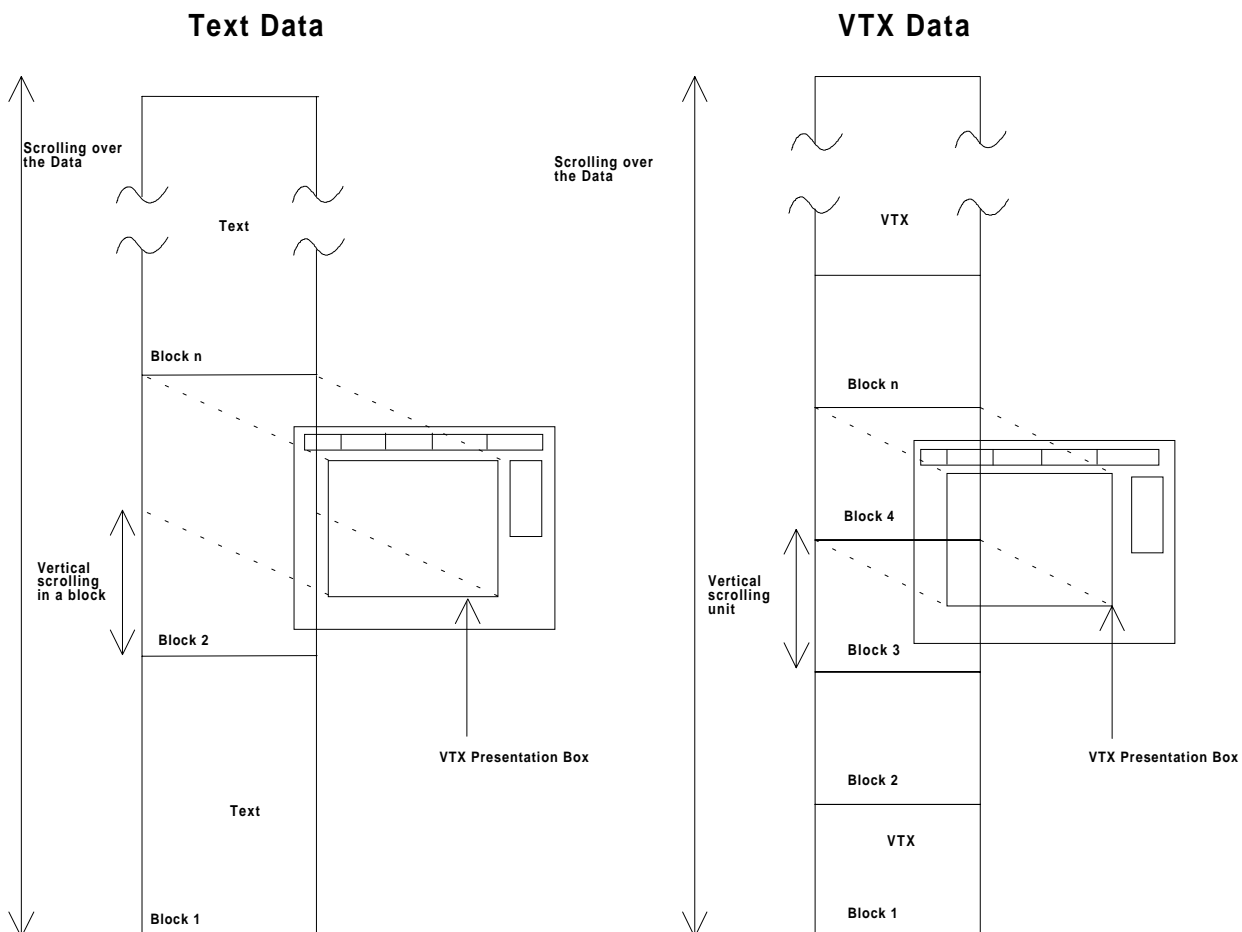
Block concept for text data

A VEMMI application may have to present text data, or Videotex data, greater than the physical size of the defined Presentation Box. In this case, the whole text data content shall be split in several blocks of data by the VEMMI application (see figure 38). The user should have the impression of a continuous text. The number of characters for a row should not exceed the number of Videotex character positions allocated for the Text Output Field. Data can be truncated if a terminal is only capable to display one text character in one Videotex character positions. The VEMMI application shall provide text line delimiters, as defined in subclause 4.5.1, control characters CR+LF then apply. There is no automatic text line folding from a row to the next one; so the text line delimiter "CR+LF" shall always be used. A block of text data shall always be terminated at the end of a line, after the CR+LF text line delimiter. If the next block of text starts with a CR+LF, this first virtual empty text line shall be considered as the first line for the positioning of associated components attached to this block.

To give the impression of a continuous text, vertical scrolling capabilities, on a row-by-row and screen-by-screen basis, shall be offered within the consecutive blocks of a Text Output Field.

Block concept for Videotex data

Each block of Videotex data content when displayed shall be physically equal to the size of the Text-Videotex Output Field area. Vertical scrolling capability is offered between consecutive Videotex blocks. This type of vertical scrolling is a block-by-block scrolling.



**Figure 38: Scrolling over text and Videotex data**

### Block transfer mechanism

Text and Videotex data blocks pertaining to the same Presentation Box shall be numbered consecutively by the VEMMI application. The first block shall be transmitted, as the content of the Text-Videotex Output Field component, within the Presentation Box description. The Presentation Box description may also include the number of blocks of the whole text or Videotex content. The local manager may use this information to design the scrolling tools and to specify the block number of the data block in the VEMMI\_More\_Data service element.

As soon as the Presentation Box is opened, the first block of data shall be displayed to the user. The user may then scroll over data. If the user requests to see data that belongs to a block which is not already stored in the terminal, the local manager shall request the transmission of this data block, with a VEMMI\_More\_Data command. The local manager should request the transmission of a new data block if the space needed for the presentation of the data block is smaller than the Text-Videotex Output Field.

If enabled by its storage capabilities, the VEMMI terminal may decide to request the transmission of some next adjacent data blocks, anticipating user scrolling. It may also store the data blocks already seen by the user. This permits a quick scrolling back over previously seen data blocks. The VEMMI application shall never send a data block through a VEMMI\_Additional\_Data service command if not invited by the terminal through a VEMMI\_More\_Data service command.

The content of a first text or Videotex data block shall be transmitted in a Text-Videotex Output Field component associated with the description of the Presentation Box. This first block is not necessarily the block number 1, it is up to the application to decide which block to present to the user first.

### Minimum block storage capacity

The VEMMI terminal shall provide a minimum storage space to store at least one block of text or Videotex data content for a Presentation Box display. This space is named the minimum requested block storage. Terminals may offer greater memory capabilities to store several blocks of content to get a better time response on user scrolling within a same Presentation Box.

For one block of text content, the minimum requested block storage size is 4 KBytes. The maximum size of a text data block is then 4 096 Bytes. This maximum size permits internal scrolling capabilities within the Text-Videotex Output Field text content in 40 or 80 columns mode.

For Videotex data content, and to ensure a minimum VEMMI service, a VEMMI terminal shall also provide a minimum size of storage for one block of Videotex content. However, as blocks of Videotex data are only limited by their physical size (full DDA), their encoded data can be exceptionally greater than the minimum requested block storage size specified hereafter. An error message is, however, provided to inform the VEMMI application when a terminal is unable to store a block exceptionally greater than the minimum requested block storage size specified.

For one block of Videotex data content the minimum requested block storage size is the following:

- 4 KByte for an alphamosaic terminal;
- 8 KByte for an alphamosaic and geometric terminal;
- 16 KByte for a photographic terminal supporting the Photographic data syntax profile 1;
- 32 KByte for photographic terminal supporting the Photographic data syntax profile 2;
- 64 KByte for a photographic terminal supporting the Photographic data syntax profiles 3,4 and 5.

For photographic terminals the above values permit the reception of a full DDA photographic block encoded with the ISO 10918-1 [16] picture coding algorithm, up to an average ratio of 1,5 bit/pixel.

### **Attributes:**

The Text-Videotex Output Field component carries the following attributes, which are related to the associated text or Videotex data block. These attributes are used both, in the Presentation Box description for an initial text or Videotex data block, and in the VEMMI\_Additional\_Data commands for subsequent text or Videotex data blocks.

**BlockIndex:** This attribute carries the index of the transmitted data block. The consecutive data blocks of the Presentation Box shall be numbered consecutively.

The BlockIndex, possibly associated with the Presentation Box description, is not necessarily the block number 1 (VEMMI application choice). When any BlockIndex number is received as initial block, all the blocks with a lower BlockIndex value shall exist in the VEMMI application.

**Text:** This attribute carries the text content of the component.

**UseDef:** The values of this Boolean attribute carry following meanings:

true: the Videotex data that shall be displayed shall be a concatenation of the data stored in the DefVTX attribute of the Presentation Box and the data content of the current block. First, the data of DefVTX attribute shall be displayed. Immediately after that, without reset of Videotex decoder, the data content of the current block shall be displayed;

false: only the data content of the current block shall be displayed.

**VTX:** This attribute carries the Videotex content of a component. It may be optional if the attribute UseDef is set to true. In this case only the default VTX content is displayed.

**MoreBlocks:** The values of this Boolean attribute carry following meanings:

true: one or more data blocks are still available on a terminal request for a data block with a higher Block Identification Number;

false: this block is the last data block of the Presentation Box.

### 7.6.1.2 The Push Button component

The need to associate some controls to the presentation of text and Videotex data in a Presentation Box, or to stop the presentation of this data, induces the introduction of Push Button components, with text and Videotex label capabilities.

#### **Description, Behaviour, Interactive functionalities, Visual aspect, Attributes.**

See subclause 7.5.1.5.

### 7.6.1.3 The Text Input Area component

To offer character input in direct relation with the text or Videotex output data, a Text Input Area component is defined as transparently overlapping the data in the Text-Videotex Output Field component.

**Description:** The Text Input Area component is a rectangular area, non-materialised over text or Videotex data in the Text-Videotex Output Field. In this area the VEMMI application permits text input from the user. The start position of a Text Input Area is defined by the row and the character position coordinate of the data block in which it is defined. The horizontal dimension is defined by the number of Videotex equivalent character positions in a row. The vertical dimension is defined by the number of rows. A Text Input Area shall not overlap more than one data block.

The type of the input data can be specified by the application. The following types are available:

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



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

A Text Input Area does not have a label.

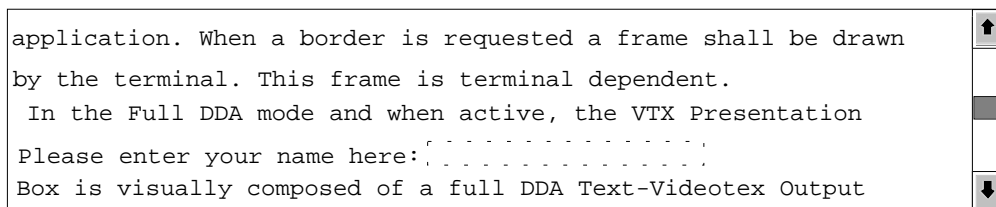
The cursor position of a Text Input Area (XCharPos, YCharPos) is defined by the row and the character position co-ordinate of the data block in which it is defined, called later on the associated data block.

**Behaviour:** The maximum number of characters that can be entered by the user shall correspond to number of characters which is reserved for the presentation of the component. If the maximum number of input characters is reached this should be indicated to the user. Scrolling shall not be used in Text Input Areas. An empty string is a valid input for a text input area.

**Interactive functionalities:**

- activation;
- validation;
- local editing functionalities of the input area (terminal dependent).

**Visual aspect (see figure 39 ):**



**Figure 39: Text Input Area**

**Attributes:**

The Text Input Area component carries the following attributes, which are related to the associated text or videotex data block. These attributes are used both, in the Presentation Box description for an initial text or videotex data block, and in the VEMMI\_Additional\_Data commands for subsequent text or videotex data blocks.

**CIN:** This attribute carries the Component Identification Number. The CIN are usually numbered as normal components with respect to the Presentation Box, and not with respect to their associated data block.

**XCharPos:** This attribute carries the horizontal cursor position of the components start, referring to the origin of the associated data block.

**YCharPos:** This attribute carries the vertical cursor position of the components start, referring to the origin of the associated data block.

**Width:** This attribute carries the width of the component.

**Height:** This attribute carries the height of the component.

**Opened:** The values of this Boolean attribute carry following meanings:

- true: the element shall be in the opened state;
- false: the element shall be in the closed state.

<b>Accessible:</b>	The values of this Boolean attribute carry following meanings:  true: the element can be accessed;  false: the element cannot be accessed.
<b>DefValue:</b>	This attribute carries the default value for the component.
<b>InputType:</b>	This attribute specifies which type of user inputs shall be accepted by the component.
<b>Echo:</b>	This attribute specifies which type of echo shall be made on user inputs.
<b>EchoChar:</b>	This attribute carries the character that shall be displayed to echo user inputs.
<b>LocActAct:</b>	This attribute carries the code for the local action which is associated to the component and is triggered by its activation.
<b>LocActVal:</b>	This attribute carries the code for the local action which is associated to the component and is triggered by its validation.

#### 7.6.1.4 The Sensitive Area component

**Positioning, Dimensioning:** The start position of a Sensitive Area is defined by the row and the character position co-ordinate of the data block in which it is defined, called later on associated data block. The horizontal dimension is defined by the number of character positions in a row. The vertical dimension is defined by the number of rows. A Sensitive Area shall not overlap more than one data block.

