VBApi Help

Api Reference Contents

VBApi Types

Type

```
Bool=WordBool;
 Enum=Byte;
Npnt=Word;
Lpnt=Pointer;
LPStr=PChar;
Hsz= LPHandle;
PHSZ=^HSZ;
HLSTR =LPHandle;
HCtl=LPHandle;
HFormFile=PHandle;
ERR=Word;
HPIC=Word;
Float=Single;
```

VBApi Messages

VBAPI.TPW defines a number of messages that originate with Visual Basic. These messages are listed below, according to category:

Messages	*Description*
VBM_CANCELMODE	Reset internal state.
VBM_CHECKPROPERTY	Check property value.
VBM_COPY	Copy data from Clipboard.
VBM_CREATED	Creation and loading of a control completed.
VBM_DATA_AVAILABLE [3.0]	Notification of data available from the data control.
VBM_DATA_REQUEST	Notification of data requested from the data control.
VBM_DATA_GET [3.0]	Get data from the data control.
VBM_DATA_METHOD [3.0]	Notification of data requested from the data control.
VBM_DATA_SET [3.0]	Send data to the data control.
VBM_DRAGDROP	Item dropped on control.
VBM_DRAGOVER	Item dragged over control.
VBM_FIREEVENT	Implement delayed event.
VBM_GETDEFSIZE [2.0]	Get default size.
VBM_GETPALETTE [2.0]	Request for palette.
VBM_GETPROPERTY	Get property value.
VBM_GETPROPERTYHSZ	Get string to display in Properties bar.
VBM_HELP	Process help request.
VBM_HITTEST [2.0]	Return mouse hit status.
VBM_INITIALIZE	HCTL allocated.
VBM_INITPROPPOPUP	Determine how value is set in Properties bar.
VBM_ISMNEMONIC [2.0]	Mnemonic entered.
VBM_LINKENUMFORMATS [2.0]	Enumerate DDE data formats.
VBM_LINKGETDATA [2.0]	Receive DDE data.
VBM_LINKGETITEMNAME [2.0]	Get DDE item name.
VBM_LINKSETDATA [2.0]	Send DDE data.
VBM_LOADED	All controls on form loaded, or control dynamically loaded.
VBM_LOADPROPERTY	Load property from disk.
VBM_LOADTEXTPROPERTY [2.0]	Load properties as text.
VBM_METHOD	One of control s methods used in a statement.
VBM_MNEMONIC	Respond to access key.
VBM_PAINT [2.0]	Repaint graphical control.
VBM_PAINTMULTISEL [2.0]	Respond to multiple selection.
VBM_PAINTOUTLINE [2.0]	Respond to moving.
VBM_PALETTECHANGED [2.0]	Request to select a palette.
VBM_PASTE	Paste data into Clipboard.
VBM_QPASTEOK	Determine if Paste or Paste Link should proceed.
VBM_SAVEPROPERTY	Save property to disk.
VBM_SAVETEXTPROPERTY [2.0]	Save properties as text.
VBM_SELECTED [2.0]	Control selected at design time.
VBM_SETPROPERTY	Set property value.
VBM_WANTSPECIALKEY [2.0]	Virtual key entered.

VB Notification messages: VBN_{Message}:

VBN_CHARTOITEM [2.0]	WM_CHARTOITEM
VBN_COMMAND	WM_COMMAND
VBN_COMPAREITEM	WM_COMPAREITEM
VBN_CTLCOLOR	WM_CTLCOLOR
VBN_DELETEITEM	WM_DELETEITEM
VBN_DRAWITEM	WM_DRAWITEM
VBN_HSCROLL	WM_HSCROLL
VBN_MEASUREITEM	WM_MEASUREITEM
VBN_PARENTNOTIFY	WM_PARENTNOTIFY
VBN_VKEYTOITEM [2.0]	WM_VKEYTOITEM
VBN_VSCROLL	WM_VSCROLL

VBApi Data Structures

TModel Record

```
Type
    LPMODEL=^TMODEL;
    TModel=Record                   { VB version used by control }
        usVersion:Word;
        fl:LongInt;                 { Bitfield structure }
        ctlproc:TFARPROC;           { The control proc