#### **Description, Behaviour, Interactive functionalities, Visual aspect:**

see subclause 7.5.1.11

#### **Attributes:**

The Sensitive Area component carries the following attributes, which are related to the associated text or Videotex data block. These attributes are used both, in the Presentation Box description for an initial text or Videotex data block, and in the VEMMI\_Additional\_Data commands for subsequent text or Videotex data blocks.

<b>CIN:</b>	This attribute carries the Component Identification Number. The CIN are usually numbered as normal components with respect to the Presentation Box, and not with respect to their associated data block.
<b>XCharPos:</b>	This attribute carries the horizontal cursor position of the components start, referring to the origin of the associated data block.
<b>YCharPos:</b>	This attribute carries the vertical cursor position of the components start, referring to the origin of the associated data block.
<b>Width:</b>	This attribute carries the width of the component.
<b>Height:</b>	This attribute carries the height of the component.
<b>Opened:</b>	The values of this Boolean attribute carry following meanings:  true: the element shall be in the opened state;  false: the element shall be in the closed state.

- Accessible:** The values of this Boolean attribute carry following meanings:
- true: the element can be accessed;
  - false: the element cannot be accessed.
- LocActAct:** This attribute carries the code for the local action which is associated to the component and is triggered by its activation.
- LocActVal:** This attribute carries the code for the local action which is associated to the component and is triggered by its validation.

#### 7.6.1.5 The Locator component

**Positioning, Dimensioning:** A Locator is always associated to one data block called later on the associated data block. The character positions that are reported by a Locator shall refer to the origin of the associated data block.

**Description, Behaviour, Interactive functionalities, Visual aspect:**

see subclause 7.5.1.12.

**Attributes:** The Locator component carries the following attributes, which are related to the associated text or videotex data block. These attributes are used both, in the Presentation Box description for an initial text or videotex data block, and in the VEMMI\_Additional\_Data commands for subsequent text or videotex data blocks.

- CIN:** This attribute carries the Component Identification Number. The CINs are usually numbered as normal components with respect to the Presentation Box, and not with respect to their associated data block.
- Opened:** The values of this Boolean attribute carry following meanings:
- true: the element shall be in the opened state;
  - false: the element shall be in the closed state.
- Accessible:** The values of this Boolean attribute carry following meanings:
- true: the element can be accessed;
  - false: the element cannot be accessed.
- LocActAct:** This attribute carries the code for the local action which is associated to the component and is triggered by its activation.
- LocActVal:** This attribute carries the code for the local action which is associated to the component and is triggered by its validation.

#### 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:

- general message;
- information message;
- warning message;
- action message.

This information can be used to add an icon characterising 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 one character position. 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 lifetime of a Message Box. The following five values are available:

- destroy by any user interaction;
- close by any user interaction;
- destroy by the user validation of an implicitly defined button;
- close by the user validation of an implicitly defined button;
- no implicit lifetime is defined (close and destroy are handled by the VEMMI application).

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 label and the size of the possible Push Button are terminal dependent.

**Interactive functionalities:**

- moving;
- validation.

**Visual aspect (see figure 40):**

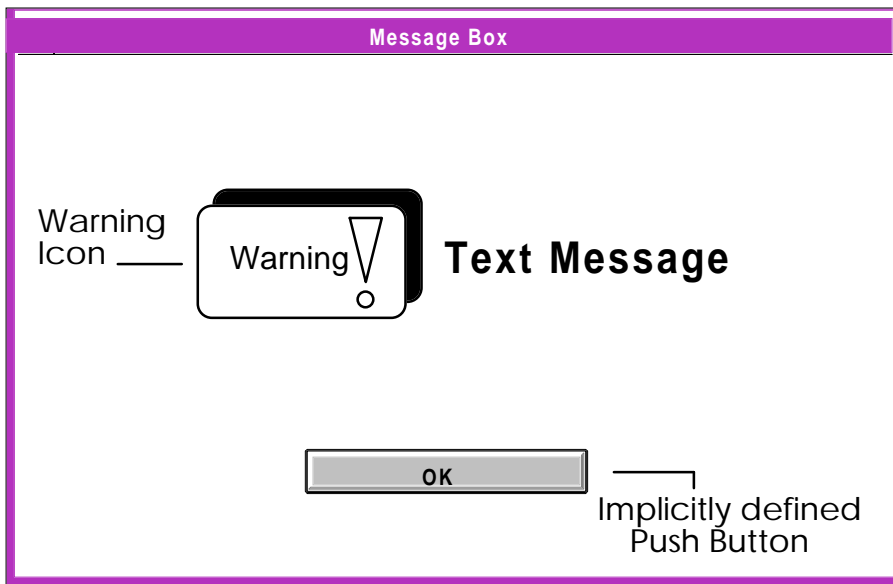


Figure 40: Message Box with a warning message

**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 component.

**Height:** This attribute carries the height of the component.

**Opened:** The values of this Boolean attribute carry following meanings:

true: the element shall be in the opened state;

false: the element shall be in the closed state.

**Border:** The values of this Boolean attribute carry following meanings:

true: a border shall be drawn by the terminal to frame the object. One character position should be reserved for the drawing of the border;

false: 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.

**MessageType:** This attribute specifies the type of message. It can be used to add a specific icon to the presentation of the Message Box.

**Text:** This attribute carries the text message that shall be displayed in the Message Box.

**Modal:** The values of this Boolean attribute carry following meanings:

true: the object shall be modal;

false: the object shall not be modal.

**Opened:** The values of this Boolean attribute carry following meanings:

true: the element shall be in the opened state;

false: the element shall be in the closed state.

**Lifetime:** This attribute specifies the which event results the destruction of the object.

**Sound:** The values of this Boolean attribute carry following meanings:

true: the terminal should perform a sound when the object is opened;

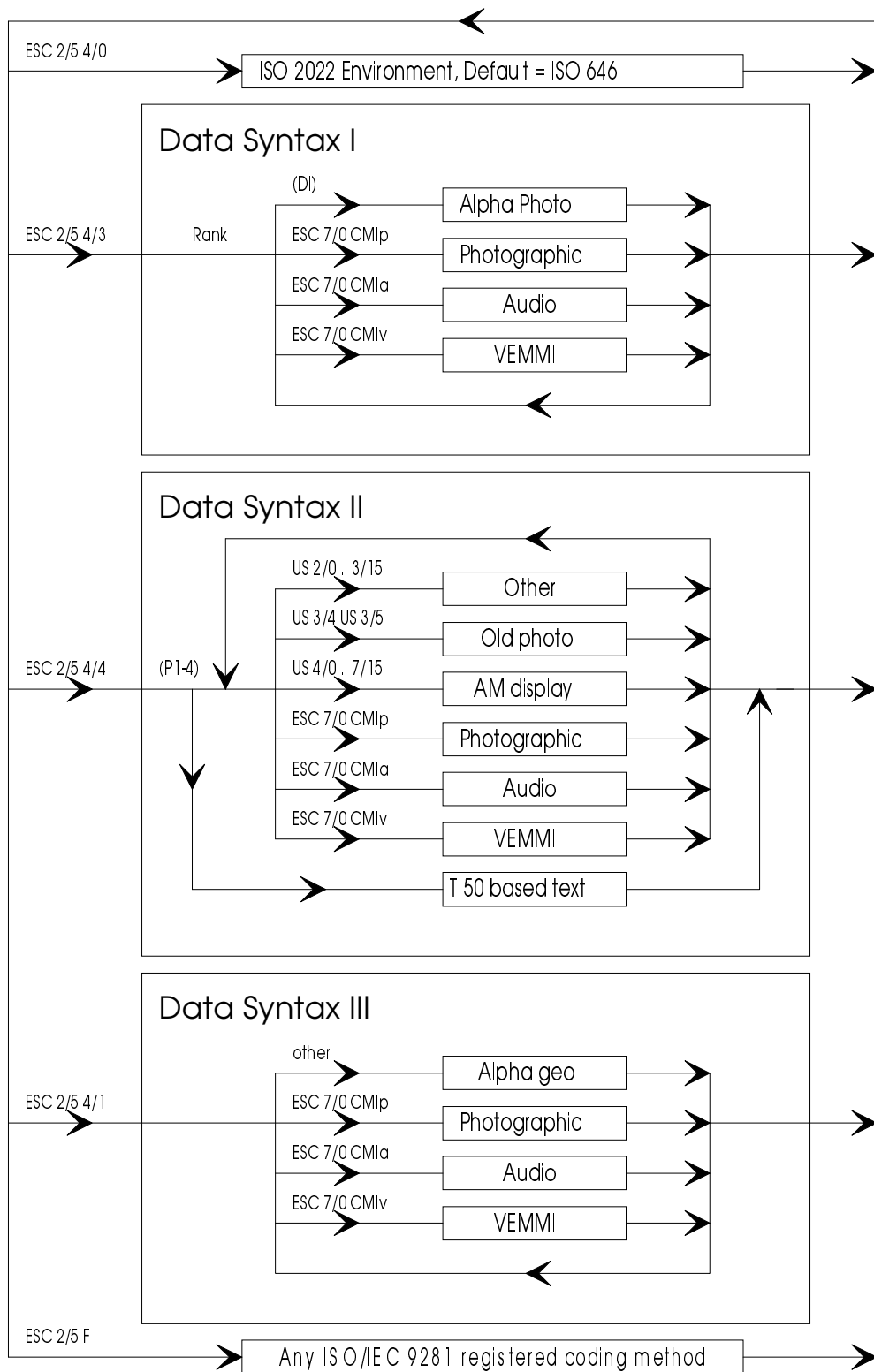
false: the terminal should not perform a sound when the object is opened.

## **8 Coding of the service elements**

### **8.1 Overall switching of coding environment**

ISO/IEC 9281 [12] describes a technique for identifying coding methods. The Videotex VEMMI mode is one of the coding methods identified by ISO/IEC 9281 [12]. The diagram in figure 41 gives an overview of the relationship between the Videotex data syntaxes and ISO/IEC 9281 [12] coding environments.

From an ISO 2022 [14] 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 [14] environment into an ISO/IEC 9281 [12] environment. The F code ("final byte") is allocated and registered, according to ISO 2022 [14] by the registration authority ISO 2375 [15]. According to ISO 2375 [15], Appendix B, the Videotex data syntaxes are regarded as "coding systems different to that of ISO 2022 [14]". 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.



**Figure 41: Global switching mechanism**

Key to figure 41:

- DI is data syntax I specific
- P is a profile in data syntax II
- F is a final code assigned by the ISO 2022 [14] registration authority
- CMI<sub>a</sub> is any CMI for videotex audio data
- CMI<sub>p</sub> is any CMI for videotex photographic data
- CMI<sub>v</sub> is any CMI for VEMMI data
- 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.

## 8.2 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 [14] can enter the ISO/IEC 9281 [12] environment of the VEMMI mode according to their own rules. In the case of Videotex for switching into the ISO/IEC 9281 [12] environment of the VEMMI mode the Picture Coding Delimiter (PCD) of the first Picture Element (PE) is used. The Coding Method Identifier (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 [12] was developed especially for the identification of picture coding in ISO 2022 [14] environments, the word "picture" is often used in the definitions, even when applicable to e.g. "audio". ISO has already accepted to make use of ISO/IEC 9281 [12] for non-pictorial information.

## 8.3 ISO/IEC 9281 syntax structure

The high level format of the syntax is as defined in ISO/IEC 9281 [12].

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/IEC 9281 [12] is also valid in a 7 bit environment. In this case the word "byte" shall be interpreted as meaning "7 bit byte" and the most significant bit, bit 8, shall not be used.

The structure of the coding is as follows:

```
PE ::= PCE PDE
PCE ::= PCD CMI LI
PCD ::= 01/11 07/00
CMI ::= PM PI
PM ::= 02/05 (Videotex VEMMI mode)
PI ::= 04/00
LI1 ::= x000 0000
      | x111 1111 <byte1> <byte2> ... <byten>
<bytek> ::= 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 42) 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 [12], subclause 5.2.7 should be consulted for the description of the Length Indicator.



PE Picture Entity				
PCE Picture Control Entity				PDE
PCD Picture Coding Delimiter	CMI Coding Method Identifier		LI Length Indicator	
	PM Picture Mode	PI Picture Identifier		Picture Data Entity

**Figure 42: Structure of a Picture Entity**

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 Picture Control Entity (PCE) consists of a Picture Coding Delimiter (PCD) and a Coding Method Identifier (CMI) followed by a Length Indicator (LI).

The Picture Coding Delimiter (PCD) is a fixed sequence of two octets: 01/11 07/00.

NOTE 2: 01/11 is ESC.

The Coding Method Identifier (CMI) consists of a Picture Mode (PM) octet, followed by a Picture Identifier octet (PI). For VEMMI, the PM is 02/05 as registered by ISO/IEC 9281 [12]. The PI octet has the value 04/00.

#### 8.4 Coding of the Picture Data Entity (PDE)

The PDE contains the VEMMI commands with or without VEMMI data.

The VEMMI commands are divided into four different classes:

- a) object specific commands (Application→Terminal);
- b) general commands (Application→Terminal);
- c) terminal commands (Terminal→ Application);
- d) error message (Terminal→ Application).

The structure of the different classes is described in the following subclauses.

**8.5 Object specific commands**

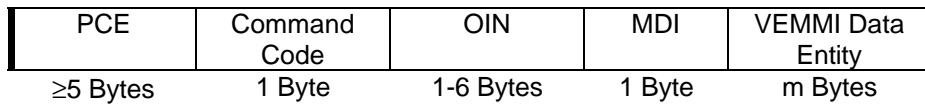
The following VEMMI commands issued by the VEMMI application are specific to VEMMI objects:

**Table 23: Object specific commands**

VEMMI Command	Command Code	VEMMI Data	Coding subclause
VEMMI_Create_Object	2/0	yes	9.1
VEMMI_Open_Object	2/1	no	
VEMMI_Close_Object	2/2	no	
VEMMI_Destroy_Object	2/3	no	
VEMMI_Obj_Access_Disable	2/4	no	
VEMMI_Obj_Access_Enable	2/5	no	
VEMMI_Additional_Data	2/6	yes	9.3
VEMMI_Modify_Component	2/7	yes	9.2
VEMMI_Obj_Location_Change	2/8	yes	9.4

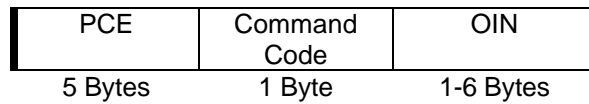
NOTE: The column "Coding subclause" references the subclause where the coding of the VEMMI data is defined.

Object specific commands which carry VEMMI data (see table 23) shall be encoded in one or more PEs using following structure:



**Figure 43: Coding structure for object specific commands that carry VEMMI data**

Object specific commands which do not carry VEMMI data (see table 23) shall be encoded in one PE using following structure:



**Figure 44: Coding structure for object specific commands that do not carry VEMMI data**

The coding of the different fields of the PDE is specified in subclause 8.9.

**8.6 General commands**

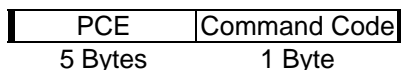
The following VEMMI commands issued by the VEMMI application are general:

**Table 24: General commands**

VEMMI Command	Command Code	VEMMI Data	Coding subclause
VEMMI_On	4/0	no	
VEMMI_Off	4/1	no	
VEMMI_Close_All	4/2	no	
VEMMI_User_Lock	4/3	no	
VEMMI_User_Unlock	4/4	no	
VEMMI_Reset	4/5	no	

NOTE: The column "Coding subclause" references the subclause where the coding of the VEMMI data is defined.

General commands shall be encoded in one PE using following structure:



**Figure 45: Coding structure for general commands**

The coding of the different fields of this PDE is specified in subclause 8.9.

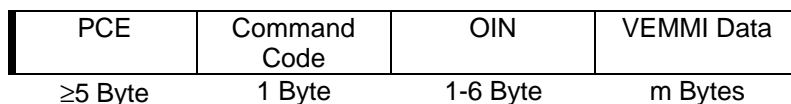
**8.7 Terminal commands**

The following VEMMI commands can be issued by the VEMMI Terminal:

**Table 25: Terminal commands**

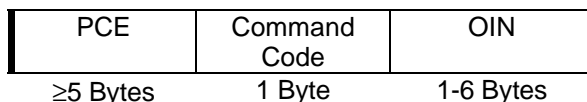
VEMMI Command	Command Code	VEMMI Data	Coding subclause
VEMMI_Object_Retransmission	5/0	no	
VEMMI_More_Data	5/1	yes	9.5
VEMMI_User_Data	5/2	yes	9.6
NOTE: The column "Coding subclause" references the subclause where the coding of the VEMMI data is defined.			

Terminal commands which carry VEMMI data (see table 25) shall be encoded in PE using following structure:



**Figure 46: Coding structure for terminal commands that carry VEMMI data**

Terminal commands which do not carry VEMMI data (see table 25) shall be encoded in one PE using following structure:



**Figure 47: Coding structure for terminal specific commands that do not carry VEMMI data**

The coding of the different fields of this PDE is specified in subclause 8.9.

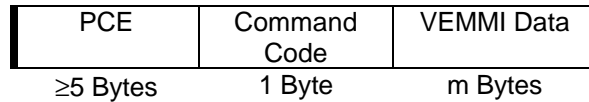
**8.8 Error Message**

The following VEMMI error message can be issued by the VEMMI Terminal:

**Table 26: Error Messages**

VEMMI Command	Command Code	VEMMI Data	Coding subclause
VEMMI_Error	6/0	yes	9.7
NOTE: The column "Coding subclause" references the subclause where the coding of the VEMMI data is defined.			

A VEMMI\_Error shall be code shall be encoded in one PE using following structure:



**Figure 48: Coding structure for error message commands**

The coding of the different fields of this PDE is specified in subclause 8.9.

## 8.9 Coding of the VEMMI command fields

### 8.9.1 Command Code

This subclause summarises the codes to identify the different VEMMI commands (see table 27).

**Table 27: VEMMI command codes**

VEMMI Command	Command Code	VEMMI Data	Coding subclause
VEMMI_Create_Object	2/0	yes	9.1
VEMMI_Open_Object	2/1	no	
VEMMI_Close_Object	2/2	no	
VEMMI_Destroy_Object	2/3	no	
VEMMI_Obj_Access_Disable	2/4	no	
VEMMI_Obj_Access_Enable	2/5	no	
VEMMI_Additional_Data	2/6	yes	9.3
VEMMI_Modify_Component	2/7	yes	9.2
VEMMI_Obj_Location_Change	2/8	yes	9.4
Reserved for future object specific commands	2/9-3/F		
VEMMI_On	4/0	no	
VEMMI_Off	4/1	no	
VEMMI_Close_All	4/2	no	
VEMMI_User_Lock	4/3	no	
VEMMI_User_Unlock	4/4	no	
VEMMI_Reset	4/5	no	
Reserved for future general commands	4/6-4/F		
VEMMI_Object_Retransmission	5/0	no	
VEMMI_More_Data	5/1	yes	9.5
VEMMI_User_Data	5/2	yes	9.6
Reserved for future terminal commands	5/3-5/F		
VEMMI_Error	6/0	yes	9.7
Reserved for future error messages	6/1-6/2		
Reserved for the commands to support local storage	6/3-6/7		annex B (informative)
Reserved for the commands to support operative objects	6/8-6/9		annex B (informative)
Reserved for the commands to support colour tables	6/A-6/B		annex B (informative)
Reserved for the commands to support sets of objects	6/C-7/2		annex B (informative)
Reserved for future use	7/3-7/F		
NOTE: The column "Coding subclause" references the subclause where the coding of the VEMMI data is defined.			

### 8.9.2 Object Identification Number (OIN)

The OIN shall be coded in the following way:

bit 8 = not used;

bit 7 = always set to one;

bit 6 = extension flag;

bits 5 to 1 = specify the OIN.

Bit 6 of each byte shall be the extension flag. The value shall be specified in binary notation as an unsigned integer using bits 5 to 1 with the weights  $2^4$ ,  $2^3$ ,  $2^2$ ,  $2^1$  and  $2^0$ , respectively. If the value 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 is larger than 31 it shall be represented by more than one byte. The most significant part of this value shall be contained in the first recorded byte. The extension flag shall be set to ONE in all bytes except the last, where it shall be set to ZERO.

The length of the OIN code is limited to 6 bytes. The value of the extension flag of byte 6 shall be set to ZERO.

### 8.9.3 More Data Indicator (MDI)

The MDI shall be coded in one byte. When VEMMI data corresponding to one VEMMI command are split into several PEs, all PEs shall be sent with the MDI value 2/1, except the final one. The last PE shall be sent with the MDI value 2/0.

Table 28 shows the possible values of the MDI:

**Table 28: Coding structure**

MDI	Coding
Last PE	2/0
More PEs to follow	2/1

### 8.9.4 User data

If present, this field contains VEMMI data, or VEMMI data entities, encoded corresponding to the selected VEMMI command. The subclauses which specify the coding of VEMMI data corresponding to the different VEMMI commands are referenced in subclause 8.9.1, table 27.

## 9 Coding of the VEMMI data

### 9.1 Structure of the VEMMI data of a VEMMI\_Create\_Object command

This subclause specifies the structure that shall be used to encode the VEMMI data of the VEMMI\_Create\_Object command. The VEMMI data of the VEMMI\_Create\_Object command shall be sent to the terminal in one or more VEMMI data entities.

The VEMMI data of the VEMMI\_Create\_Object command shall be encoded using the following structure (given by figure 49):

Obj. Def. Block	::=	Object Type	Length	Object Definition
Comp. Def. Block 1	::=	Component 1Type	Length	Component Definition
Comp. Def. Block 2	::=	Component 2Type	Length	Component Definition
Comp. Def. Block 3	::=	Component 3Type	Length	Component Definition
⋮	⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮	⋮
Comp. Def. Block n	::=	Component n Type	Length	Component Definition

**Figure 49: Coding structure of the VEMMI data of the VEMMI\_Create\_Object command**

The following items, a) to h), contain the coding rules which shall be observed when coding the VEMMI data of a VEMMI\_Create\_Object command:

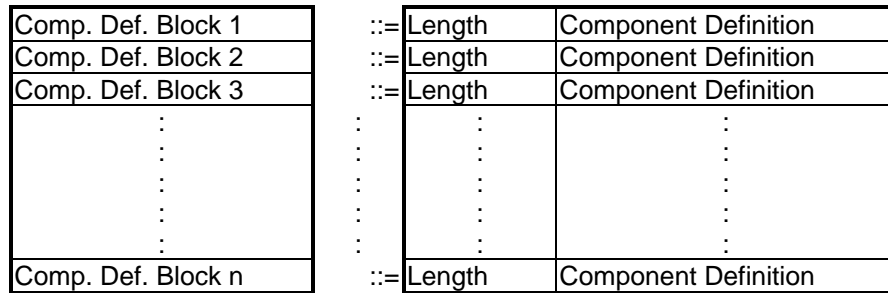
- a) the VEMMI data of the VEMMI\_Create\_Object command shall consist of one object definition block, optionally followed by one or more component definition blocks;
- b) for the objects Application Bar and Pop-Up Menu the order of appearance of the component definition blocks within the VEMMI data of the VEMMI\_Create\_Object command shall correspond to the canonical descending order of the components (see subclause 7.2 and subclause 7.4);
- c) each object definition block shall consist of a object type field, a "Length" field and a object definition field;
- d) each component definition block shall consist of a component type field, a "Length" field and a component definition field;
- e) the "Object-Component Type" field shall be encoded in one byte, using the codes given in subclause 9.9, table 34, to designate a specific object or component;
- f) the "Length" field shall encode the length of the following object or component definition field. It shall be encoded as defined in subclause 9.8, item j);
- g) the "Object-Component Definition" field shall contain all attribute codes necessary to define the specific object or component. The attribute codes which are available for every object or component are defined in subclause 9.10. The coding rules for encoding the attribute codes are defined in subclause 9.8. The appearance of the attribute codes within the "Object-Component Definition" field shall correspond to the order defined in subclause 9.10;
- h) if optional attribute codes are not present within an "Object-Component Definition" field the local manager shall apply the default values.

**9.2 Structure of the VEMMI data of a VEMMI\_Modify\_Component command**

This subclause specifies the structure that shall be used to encode the VEMMI data of the VEMMI\_Modify\_Component command.

The VEMMI data the VEMMI data of the VEMMI\_Modify\_Component command shall be sent to the terminal in one or more VEMMI data entities.

The VEMMI data of the VEMMI\_Modify\_Component command shall be encoded using the following structure (given in figure 50).



**Figure 50: Coding structure of the VEMMI data of the VEMMI\_Modify\_Component command**

The following items, a) to f), contain the coding rules which shall be observed when coding the VEMMI data of a VEMMI\_Modify\_Component command:

- a) the VEMMI data of the VEMMI\_Modify\_Component command shall consist of one or more component definition blocks;
- b) each component definition block shall consist of a "Length" field and a "Component Definition" field;
- c) the "Length" field shall encode the length of the following "Component Definition" field. It shall be encoded as defined in subclause 9.8, item j);
- d) the "Component Definition" field shall contain all attribute codes necessary to define the specific component. The attribute codes which are available for every component are defined in subclause 9.10. The coding rules for encoding the attribute codes are defined in subclause 9.8. The appearance of the attribute codes within the "Component Definition" field shall correspond to the order defined in subclause 9.10;
- e) only attributes which are designated as modifiable (see component attribute tables in subclause 9.10) shall be present within a "Component Definition" field;
- f) if optional attribute codes are not present within an "Component Definition" field the local manager shall not change their current values.

### 9.3 Structure of the VEMMI data of a VEMMI\_Additional\_Data command

This subclause specifies the structure that shall be used to encode the VEMMI data of the VEMMI\_Additional\_Data command.

The VEMMI data of the VEMMI\_Additional\_Data command shall be sent to the terminal in one or more VEMMI data entities.

The VEMMI data of the VEMMI\_Additional\_Data command shall be encoded using the following structure (given in figure 51):

Comp. Def. Block 1	::=	TextVTXOutputField	Length	Component Definition
Comp. Def. Block 2	::=	Component 2 Type	Length	Component Definition
Comp. Def. Block 3	::=	Component 3 Type	Length	Component Definition
⋮	⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮	⋮
⋮	⋮	⋮	⋮	⋮
Comp. Def. Block n	::=	Component n Type	Length	Component Definition

**Figure 51: Coding structure of the VEMMI data of the VEMMI\_Additional\_Data command**

The following items, a) to h), contain the coding rules which shall be observed when coding the VEMMI data of a VEMMI\_Additional\_Data command:

- a) the VEMMI data of the VEMMI\_Additional\_Data command shall consist of one or more component definition blocks;
- b) the VEMMI data of the VEMMI\_Additional\_Data command shall start with Text Videotex Output Field definition block. Only one Text Videotex Output Field shall be defined in the VEMMI data of one VEMMI\_Additional\_Data command;
- c) only component definition blocks for the following components shall be present:
  - Videotex Output Field;
  - Text Input Field;
  - Sensitive Area;
  - Locator.
- d) each component definition block shall consist of a "Component Type" field, a length field and a component description field;
- e) the "Component Type" field shall be encoded in one byte, using the codes given in subclause 9.9, table 34, to designate a specific component;
- f) the "Length" field shall encode the length of the following "Component Definition" field. It shall be encoded as defined in subclause 9.8, item j);
- g) the "Component Definition" field shall contain all attribute codes necessary to define the specific component. The attribute codes which are available for every component are defined in subclause 9.10. The coding rules for encoding the attribute codes are defined in subclause 9.8. The appearance of the attribute codes within the "Component Definition" field shall correspond to the order defined in subclause 9.10;
- h) if optional attribute codes are not present within an "Object-Component Definition" field the local manager shall apply the default values.



#### 9.4 Structure of the VEMMI data of a VEMMI\_Obj\_Location\_Change command

This subclause specifies the structure that shall be used to encode the VEMMI data of the VEMMI\_Obj\_Location\_Change command.

The VEMMI data of the VEMMI\_Obj\_Location\_Change command shall be sent to the terminal in one or more VEMMI data entities.

The VEMMI data of the VEMMI\_Obj\_Location\_Change command shall be encoded using the structure shown in figure 52:

<XPos> <YPos>
---------------

**Figure 52: Coding structure of the VEMMI data of the VEMMI\_Obj\_Location\_Change command**

The following items, a) and b), contain the coding rules which shall be observed when coding the VEMMI data of a VEMMI\_Obj\_Location\_Change command:

- a) the VEMMI data of the VEMMI\_Obj\_Location\_Change command shall consist of two attribute codes. The first attribute code shall represent the new horizontal position of the object. The second attribute code represents the new vertical position of the object;
- b) the coding rules for encoding the attribute codes are defined in subclause 9.8.

#### 9.5 Structure of the VEMMI data of a VEMMI\_More\_Data command

This subclause specifies the structure that shall be used to encode the VEMMI data of the VEMMI\_More\_Data command.

The VEMMI data of the VEMMI\_More\_Data command shall be encoded using the structure shown in figure 53:

<BlockIndex>
--------------

**Figure 53: Coding structure of the VEMMI data of the VEMMI\_More\_Data command**

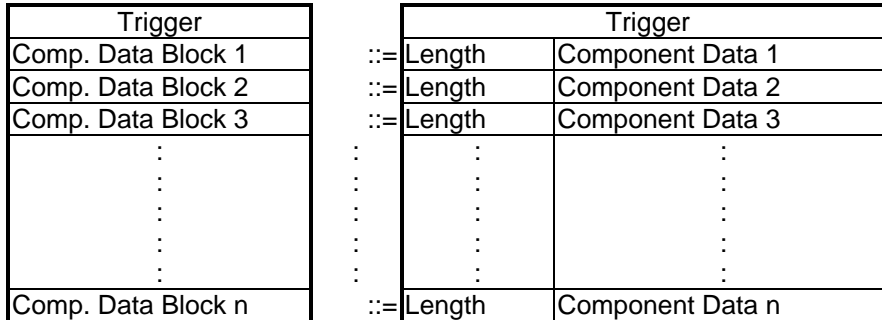
The following items, a) and b), contain the coding rules which shall be observed when coding the VEMMI data of a VEMMI\_More\_Data command:

- a) the VEMMI data of the VEMMI\_More\_Data command shall consist of one attribute code representing the block number of the block the VEMMI application should transmit to the VEMMI terminal;
- b) the coding rule for encoding the attribute code is defined in subclause 9.8.

**9.6 Structure of the VEMMI data of a VEMMI\_User\_Data command**

This subclause specifies the structure that shall be used to encode the VEMMI data of the VEMMI\_User\_Data command.

The VEMMI data of the VEMMI\_User\_Data command shall be encoded using the structure shown in figure 54:



**Figure 54: Coding structure of the VEMMI data of the VEMMI\_User\_Data command**

The following items, a) to d), contain the coding rules which shall be observed when coding the VEMMI data of a VEMMI\_User\_Data command:

- a) the VEMMI data of the VEMMI\_User\_Data command shall consist of one byte specifying the trigger event that induced the report command followed by one or more component data blocks;
- b) the value of the byte shall be 2/0 if the report was triggered on the activation of the component and 2/1 if the report was triggered on the validation of the component;
- c) the first component in the VEMMI data of the VEMMI\_User\_Data command shall be the component which induced the report action;
- d) the component data block shall contain all attribute codes necessary to report the value of the specific component. The attribute codes which are available to report the value of the specific components are defined in subclause 9.6.1. The coding rules for encoding the attribute codes are defined in subclause 9.8 The appearance of the attribute codes within the component data field shall correspond to the order defined in subclause 9.6.1.

**9.6.1 Coding of the report values for VEMMI components**

**9.6.1.1 Menu Choice Bar, Menu Choice Pull-Down, Menu Choice Cascading, Menu Choice Pop-Up, Push Button, Button Sensitive Area**

Component data ::=	<b>&lt;CIN&gt;</b>
--------------------	--------------------

**<CIN>**: Component Identification Number.

**Table 29**

<b>Component:</b> Menu Choice Bar, Menu Choice Pull-Down, Menu Choice Cascading, Menu Choice Pop-Up, Push Button, Button Sensitive Area	
<b>Attribute Code Identifier</b>	<b>Type</b>
<b>&lt;CIN&gt;</b>	integer

**9.6.1.2 Text Input Field, List Box, Combination Box**

Component data ::= <CIN> (UserText)

<CIN>: Component Identification Number.

(UserText): Current content of a Text Input Field or currently marked List items.

**Table 30**

<b>Component: Text Input Field, Text Input Area List Box, Combination Box</b>	
<b>Attribute Code Identifier</b>	<b>Type</b>
<CIN>	integer
(UserText)	string

**9.6.1.3 Check Box, Radio Button**

Component data ::= <CIN> [UserBoolean]

<CIN>: Component Identification Number.

[UserBoolean]: Current value of the component.

**Table 31**

<b>Component: Check Box, Radio Button</b>	
<b>Attribute Code Identifier</b>	<b>Type</b>
<CIN>	integer
[UserBoolean]	Boolean

**9.6.1.3.1 Locator**

Component data ::= <CIN> <XCharPos> <YCharPos>

<CIN>: Component Identification Number.

<XCharPos>: horizontal character position.

<YCharPos>: vertical character position.

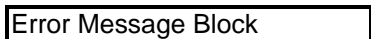
**Table 32**

<b>Component Locator</b>	
<b>Attribute Code Identifier</b>	<b>Type</b>
<CIN>	integer
<XCharPos>	integer
<YCharPos>	integer

**9.7 Structure of the VEMMI data of a VEMMI\_Error command**

This subclause specifies the structure that shall be used to encode the VEMMI data of the VEMMI\_Error command.

The VEMMI data of the VEMMI\_Error command shall be encoded using the general structure shown in figure 55:



**Figure 55: General coding structure of the VEMMI data of the VEMMI\_Error command**

The following items, a) to b), contain the coding rules which shall be observed when coding the VEMMI data of a VEMMI\_Error command:

- a) the VEMMI data of the VEMMI\_Error command shall consist of one error message block;
- b) the error message block shall contain all attribute codes necessary to specify the error. The attribute codes which are available to specify the error are defined in subclause 9.7.1. The coding rules for encoding the attribute codes are defined in subclause 9.8. The appearance of the attribute codes within the error message block shall correspond to the order defined in subclause 9.7.1.

**9.7.1 Coding of the error message**

ErrorMessage ::=	<ErrorType> [OINId] [CIN]
------------------	---------------------------

<ErrorType>: enumeration:

0:	general error
1:	unknown VEMMI command
2:	erroneous VEMMI command
3:	unexpected VEMMI command
4:	object syntax error
5:	out of memory
6:	service not supported

[OINId]: Object Identification Number of the object the element that caused the error. The OIN shall be present if the value of the error type code is 4 or 5.

[CIN]: Component Identification Number of the component the element specific action is referring to. The CIN may only be present if the OIN attribute is present as well.

**Table 33**

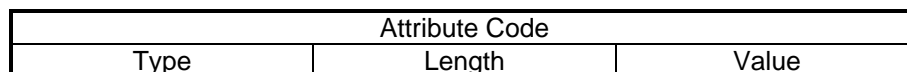
Error Message	
Attribute Code Identifier	Type
<ErrorType>	enum
[OINId]	integer
[CIN]	integer

## 9.8 General rules for coding VEMMI data

Coding rules shall be applied to the coding of all VEMMI elements and their attributes. A 7 bit coding technique shall be used to facilitate the implementation of the VEMMI data syntax in a 7 or an 8 bit environment. In general, a byte will be used to encode the 7 bit codes, bit 8 is, however, never used for coding purposes.

An attribute code shall be represented using three fields: **type**, **length** and **value**. The **attribute field type**, shall be encoded using one 7 bit code which indicates the data type of the field value (integer, string ..) as well as the type of attribute encoded (vertical position, width, title..). The **attribute field length**, consists of a number of 7 bit codes containing the length (number) of 7 bits codes used by the **attribute field value**. The **attribute field value**, consists of a number of 7 bit codes. All coding techniques defined in this ETS use bit 7 set to one in order to prevent codes in the range from 0/0 to 1/F.

Figure 56 shows the structure of an attribute code.



**Figure 56: Structure of an attribute code**

The following items, a) to s), contain the coding rules which shall be observed when coding the attributes of the VEMMI elements.

- a) in a 7 bit environment, bit 8 of a byte shall be set to zero, only 7 of the 8 bits are used for coding purposes;
- b) the sequence of the attribute codes is significant. All attribute codes shall appear in the sequence defined. Only the attribute codes which define the local actions (ReportAct, GeneralAct, ElementSpecificAct) can appear in any possible order. The order of appearance in the code shall then be equal to the order of their execution by the local manager;
- c) "<attr>" means the attribute code shall appear one time within the definition block of the element;
- d) "[attr]" means the attribute code shall appear zero or one time within the definition block of the element. If the code does not appear within the creation command of an element the default value for the attribute is applied. If the code does not appear within a modify command of an element the value for the element is not changed;
- e) "(attr)" means that the attribute code or the group of attribute codes shall appear zero, one or more than once, within the definition block of the element;
- f) "|" means that the sender shall make a choice which attribute code or group of attribute codes shall be sent. In the case of a group of attributes all members of the group of attributes shall always be sent;
- g) an attribute code consists of a **type field**, a **length field** and a **value field**;
- h) the **attribute field type**, shall be coded by one byte from column 3 of table 61 in subclause 9.11;
- j) the **attribute field length** shall be coded in the following way:

The length indicating the number of subsequent bytes shall have the following structure, where:

bit 8 = not used;

bit 7 = always set to one;

bit 6 = extension flag;

bits 5 to 1 = specify the number of bytes used to encode the attribute value.

Bit 6 of each byte shall be the extension flag. The value shall be specified in binary notation as an unsigned integer using bits 5 to 1 with the weights  $2^4$ ,  $2^3$ ,  $2^2$ ,  $2^1$  and  $2^0$ , respectively. If the value 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 is larger than 31 it shall be represented by more than one byte. The most significant part of this value shall be contained in the byte recorded first. The extension flag shall be set to ONE in all bytes except the last where it shall be set to ZERO.

The nil value of an attribute shall be encoded by a length only, setting all significant bits to ZERO;

- k) the **attribute field value**, shall be encoded using a number of bytes, the number shall be indicated in the **attribute field length**;
- l) the following attribute data types shall be defined: integer, normalised, enumeration, Boolean, string and VTX;
- m) encoding of an integer value:
  - bit 7 shall always be set to one;
  - the byte or bytes of the **attribute field value**, shall be equal to an integer value, consisting of bits 5 to 1 of the first byte, followed by bits 6 to 1 of the second byte, followed by bits 6 to 1 of each byte in turn up to and including the last byte of the **attribute field value**. Bit 7 of all bytes of the **attribute field value** shall be set to one and shall not be used for coding purposes;
  - the value of the integer binary number shall be derived by numbering the bits of the bytes, N representing this number, in the **attribute field value**, starting with bit 1 of the last byte as bit one (N=1) and ending the numbering with bit 5 of the first byte;
  - each bit is assigned a numerical value  $2^{N-1}$ , where N is its position in the above numbering sequence. The integer value shall be obtained by summing the numerical values assigned to each bit for those bits which are set to one, excluding bit 6 of the first byte;
  - negative values are represented by the two's complement notation, which uses bit 6 of the first byte as a sign bit (0/1 positive/negative values).

According to the above rules the integer values + 6 837 and - 6 837 shall be coded:

EXAMPLE 1: Coded representation of a positive integer number + 6 837.

byte 1 is X1000001

	X	1	Sign	N=17	N=16	N=15	N=14	N=13
	X	1	0	0	0	0	0	1
$2^{N-1}$	X	X	+	0	0	0	0	4 096

byte 2 is X1101010

	X	1	N=12	N=11	N=10	N=9	N=8	N=7
	X	1	1	0	1	0	1	0
$2^{N-1}$	X	X	2 048	0	512	0	128	0

byte 3 is X1110101

	X	1	N=6	N=5	N=4	N=3	N=2	N=1
	X	1	1	1	0	1	0	1
$2^{N-1}$	X	X	32	16	0	4	0	1

+ 6 837 = 4 096 + 2 048 + 512 + 128 + 32 + 16 + 4 + 1.

EXAMPLE 2: Coded representation of a negative integer number - 6 837.

byte 1	byte 2	byte 3	
X 1 1 1 1 1 1 0	X 1 0 1 0 1 0 1	X 1 0 0 1 0 1 1	X = bit 8 and is not used, bit 7 is always set to 1.

n) Encoding of a normalised value (NDC co-ordinates):

- the range of normalised values shall be from - 1 to + 1, inclusive;

NOTE: In this ETS, the term "normalised" is not used in its classical sense because it includes + 1 or - 1.

- bit 7 shall always be set to one and shall not be used for coding purposes;
- if bit 5 of the first byte is set to one, all other bits shall be set to zero with the exception of the sign bit, bit 6;
- the bytes of the **attribute field value**, shall be equal to a normalised value, consisting of bit 5 to 1 of the first byte, followed by bits 6 to 1 of the second byte, followed by bits 6 to 1 of each byte in turn up to and including the last byte of the attribute field, value;
- the value of the normalised binary number shall be derived by numbering the bits of the bytes in the **attribute field value**, starting with bit 5 of the first byte as bit one (N=1) and ending the numbering with bit 1 of the last byte;
- each bit shall be assigned a numerical value  $1/(2^{N-1})$ , where N is its position in the above numbering sequence. The normalised value is obtained by summing the numerical fractions assigned to each bit for those bits which are set to one, excluding bit 6 of the first byte;
- negative values shall be represented by the two's complement, which uses bit 6 of the first byte as a sign bit (0/1 positive/negative values).

EXAMPLE 3: Coding of a positive normalised number + 0,58984375.

byte 1	byte 2	byte 3
X 1 0 0 1 0 0 1	X 1 0 1 1 1 0 0	X 1 0 0 0 0 1 0

X = not used; bit 7 is always set to one.

EXAMPLE 4: Coding of a negative normalised number - 0,58984375.

byte 1	byte 2	byte 3
X 1 1 1 0 1 1 0	X 1 1 0 0 0 1 1	X 1 1 1 1 1 1 0

X = not used, bit 7 is always set to one.

p) Encoding of an enumerated value;

- an enumerated value shall be encoded with an integer. The value of the integer indicates which value is assigned to an attribute.

- q) Encoding of a Boolean value:
- bit 7 shall always be set to one and shall not be used for coding purposes;
  - if the Boolean value is "FALSE" the value of the byte shall be zero;
  - if the Boolean value is "TRUE" the value of the byte shall be any non zero number (bit 8 is not used).
- r) Encoding of a string value.
- the bytes of the **attribute field value**, shall be the codes of the 7 bit environment of ITU-T Recommendation T.51 [9].

EXAMPLE 5: Coding of a string "ETSI"

byte 1	byte 2	byte 3	byte 4
X 1 0 0 0 1 0 1	X 1 0 1 0 1 0 0	X 1 0 1 0 0 1 1	X 1 0 0 1 0 0 1
E	T	S	I

X = not used.

- s) Encoding of a VTX value.
- the bytes of the **attribute field value**, shall be the codes of the corresponding Videotex ETSS referenced in clause 2.



9.9 Code assignments for VEMMI objects and components

Table 34: Code assignment for object and component definition blocks

Object	Component	Object Code	Component Code	
Application Bar		2/0		
	Menu Choice Bar		3/0	
	Menu Choice Pull-Down		3/1	
	Menu Choice Cascading		3/2	
	Menu Choice Separator		3/3	
Button Bar		2/1		
	Button		3/4	
Pop-Up Menu		2/2		
	Menu Choice Pop-Up		3/5	
	Menu Choice Cascading		3/6	
	Menu Choice Separator		3/7	
Dialogue Box		2/3		
	Separator		3/8	
	Frame		3/9	
	Text Presentation Area		3/A	
	VTX Presentation Area		3/B	
	Push Button		3/C	
	Text Input Field		3/D	
	Check Box		3/E	
	Radio Button		3/F	
	List Box		4/0	
	Combination Box		4/1	
	Sensitive Area		4/2	
	Locator		4/3	
	Presentation Box		2/5	
		Text-Videotex Output Field		4/4
Push Button			4/5	
Text Input Area			4/6	
Sensitive Area			4/7	
Locator			4/8	
Message Box		2/6		

9.10 Coding of VEMMI elements

The following subclauses define the coding of the object and component definition blocks used in the VEMMI data entities of the VEMMI\_Create\_Object, the VEMMI\_Modify\_Component and the VEMMI\_Additional\_Data command.

9.10.1 Application Bar

Object Definition ::=	[Horizontal] [Position] [Size] [FirstActive] [Opened] [Accessible]
[Position] ::=	[XPos]   [YPos]
[Size] ::=	[Width]   [Height]

[Horizontal]: Boolean: true: horizontal Bar;  
false: vertical Bar.

[XPos]: horizontal position. This attribute shall only be present at horizontal Application Bars.

[YPos]: vertical position. This attribute shall only be present at vertical Application Bars.

- [Width]: number of columns for vertical Bars. It shall only be present at vertical Bars. All Menu Choices in a vertical Application Bar shall have the same number of columns.
- [Height]: number of rows for horizontal Bars. It shall only be present at horizontal Bars. All Menu Choices in a horizontal Application Bar shall have the same number of rows.
- [FirstActive]: CIN of the Bar component that is activated by default the first time the object is opened.
- [Opened]: Boolean: true: opened state;  
false: closed state.
- [Accessible]: Boolean: true: accessible state parameter;  
false: inaccessible state parameter.

**Table 35**

<b>Object Application Bar</b>		
<b>Attribute Code Identifier</b>	<b>Type</b>	<b>Default</b>
[Horizontal]	Boolean	true
[XPos]	NDC	0
[YPos]	NDC	0
[Width]	integer	1
[Height]	integer	1
[FirstActive]	integer	note
[Opened]	Boolean	true
[Accessible]	Boolean	true
NOTE: By default the first defined opened accessible component shall be the active one.		

**9.10.1.1 Menu Choice Bar, Menu Choice Pull-Down, Menu Choice Cascading, Menu Choice Pop-Up components**

The structure of the Application Bar is given in subclause 7.2. The coding for Menu Choice Bar, Menu Choice Pull-Down, Menu Choice Cascading, Menu Choice Pop-Up components is the same.

Comp. Definition ::=	<CIN> [Width] [Height] [Accessible] [Content] [Mnemonic] [LocActAct] [LocActVal]
[Content] ::=	[Text]   [VTX]

- <CIN>: Component Identification Number.
- [Width]: number of columns allocated for the component. It may only be present at components belonging to:
  - horizontal Bars;
  - Pull-Down Menus;
  - Cascading Menus.
- [Height]: number of rows allocated for the component. It may only be present at menu components belonging to:
  - vertical Bars;
  - Pull-Down Menus;
  - Primary Pop-Up Menus;
  - Cascading Menus.
- [Accessible]: Boolean: true: accessible state parameter;  
false: inaccessible state parameter.

- [Text]: text content.
- [Mnemonic]: character may be entered by the user to validate the component.
- [VTX]: Videotex content.
- [LocActAct]: local action code for component activation (see subclause 9.10.7).
- [LocActVal]: local action code for component validation (see subclause 9.10.7).

**Table 36**

<b>Component: Menu Choice Bar, Menu Choice Pull-Down, Menu Choice Cascading, Menu Choice Pop-Up</b>			
Attribute Code Identifier	Type	Default	Modifiable
<CIN>	integer		no
[Width]	integer	note	no
[Height]	integer	note	no
[Accessible]	Boolean	true	yes
[Text]	string	nil	yes
[Mnemonic]	char	nil	yes
[VTX]	VTX	nil	yes
[LocActAct]	see subclause 9.10.7		yes
[LocActVal]	see subclause 9.10.7		yes
NOTE:	The default value for dimensions of the component shall be equal to the last defined dimension values of a component in this object. The default values for the first component in an object are: Width=1, Height =1.		

**9.10.1.2 Menu Choice Separator component**

Comp. Definition ::=	<CIN>
----------------------	-------

<CIN>: Component Identification Number.

**Table 37**

<b>Component: Menu Choice Separator</b>			
Attribute Code Identifier	Type	Default	Modifiable
<CIN>	integer		no

**9.10.2 Button Bar**

Object Definition ::=	[XPos] [YPos] [Horizontal] [Size] [FirstActive] [Modal] [Opened] [Accessible]
[Size] ::=	[Width]   [Height]

[XPos], [YPos]: horizontal position, vertical position.

[Horizontal]: Boolean: true: horizontal Bar;  
false: vertical Bar.

[Width]: number of columns only for vertical Bars. It shall only be present at vertical Bars. All Buttons in a vertical Bar shall have the same number of columns.

[Height]: number of rows only for horizontal Bars. It shall only be present at horizontal Bars. All Buttons in a horizontal Bar shall have the same number of rows.

- [FirstActive]: CIN of the Button Bar component that is activated by default the first time the object is opened.
- [Modal]: Boolean: true: modal;  
false: non-modal.
- [Opened]: Boolean: true: opened state;  
false: closed state.
- [Accessible]: Boolean: true: accessible state parameter;  
false: inaccessible state parameter.

**Table 38**

Object Button Bar		
Attribute Code Identifier	Type	Default
[XPos]	NDC	0
[YPos]	NDC	0
[Width]	integer	1
[Height]	integer	1
[FirstActive]	integer	note
[Modal]	Boolean	false
[Opened]	Boolean	true
[Accessible]	Boolean	true
NOTE: By default the first defined opened accessible component shall be the active one.		

**9.10.2.1 Button**

Comp. Definition ::=	<CIN> [Size] [Accessible] [Content] [Mnemonic] [LocActAct] [LocActVal]
[Size] ::=	[Width]   [Height]
[Content] ::=	[Text]   [VTX]

- <CIN>: Component Identification Number.
- [Width]: number of columns allocated for the Button. It shall only be present at horizontal Button Bars.
- [Height]: number of rows allocated for the Button. It shall only be present at vertical Button Bars.
- [Accessible]: Boolean: true: accessible state parameter;  
false: inaccessible state parameter.
- [Text]: text content.
- [Mnemonic]: character may be entered by the user to validate the component.
- [VTX]: Videotex content.
- [LocActAct]: local action code for component activation (see subclause 9.10.7).
- [LocActVal]: local action code for component validation (see subclause 9.10.7).

Table 39

Component Button			
Attribute Code Identifier	Type	Default	Modifiable
<CIN>	integer		no
[Width]	integer	note	no
[Height]	integer	note	no
[Accessible]	Boolean	true	yes
[Text]	string	nil	yes
[Mnemonic]	char	nil	yes
[VTX]	VTX	nil	yes
[LocActAct]	see subclause 9.10.7		yes
[LocActVal]	see subclause 9.10.7		yes
NOTE:	The default value for dimensions of the Push Button shall be equal to the last defined dimension values of a Push Button in this object. The default values for the first component in an object are: Width=1, Height =1.		

### 9.10.3 Pop-Up Menu

Object Definition ::= [XPos] [YPos] [Width] [Title] [FirstActive] [Modal] [Opened] [Accessible]

- [XPos], [YPos]: horizontal position, vertical position.
- [Width]: number of columns for the Pop-Up Menu. All components shall have the same number of columns.
- [Title]: text title of the object.
- [FirstActive]: CIN of the Menu component that is activated by default the first time the object is opened.
- [Modal]: Boolean: true: modal;  
false: non-modal.
- [Opened]: Boolean: true: opened state;  
false: closed state.
- [Accessible]: Boolean: true: accessible state parameter;  
false: inaccessible state parameter.

Table 40

Object Pop-Up Menu		
Attribute Code Identifier	Type	Default
[XPos]	NDC	0
[YPos]	NDC	0
[Horizontal]	Boolean	true
[Width]	integer	1
[Title]	string	no title
[FirstActive]	integer	note
[Modal]	Boolean	false
[Opened]	Boolean	true
[Accessible]	Boolean	true
NOTE:	By default the first defined opened accessible component shall be the active one.	

**9.10.3.1 Menu Choice Pop-Up, Menu Choice Cascading components**

The structure of the Pop-Up Menu is given in subclause 7.4. The coding for Menu Choice Pop-Up, Menu Choice Cascading components is the same and defined in subclause 9.10.1.1.

**9.10.4 Dialogue Box**

Object Definition ::=	[XPos] [YPos] [Width] [Height] [Border] [Title] [FirstActive] [Modal] [Opened] [Accessible] [StoreInitialValues]
-----------------------	--

- [XPos], [YPos]: horizontal position, vertical position.
- [Width], [Height]: width and height.
- [Border]: Boolean: true: a frame shall be drawn around the Dialogue Box;  
false: no frame shall be drawn around the Dialogue Box.
- [Title]: text title of the object.
- [FirstActive]: CIN of the component that is activated by default, the first time the object is opened.
- [Modal]: Boolean: true: modal;  
false: non-modal.
- [Opened]: Boolean: true: opened state;  
false: closed state.
- [Accessible]: Boolean: true: accessible state parameter;  
false: inaccessible state parameter.
- [StoreInitialValues]: Boolean: true: the terminal shall store the initial values of the components;  
false: it is not mandatory for the terminal to store the initial values.

**Table 41**

Object Dialogue Box		
Attribute Code Identifier	Type	Default
[XPos]	NDC	0
[YPos]	NDC	0
[Width]	integer	40
[Height]	integer	24
[Border]	Boolean	true
[Title]	string	no title
[FirstActive]	integer	note
[Modal]	Boolean	true
[Opened]	Boolean	true
[Accessible]	Boolean	true
[StoreInitialValues]	Boolean	true
NOTE:	By default the first defined opened accessible component shall be the active one.	

9.10.4.1 Separator

Comp. Definition ::=	<CIN> [XPos] [YPos] [Horizontal] [Size] [Opened]
[Size] ::=	[Width]   [Height]

<CIN>: Component Identification Number.

[XPos], [YPos]: horizontal position, vertical position.

[Horizontal]: Boolean: true: horizontal;  
false: vertical.

[Width]: number of columns for horizontal Separator. It shall only be present at horizontal Separators.

[Height]: number of rows for vertical Separators. It shall only be present at vertical Separators.

[Opened]: Boolean: true: opened state;  
false: closed state.

Table 42

Component Separator			
Attribute Code Identifier	Type	Default	Modifiable
<CIN>	integer		no
[XPos]	NDC	0	no
[YPos]	NDC	0	no
[Horizontal]	Boolean	true	no
[Width]	integer	note 1	no
[Height]	integer	note 2	no
[Opened]	Boolean	true	yes
NOTE 1:	The default value for vertical dimension shall be the height of the Dialogue Box.		
NOTE 2:	The default value for horizontal dimension shall be the width of the Dialogue Box.		

9.10.4.2 Frame

Comp. Definition ::= <CIN> [XPos] [YPos] [Width] [Height] [Opened]
--

- <CIN>: Component Identification Number.
- [XPos], [YPos]: horizontal position, vertical position.
- [Width], [Height]: width and height.
- [Opened]: Boolean: true: opened state;  
false: closed state.

**Table 43**

<b>Component Frame</b>			
<b>Attribute Code Identifier</b>	<b>Type</b>	<b>Default</b>	<b>Modifiable</b>
<CIN>	integer		no
[XPos]	NDC	0	no
[YPos]	NDC	0	no
[Width]	integer	note 1	no
[Height]	integer	note 2	no
[Opened]	Boolean	true	yes
NOTE 1: The default value for horizontal dimension shall be the width of the Dialogue Box.			
NOTE 2: The default value for vertical dimension shall be the height of the Dialogue Box.			



9.10.4.3 Text Presentation Area

Comp. Definition ::=	<CIN> [XPos] [YPos] [Width] [Height] [Opened] [TextLabel] [Text]
----------------------	--

- <CIN>: Component Identification Number.
- [XPos], [YPos]: horizontal position, vertical position.
- [Width], [Height]: width and height.
- [Opened]: Boolean: true: opened state;  
false: closed state.
- [TextLabel]: text label.
- [Text]: text content.

Table 44

Component Text Presentation Area			
Attribute Code Identifier	Type	Default	Modifiable
<CIN>	integer		no
[XPos]	NDC	0	no
[YPos]	NDC	0	no
[Width]	integer	note 1	no
[Height]	integer	note 2	no
[Opened]	Boolean	true	yes
[TextLabel]	string	nil	yes
[Text]	string	nil	yes
NOTE 1:	The default value for horizontal dimension shall be the width of the Dialogue Box.		
NOTE 2:	The default value for vertical dimension shall be the height of the Dialogue Box.		

9.10.4.4 Videotex Presentation Area

Comp. Definition ::= <CIN> [XPos] [YPos] [Width] [Height] [Opened] [VTX]
--

- <CIN>: Component Identification Number.
- [XPos], [YPos]: horizontal position, vertical position.
- [Width], [Height]: width and height.
- [Opened]: Boolean: true: opened state;  
false: closed state.
- [VTX]: Videotex content.

Table 45

<b>Component Videotex Presentation Area</b>			
<b>Attribute Code Identifier</b>	<b>Type</b>	<b>Default</b>	<b>Modifiable</b>
<CIN>	integer		no
[XPos]	NDC	0	no
[YPos]	NDC	0	no
[Width]	integer	note 1	no
[Height]	integer	note 2	no
[Opened]	Boolean	true	yes
[VTX]	VTX	nil	yes
NOTE 1: The default value for horizontal dimension shall be the width of the Dialogue Box.			
NOTE 2: The default value for vertical dimension shall be the height of the Dialogue Box.			

9.10.4.5 Push Button

Comp. Definition ::=	<CIN> [XPos] [YPos] [Width] [Height] [Opened] [Accessible] [Content] [Mnemonic] [LocActAct] [LocActVal]
<Content> ::=	[Text]   [VTX]

- <CIN>: Component Identification Number.
- [XPos], [YPos]: horizontal position, vertical position.
- [Width], [Height]: width and height.
- [Opened]: Boolean: true: opened state;  
false: closed state.
- [Accessible]: Boolean: true: accessible state parameter;  
false: inaccessible state parameter.
- [Text]: text content.
- [Mnemonic]: character may be entered by the user to validate the component.
- [VTX]: Videotex content.
- [LocActAct]: local action code for component activation (see subclause 9.10.7).
- [LocActVal]: local action code for component validation (see subclause 9.10.7).

Table 46

Component Push Button			
Attribute Code Identifier	Type	Default	Modifiable
<CIN>	integer		no
[XPos]	NDC	0	no
[YPos]	NDC	0	no
[Width]	integer	note 1	no
[Height]	integer	note 2	no
[Opened]	Boolean	true	yes
[Accessible]	Boolean	true	yes
[Text]	string	nil	yes
[Mnemonic]	char	nil	yes
[VTX]	VTX	nil	yes
[LocActAct]	see subclause 9.10.7		yes
[LocActVal]	see subclause 9.10.7		yes
NOTE 1:	The default value for horizontal dimension shall be the width of the Dialogue Box.		
NOTE 2:	The default value for vertical dimension shall be the height of the Dialogue Box.		

#### 9.10.4.6 Text Input Field

Comp. Definition ::=	<CIN> [XPos] [YPos] [Width] [Height] [Opened] [Accessible] [TextLabel] [DefValue] [InputType] [Echo] [EchoChar] [MaxChar] [InputTransformation] [LocActAct] [LocActVal]
----------------------	---

- <CIN>: Component Identification Number.
- [XPos], [YPos]: horizontal position, vertical position.
- [Width], [Height]: width and height.
- [Opened]: Boolean: true: opened state;  
false: closed state.
- [Accessible]: Boolean: true: accessible state parameter;  
false: inaccessible state parameter.
- [TextLabel]: text label.
- [DefValue]: default value of the component at the first time the object is opened.
- [InputType]: enumeration: 0: any;  
1: numeric;  
2: alphabetic;  
3: alphanumeric.
- [Echo]: enumeration: 0: local echo with the characters entered by the user;  
1: no echo;  
2: echo with a character defined by the VEMMI application.
- [EchoChar]: character which is echoed by the local manager when the user enters a character in the Input Field and when the echo attribute value is 2.
- [MaxChar]: maximum number of characters the user can enter in a Text Input Field.
- [InputTransformation]: enumeration: 0: no transformation;  
1: the terminal should transform the input to upper case;  
2: the terminal should transform the input to lower case.
- [LocActAct]: local action code for component activation (see subclause 9.10.7).
- [LocActVal]: local action code for component validation (see subclause 9.10.7).

Table 47

Component Text Input Field			
Attribute Code Identifier	Type	Default	Modifiable
<CIN>	integer		no
[XPos]	NDC	0	no
[YPos]	NDC	0	no
[Width]	integer	note 1	no
[Height]	integer	note 2	no
[Opened]	Boolean	true	yes
[Accessible]	Boolean	true	yes
[TextLabel]	string	nil	yes
[DefValue]	string	nil	yes
[InputType]	enum.	0	yes
[Echo]	enum.	0	yes
[EchoChar]	char	"-"	yes
[MaxChar]	integer	1	yes
[InputTransformation]	enum.	0	yes
[LocActAct]	see subclause 9.10.7		yes
[LocActVal]	see subclause 9.10.7		yes
NOTE 1:	The default value for horizontal dimension shall be the width of the Dialogue Box.		
NOTE 2:	The default value for vertical dimension shall be the height of the Dialogue Box.		

9.10.4.7 Check Box

Comp. Definition ::=	<CIN> [XPos] [YPos] [Width] [Height] [Opened] [Accessible] [TextLabel] [DefMarked] [LocActAct] [LocActVal]
----------------------	--

- <CIN>: Component Identification Number.
- [XPos], [YPos]: horizontal position, vertical position.
- [Width], [Height]: width and height.
- [Opened]: Boolean: true: opened state;  
false: closed state.
- [Accessible]: Boolean: true: accessible state parameter;  
false: inaccessible state parameter.
- [TextLabel]: text label.
- [DefMarked]: Boolean: true: marked by default;  
false: unmarked by default.
- [LocActAct]: local action code for component activation (see subclause 9.10.7).
- [LocActVal]: local action code for component validation (see subclause 9.10.7).

Table 48

Component Check Box			
Attribute Code Identifier	Type	Default	Modifiable
<CIN>	integer		no
[XPos]	NDC	0	no
[YPos]	NDC	0	no
[Width]	integer	note 1	no
[Height]	integer	note 2	no
[Opened]	Boolean	true	yes
[Accessible]	Boolean	true	yes
[TextLabel]	string	nil	yes
[DefMarked]	Boolean	false	yes
[LocActAct]		see subclause 9.10.7	yes
[LocActVal]		see subclause 9.10.7	yes
NOTE 1:	The default value for horizontal dimension shall be the width of the Dialogue Box.		
NOTE 2:	The default value for vertical dimension shall be the height of the Dialogue Box.		

9.10.4.8 Radio Button

Comp. Definition ::=	<CIN> [XPos] [YPos] [Width] [Height] [Opened] [Accessible] [TextLabel] [DefMarked] [Group] [LocActAct] [LocActVal]
----------------------	--

- <CIN>: Component Identification Number.
- [XPos], [YPos]: horizontal position, vertical position.
- [Width], [Height]: width and height.
- [Opened]: Boolean: true: opened state;  
false: closed state.
- [Accessible]: Boolean: true: accessible state parameter;  
false: inaccessible state parameter.
- [TextLabel]: text label.
- [Group]: identifier of the Radio Button group. Only one Radio Button in a group can be marked at one time.
- [DefMarked]: Boolean: true: marked by default;  
false: unmarked by default.
- [LocActAct]: local action code for component activation (see subclause 9.10.7).
- [LocActVal]: local action code for component validation (see subclause 9.10.7).

Table 49

Component Radio Button			
Attribute Code Identifier	Type	Default	Modifiable
<CIN>	integer		no
[XPos]	NDC	0	no
[YPos]	NDC	0	no
[Width]	integer	note 1	no
[Height]	integer	note 2	no
[Opened]	Boolean	true	yes
[Accessible]	Boolean	true	yes
[TextLabel]	string	nil	yes
[Group]	integer	1	no
[DefMarked]	Boolean	false	yes
[LocActAct]	see subclause 9.10.7		yes
[LocActVal]	see subclause 9.10.7		yes
NOTE 1:	The default value for horizontal dimension shall be the width of the Dialogue Box.		
NOTE 2:	The default value for vertical dimension shall be the height of the Dialogue Box.		

9.10.4.9 List Box

Comp. Definition ::=	<CIN> [XPos] [YPos] [Width] [Height] [Opened] [Accessible] [DeflItem] (List Item) [SingleChoice] [LocActAct] [LocActVal]
(ListItem) ::=	[ListIndex] [ListText]

- <CIN>: Component Identification Number.
- [XPos], [YPos]: horizontal position, vertical position.
- [Width], [Height]: width and height.
- [Opened]: Boolean: true: opened state;  
false: closed state.
- [Accessible]: Boolean: true: accessible state parameter;  
false: inaccessible state parameter.
- [DeflItem]: index of the list item that is selected by default when the component is opened for the first time.
- [ListIndex]: index of the next list item in the component description.
- [ListText]: text of a list items of the List Box.
- [SingleChoice]: Boolean: true: single choice;  
false: multiple choice.
- [LocActAct]: local action code for component activation (see subclause 9.10.7).
- [LocActVal]: local action code for component validation (see subclause 9.10.7).

Table 50

Component List Box			
Attribute Code Identifier	Type	Default	Modifiable
<CIN>	integer		no
[XPos]	NDC	0	no
[YPos]	NDC	0	no
[Width]	integer	note 1	no
[Height]	integer	note 2	no
[Opened]	Boolean	true	yes
[Accessible]	Boolean	true	yes
[DeflItem]	integer	1	yes
[ListIndex]	integer	note 3	yes
[ListText]	string	nil	yes
[SingleChoice]	Boolean	true	no
[LocActAct]	see subclause 9.10.7		yes
[LocActVal]	see subclause 9.10.7		yes
NOTE 1:	The default value for horizontal dimension shall be the width of the Dialogue Box.		
NOTE 2:	The default value for vertical dimension shall be the height of the Dialogue Box.		
NOTE 3:	The default value for the first list item is 1. The default value for all following list item is the index of the previously defined list item incremented by one.		



9.10.4.10 Combination Box

Comp. Definition ::=	<CIN> [XPos] [YPos] [Width] [Height] [Opened] [Accessible] [DeflItem] (List Item) [DropDown] [EditableInput] [ConsistencyCheck] [MaxChar] [LocActAct] [LocActVal]
(ListItem):=	[ListIndex] [ListText]

<CIN>:	Component Identification Number.
[XPos], [YPos]:	horizontal position, vertical position.
[Width], [Height]:	width and height.
[Opened]:	Boolean: true: opened state; false: closed state.
[Accessible]:	Boolean: true: accessible state parameter; false: inaccessible state parameter.
[DeflItem]:	index of the list item that is selected by default when the component is opened for the first time.
[ListIndex]:	index of the next list item in the component description.
[ListText]:	text of a list items of the List Box.
[DropDown]:	Boolean: true: Drop Down Combination Box; false: regular Combination Box.
[EditableInput]:	Boolean: true: the text field shall be editable; false: the text field shall not be editable.
[MaxChar]:	maximum number of characters the user can enter in a Text Input Field.
[ConsistencyCheck]	Boolean: true: the text entered shall correspond to a list item; false: the user may enter any text string.
[LocActAct]:	local action code for component activation (see subclause 9.10.7).
[LocActVal]:	local action code for component validation (see subclause 9.10.7).

Table 51

Component Combination Box			
Attribute Code Identifier	Type	Default	Modifiable
<CIN>	integer		no
[XPos]	NDC	0	no
[YPos]	NDC	0	no
[Width]	integer	note 1	no
[Height]	integer	note 2	no
[Opened]	Boolean	true	yes
[Accessible]	Boolean	true	yes
[DefItem]	integer	1	yes
[ListIndex]	integer	note 3	yes
[ListText]	string	nil	yes
[DropDown]	Boolean	true	no
[EditableInput]	Boolean	true	yes
[MaxChar]	integer	1	yes
[ConsistencyCheck]	Boolean	true	yes
[LocActAct]	see subclause 9.10.7		yes
[LocActVal]	see subclause 9.10.7		yes
NOTE 1:	The default value for horizontal dimension shall be the width of the Dialogue Box.		
NOTE 2:	The default value for vertical dimension shall be the height of the Dialogue Box.		
NOTE 3:	The default value for the first list item is 1. The default value for all following list item is the index of the previously defined list item incremented by one.		

9.10.4.11 Sensitive Area

Comp. Definition ::=	<CIN> [AssCIN] [XCharPos] [YCharPos] [Width] [Height] [Opened] [Accessible] [LocActAct] [LocActVal]
----------------------	---

- <CIN>: Component Identification Number.
- [AssCIN]: Component Identification Number of the associated component.
- [XCharPos]: horizontal cursor position of the components start, referred to the origin of the associated component.
- [YCharPos]: vertical cursor position of the components start, referred to the origin of the associated component.
- [Width]: number of rows allocated for the component.
- [Height] : number of columns allocated for the component.
- [Opened]: Boolean: true: opened state;  
false: closed state.
- [Accessible]: Boolean: true: accessible state parameter;  
false: inaccessible state parameter.
- [LocActAct]: local action code for component activation (see subclause 9.10.7).
- [LocActVal]: local action code for component validation (see subclause 9.10.7).

Table 52

Component Sensitive Area			
Attribute Code Identifier	Type	Default	Modifiable
<CIN>	integer		no
[AssCIN]	integer	note	no
[XCharPos]	integer	0	no
[YCharPos]	integer	0	no
[Width]	integer	1	no
[Height]	integer	1	no
[Opened]	Boolean	true	yes
[Accessible]	Boolean	true	yes
[LocActAct]	see subclause 9.10.7		yes
[LocActVal]	see subclause 9.10.7		yes
NOTE:	First defined Text or Videotex Presentation Area.		

9.10.4.12 Locator

Comp. Definition ::=	<CIN> [AssCIN] [Opened] [Accessible] [LocActAct] [LocActVal]
----------------------	--

- <CIN>: Component Identification Number.
- [AssCIN]: Component Identification Number of the component to which the Locator is associated.
- [Opened]: Boolean: true: opened state;  
false: closed state.
- [Accessible]: Boolean: true: accessible state parameter;  
false: inaccessible state parameter.
- [LocActAct]: local action code for component activation (see subclause 9.10.7).
- [LocActVal]: local action code for component validation (see subclause 9.10.7).

Table 53

Component Locator			
Attribute Code Identifier	Type	Default	Modifiable
<CIN>	integer		no
[AssCIN]	integer	note	no
[Opened]	Boolean	true	yes
[Accessible]	Boolean	true	yes
[LocActAct]	see subclause 9.10.7		yes
[LocActVal]	see subclause 9.10.7		yes
NOTE:	First defined Text or Videotex Presentation Area.		

9.10.5 Presentation Box

Object Definition ::=	[XPos] [YPos] [Width] [Height] [ContentType] [ScrollingTools] [OutputHeight] [Border] [FirstActive] [DefVTX] [Modal] [Opened] [Accessible] [NumBlocks]
-----------------------	---

- [XPos], [YPos]: horizontal position, vertical position.
- [Width], [Height]: Width and Height.
- [ContentType]: enumeration: 0: Videotex;  
1: text.
- [ScrollingTools]: enumeration: 0: scrolling tools are terminal dependent;  
1: two columns of the Text-Videotex Output Field shall be reserved for the scrolling tools;  
2: two rows of the Text-Videotex Output Field shall be reserved for the scrolling tools.
- [OutputHeight]: height of the Text-Videotex Output Field.
- [Border]: Boolean: true: a frame shall be drawn around the Presentation Box;  
false: no frame shall be drawn around the Presentation Box.
- [FirstActive]: CIN of the component that is activated by default, the first time the object is opened.
- [DefVTX]: Videotex data which can be used as a template for the Videotex data blocks. This attribute shall may only be present at Presentation Boxes with Videotex content.
- [Modal]: Boolean: true: modal;  
false: non-modal.
- [Opened]: Boolean: true: opened state;  
false: closed state.
- [Accessible]: Boolean: true: accessible state parameter;  
false: inaccessible state parameter.
- [NumBlocks]: Number of Data blocks pertaining to this object (can be used for the presentation of scrolling tools).

Table 54

Object Presentation Box		
Attribute Code Identifier	Type	Default
[XPos]	NDC	0
[YPos]	NDC	0
[Width]	integer	40
[Height]	integer	24
[ContentType]	enum.	0
[ScrollingTools]	enum.	0
[OutputHeight]	integer	note 1
[Border]	Boolean	true
[FirstActive]	integer	note 2
[DefVTX]	VTX	nil
[Modal]	Boolean	true
[Opened]	Boolean	true
[Accessible]	Boolean	true
[NumBlocks]	integer	1
NOTE 1:	The default value shall be the height of the object reduced by two rows if the border attribute is set true.	
NOTE 2:	By default the first defined opened accessible component shall be the active one.	

**9.10.5.1 Text-Videotex Output Field**

Within one Presentation Box object only one Text-Videotex Output field definition shall be present.

Comp. Definition ::=	<BlockIndex> <Content> [MoreBlocks]
[Content] ::=	<Text>   [UseDef] <VTX>

<BlockIndex>: index of the data block.

<Text>: text content.

[UseDef]: Boolean: true: the data presented shall be a combination of the default data defined in the object and the Videotex content (DefVTX + VTX);  
false: no default data shall be used.

<VTX>: Videotex content.

[MoreBlocks]: Boolean: true: the data block is not the last block of the Presentation Box;  
false: the data block is the last block of the Presentation Box.

**Table 55**

<b>Component Text-Videotex Output Field</b>			
<b>Attribute Code Identifier</b>	<b>Type</b>	<b>Default</b>	<b>Modifiable</b>
<BlockIndex>	integer		no
<Text>	string	nil	no
[UseDef]	Boolean	true	no
<VTX>	VTX	nil	no
[MoreBlocks]	Boolean	true	no

**9.10.5.2 Push Button**

See subclause 9.10.4.5.

9.10.5.3 Text Input Area

Comp. Definition ::=	<CIN> [XCharPos] [YCharPos] [Width] [Height] [Opened] [Accessible] [DefValue] [InputType] [Echo] [EchoChar] [LocActAct] [LocActVal]
----------------------	---

- <CIN>: Component Identification Number.
- [XCharPos], [YCharPos]: cursor position of the components start, referred to the associated data block.
- [Width]: number of rows allocated for the component.
- [Height] : number of columns allocated for the component.
- [Opened]: Boolean: true: opened state;  
false: closed state.
- [Accessible]: Boolean: true: accessible state parameter;  
false: inaccessible state parameter.
- [DefValue]: default value of the component at the first time the object is opened.
- [InputType]: enumeration: 0: any;  
1: numeric;  
2: alphabetic;  
3: alphanumeric.
- [Echo]: enumeration: 0: local echo with the characters entered by the user;  
1: no echo;  
2: echo with a character defined by the VEMMI application.
- [EchoChar]: character which is echoed by the local manager when the user enters a character in the Input Area and when the echo attribute value is 2.
- [LocActAct]: local action code for component activation (see subclause 9.10.7).
- [LocActVal]: local action code for component validation (see subclause 9.10.7).

Table 56

Component Text Input Area			
Attribute Code Identifier	Type	Default	Modifiable
<CIN>	integer		no
[XCharPos]	integer	0	no
[YCharPos]	integer	0	no
[Width]	integer	1	no
[Height]	integer	1	no
[Opened]	Boolean	true	yes
[Accessible]	Boolean	true	yes
[DefValue]	string	nil	yes
[InputType]	enum.	0	yes
[Echo]	enum.	0	yes
[EchoChar]	char	"_"	yes
[LocActAct]	see subclause 9.10.7		yes
[LocActVal]	see subclause 9.10.7		yes



9.10.5.4 Sensitive Area

Comp. Definition ::=	<CIN> [XCharPos] [YCharPos] [Width] [Height] [Opened] [Accessible] [LocActAct] [LocActVal]
----------------------	---

- <CIN>: Component Identification Number.
- [XCharPos]: horizontal cursor position of the components start, referring to the origin of the data block.
- [YCharPos]: vertical cursor position of the components start, referring to the origin of the data block.
- [Width]: number of rows allocated for the component.
- [Height] : number of columns allocated for the component.
- [Opened]: Boolean: true: opened state;  
false: closed state.
- [Accessible]: Boolean: true: accessible state parameter;  
false: inaccessible state parameter.
- [LocActAct]: local action code for component activation (see subclause 9.10.7).
- [LocActVal]: local action code for component validation (see subclause 9.10.7).

Table 57

Component Sensitive Area			
Attribute Code Identifier	Type	Default	Modifiable
<CIN>	integer		no
[XCharPos]	integer	0	no
[YCharPos]	integer	0	no
[Width]	integer	1	no
[Height]	integer	1	no
[Opened]	Boolean	true	yes
[Accessible]	Boolean	true	yes
[LocActAct]	see subclause 9.10.7		yes
[LocActVal]	see subclause 9.10.7		yes

9.10.5.5 Locator

Comp. Definition ::=	<CIN> [Opened] [Accessible] [LocActAct] [LocActVal]
----------------------	---

- <CIN>: Component Identification Number.
- [Opened]: Boolean: true: opened state;  
false: closed state.
- [Accessible]: Boolean: true: accessible state parameter;  
false: inaccessible state parameter.
- [LocActAct]: local action code for component activation (see subclause 9.10.7).
- [LocActVal]: local action code for component validation (see subclause 9.10.7).

Table 58

Component Locator			
Attribute Code Identifier	Type	Default	Modifiable
<CIN>	integer		no
[Opened]	Boolean	true	yes
[Accessible]	Boolean	true	yes
[LocActAct]	see subclause 9.10.7		yes
[LocActVal]	see subclause 9.10.7		yes

9.10.6 Message Box

Object Definition ::=	[XPos] [YPos] [Width] [Height] [Opened] [Border] [Title] [MessageType] [Text] [Modal] [Opened] [Lifetime] [Sound]
-----------------------	--

- [XPos], [YPos]: horizontal position, vertical position.
- [Width], [Height]: width and height.
- [Opened]: Boolean: true: opened state;  
false: closed state.
- [Border]: Boolean: true: a frame shall be drawn around the Message Box;  
false: no frame shall be drawn around the Message Box.
- [Title]: text title.
- [MessageType]: enumeration: 0: general message;  
1: information message;  
2: warning message;  
3: action message.
- [Text]: message which shall be displayed.
- [Modal]: Boolean: true: modal;  
false: non-modal.
- [Opened]: Boolean: true: opened state;  
false: closed state.
- [Lifetime]: enumeration 0: destroy by any user interaction;  
1: close by any user interaction;  
2: destroy by a user validation of an implicitly defined button;  
3: close by a user validation of an implicitly defined button;  
4: no implicit lifetime defined. Changes of the state shall be induced by the application with the corresponding VEMMI commands.
- [Sound]: Boolean: true: a sound should be performed at the time the object is opened;  
false: no sound should be performed at the time the object is opened.

Table 59

Object Message Box		
Attribute Code Identifier	Type	Default
[XPos]	NDC	note
[YPos]	NDC	note
[Width]	integer	note
[Height]	integer	note
[Opened]	Boolean	true
[Border]	Boolean	true
[Title]	string	no title
[MessageType]	enum	0
[Text]	string	nil
[Modal]	Boolean	false
[Opened]	Boolean	true
[Accessible]	Boolean	true
[Lifetime]	enum	0
[Sound]	Boolean	true
NOTE:	The local manager shall calculate the appropriate size and centre the Message Box.	

### 9.10.7 Coding of local actions

The local action code specified in this subclause shall be the **value field** of an attribute code. The **type field** of the attribute code shall specify the trigger event for the local action.

The attribute codes which define the local actions (ReportAct, GeneralAct, ElementSpecificAct) can appear in the local action code in any possible order. The order of appearance in the code shall be equal to the order of their execution by the local manager.

A list of element specific actions may be defined within one local action. When a list of element specific actions is executed by the terminal, each action shall be completed before the execution of the next.

LocalAction ::=	[ReportAct] [GeneralAct] (<ElementSpecificAct> <ElementId>)
<ElementId> ::=	<OINId>   <CIN>

[ReportAct]: enumeration: 0: report CIN of the component;  
1: report the current value of the component (see note 1);  
2: report the values of all components in the object (see note 1);  
3-20: Reserved for future extensions;  
21-: Reserved for private use.

NOTE 1: This command only applies to input components.

[GeneralAct] enumeration: 0: User Lock;  
1: Restore initial values of the components of the object (see note 2);  
2-20: Reserved for future extensions;  
21-: Reserved for private use.

NOTE 2: This command should only be applied if the object was created with the "StoreInitialValue" attribute set true.

<ElementSpecificAct>: enumeration: 0: open component of parent object;  
1: close component of parent object;  
2: open object;  
3: close object;  
4: change component state parameter to inaccessible;  
5: change component state parameter to accessible;  
6: destroy object;  
7-20: Reserved for future extensions;  
21-: Reserved for private use.

<OINId>: Object Identification Number of the object the element specific action is referring to. This attribute shall only be present if the value of the element specific action code is 2, 3 or 6.

<CIN>: Component Identification Number of the component the element specific action is referring to. This attribute shall only be present if the value of the element specific action code is 0, 1, 4 or 5.

Table 60

Local Action	
Attribute Code Identifier	Type
[ReportAct]	enum
[GeneralAct]	enum
<ElementSpecificAct>	enum
<OINId>	integer
<CIN>	integer

9.11 Attribute field type codes

Table 61

Attribute Code Identifier	Type	Code
AssCIN	integer	2/0
BlockIndex	integer	2/1
CIN	integer	2/2
DefItem	integer	2/3
FirstActive	integer	2/4
Group	integer	2/5
Height	integer	2/6
ListIndex	integer	2/7
MaxChar	integer	2/8
NumBlocks	integer	2/9
OINId	integer	2/A
OutputHeight	integer	2/B
Width	integer	2/C
XCharPos	integer	2/D
YCharPos	integer	2/E
Accessible	Boolean	2/F
Border	Boolean	3/0
ConsistencyCheck	Boolean	3/1
DefMarked	Boolean	3/2
DropDown	Boolean	3/3
EditableInput	Boolean	3/4
Horizontal	Boolean	3/5
Modal	Boolean	3/6
MoreBlocks	Boolean	3/7
Opened	Boolean	3/8
SingleChoice	Boolean	3/9
Sound	Boolean	3/A
StoreInitialValues	Boolean	3/B
UseDef	Boolean	3/C
UserBoolean	Boolean	3/D
DefValue	string	3/E
EchoChar	string	3/F
ListText	string	4/0
Mnemonic	string	4/1
Text	string	4/2
TextLabel	string	4/3
Title	string	4/4
UserText	string	4/5
ContentType	enumeration	4/6
Echo	enumeration	4/7
ElementSpecificAct	enumeration	4/8
ErrorType	enumeration	4/9
GeneralAct	enumeration	4/A
InputTransformation	enumeration	4/B
InputType	enumeration	4/C
Lifetime	enumeration	4/D
MessageType	enumeration	4/E
ReportAct	enumeration	4/F
ScrollingTools	enumeration	5/0
DefVTX	VTX	5/1

(continued)

**Table 61 (concluded)**

<b>Attribute Code Identifier</b>	<b>Type</b>	<b>Code</b>
VTX	VTX	5/2
XPos	NDC	5/3
YPos	NDC	5/4
LocActAct	Local Action	5/5
LocActVal	Local Action	5/6
Reserved for future data types		5/7 - 6/4
Reserved for future extensions		6/5-7/F

## **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.

### **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.



**Annex A (normative): T.51String**

**A.1 Introduction**

This annex describes the rules to be applied when encoding names, strings etc. using ITU-T Recommendation T.51 [9]. The T.51String defined by this annex serves as a reference model by restricting ITU-T Recommendation T.51 [9] 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 A.1) is identical with the set of graphic characters of the International Reference Version (IRV) of the 7-bit coded character set of ITU-T Recommendation T.50 [8].

				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 T.51String

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	Κ	
0	0	0	1	1			ı	±	`	¹	Æ	æ	
0	0	1	0	2			ç	²	´	®	Ð	ð	
0	0	1	1	3			£	³	^	©	à	ä	
0	1	0	0	4				x	~	™	Ŧ	ħ	
0	1	0	1	5			¥	μ	—	♪		ı	
0	1	1	0	6				¶	˘	¬	ıı	ij	
0	1	1	1	7			§	•	•	ı	Ł	ł	
1	0	0	0	8			α	÷	••		κ	κ	
1	0	0	1	9			ˆ	˚			Ø	ø	
1	0	1	0	10			“	”	°		Œ	œ	
1	0	1	1	11			«	»	˘		o	ß	
1	1	0	0	12			←	¼		⅛	þ	þ	
1	1	0	1	13			↑	½	”	⅜	ƒ	ƒ	
1	1	1	0	14			→	¾	˘	⅝	ŋ	ŋ	
1	1	1	1	15			↓	¿	˘	⅞	'n	SHY	

 These codepositions shall not be used

**Figure A.2: Supplementary set of graphic characters for T.51String**

For the T.51String the following applies:

- 1) The primary set is designated as the G0 set and invoked in column 2 to 7 of the code table.
- 2) The supplementary set is designated as a G2 set. In the 8-bit environment the set is invoked in column 10 to 15.
- 3) In the T.51String no designation and invocation sequences are allowed.
- 4) All characters in column 4 of the supplementary set are non-spacing characters (diacritical marks).
- 5) Unallocated code positions are reserved and shall not be used.
- 6) The compatibility provisions described in ITU-T Recommendation T.51 [9] §2.2.4, Notes 3, 4 and 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 T.51String repertoire is identical with the super-set of the repertoire of the latin based character set specified in ITU-T Recommendation T.51 [9], annex A. All combinations of diacritical marks with basic letters as specified in Figure A-2 of ITU-T Recommendation T.51 [9] are supported.

The notes and remarks from ITU-T Recommendation T.51 [9], annex A, §A.4 shall not apply.

### A.5 Control functions

Invocation and designation sequences for control functions are not allowed. 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 ITU-T Recommendation T.50 [8].

## Annex B (informative): Future VEMMI concepts

### B.1 Local object storage

A host application has the possibility to request the load of local, in the terminal stored objects. 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. Three attributes characterise an object set: **application identifier**, **additional application data**, **time stamp**. The objects set attributes are stored together with the objects and are not changed by the terminal application. They are reused for forthcoming application identifications.

A stored set of objects contains the following:

- the attributes: application identifier, additional application data, time stamp. At least the application identifier and one of the two others shall be present;
- the objects including their components;
- the relevant Bitmaps, attributed fonts;
- the referenced files;
- the relevant loaded colour table.

The storage format is terminal specific because the objects are loaded and stored only by the same terminal application.

The following subclauses present the VEMMI commands that may be used to implement local object storage.

### B.1.1 VEMMI\_OpenApplication

VEMMI\_OpenApplication is used by the VEMMI application to load a set of objects in the terminal.

#### Parameters:

AppId: String with the application name, (mandatory).  
AppAddData: Additional application identification data; e.g. a version number (optional, see note).  
Date&Time: Timestamp; (optional, see note).

NOTE: At least AppAddData or Date&Time are mandatory.

Upon reception of this command, the terminal application looks for the specified objects set, loads it and makes it available for the further dialogue.

### B.1.2 VEMMI\_OpenApplicationResponse

The terminal responds to VEMMI\_OpenApplication with the command VEMMI\_OpenApplication-Response. The parameters are of the following cases:

#### Case 1

OpenAppResult: True: the application has been identified and the objects have been loaded.

#### Case 2

OpenAppResult: False: the application has not been identified, no objects have been loaded.

#### Case 3

AppId, AppAddData, Date&Time

The terminal sends a "Result = true" if the attributes of the stored objects match those requested in the VEMMI\_OpenApplication command in the following way:

AppId and AppAddData are equal or AppId and Date&Time are equal.

If AppId differs, the terminal sends a "Result = false". In all other cases the attributes of the locally identified objects are sent (Case 3).

### B.1.3 VEMMI\_StoreObjects

The host uses this command to request the terminal to store the current loaded objects or a part of them. The host supplies also the attributes which describe the set of objects to be stored.

#### Parameters:

AppId: see above.

AppAddData: see above.

Date&Time: see above.

ObjNum: Number of objects to be stored.

ObjNrList: Object number list of the objects to be stored. The objects include their components with their current values and all other information relevant to the objects. If the parameter is absent all currently loaded objects and resources are stored.

#### **B.1.4 VEMMI\_StoreObjectsResponse**

The terminal informs the host with the command VEMMI\_StoreObjectsResponse about the previous request to store a set of objects.

##### **Parameter:**

StoreResult:                    True: the objects have been stored.  
  
                                  False: the objects could not be stored. The host reaction to this answer is application specific.

A detailed error message may follow this command.

### **B.2 Operative objects**

This functionality provides a method to extent the capabilities of a application during runtime. Via a command the host application can start a program which is located in the terminal and which is application specific and not further specified in this ETS. After termination the control is given back to the VEMMI application. The program has been downloaded via file transfer, is part of the operating system or its presence has been negotiated in an other way. It is up to the host to ensure that the called program is started faultless.

#### **B.2.1 VEMMI\_ExecuteProgram**

The command VEMMI\_ExecuteProgram is used by the VEMMI application to define start a program on the VEMMI terminal.

##### **Parameters:**

ProgName:                        The name of the program to be started.

### **B.3 Colour table**

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. Because the device colour table is shared by more than one application, the GUIs 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 GUI 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 number from these colours. The host application has the possibility to 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 has to be managed by the local GUI.

The following subclauses present the VEMMI commands that may be used to implement colour tables:

### B.3.1 VEMMI\_LoadCoITable

The command VEMMI\_LoadCoITable is used by the VEMMI application to define colours in the colour table.

#### Parameters:

ColEntry:	The colour index from which the colours are defined in ascending order. (optional, default: 0).
ColNum:	Number of colours to be defined (mandatory).
ColRGBList:	This list consists of ColNum elements, each of which is a triplet with the Red, Green and Blue components. Each component value is in the range 0 .. 63. The value 0 means no intensity for the component, the value 63 means maximum intensity (mandatory).

The colours are defined consecutively, started with the index "ColEntry". The terminal sends 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.

### B.3.2 VEMMI\_ResetCoITable

This command is used by the VEMMI application to reset all previously defined colours. If the default colours have been redefined with VEMMI\_LoadColourTable, this command sets the default colours to their predefined values.

## B.4 Set of Objects concept

The purpose of the "Set of Objects" is to offer to the VEMMI application the possibility to establish a link between some VEMMI objects having a strong relationship from the applicative standpoint. This link is expressed by a permanent visual relation between the VEMMI objects of the Set. On the terminal side, any management operation of one VEMMI object of a "Set of Objects" (Open, Close, Accessible, Inaccessible), will lead to the same management operation on all the other VEMMI objects of the Set.

The introduction of the Set of Objects concept leads to the definition of a new VEMMI command:

VEMMI\_CreateSetofObjects <OIN> (OINId)

The Set of Objects is identified by an OIN, because the Set of Objects can be considered as a super-object. The VEMMI\_CreateSetofObjects data part contains only the list of the VEMMI objects of the Set, identified by their OINId.

On reception of a VEMMI\_CreateSetofObjects command, the terminal verifies that all the VEMMI objects of the set are present. Objects that are missing are requested from the VEMMI application. Then the object state and the state parameters accessible / inaccessible of the first VEMMI object listed in the VEMMI\_CreateSetofObjects command are applied to all the other VEMMI objects of the set.

Any command (VEMMI\_Open\_Object, VEMMI\_Close\_Object, VEMMI\_Destroy\_Object, VEMMI\_Obj\_Access\_Enable, VEMMI\_Obj\_Access\_Disable) referred to one VEMMI object of a set is also applied by the terminal to all the other VEMMI objects of the set.

On opening a VEMMI object of a set, from the state "Closed", the terminal verifies that all the VEMMI objects belonging to this set are present. Missing objects are requested. The terminal then opens and displays all these VEMMI objects in the order listed in the VEMMI\_CreateSetofObjects command and gives the focus to the VEMMI object which receives the initial VEMMI\_Open\_Object command.

On selecting an open and accessible VEMMI object of a set, either by the user or by re-opening an open VEMMI object, the terminal displays again all the VEMMI objects of the set in the order listed in the VEMMI\_CreateSetofObjects command and gives the focus to the VEMMI object selected.

When the terminal reaches the limit of its memory capacity, it may decide to destroy one or more VEMMI closed objects of the set as in the general process of local memory management.

#### B.4.1 VEMMI\_CreateSetofObjects

The command VEMMI\_CreateSetofObjects is used by the VEMMI application to create a set of objects.

##### Parameters:

OIN: OIN of the set.  
OINId: List of the OINs of the objects that belong to the set.

#### B.5 Coding

Improvements of the coding structure providing compatibility with the existing coding are for further study.

#### B.6 Provisional command codes

This subclause summarises the provisional codes allocated to identify the future VEMMI commands described in this annex (see table B.1).

**Table B.1: Provisional VEMMI command codes**

VEMMI Command	Command Code
VEMMI_OpenApplication	6/3
VEMMI_OpenApplicationResponse	6/4
VEMMI_StoreObjects	6/5
VEMMI_StoreObjectsResponse	6/6
VEMMI_ExecuteProgram	6/8
VEMMI_LoadColTable	6/A
VEMMI_ResetColTable	6/B
VEMMI_CreateSetofObjects	6/C

## History

Document history	
February 1995	First Edition
February 1996	Converted into Adobe Acrobat Portable Document Format (PDF)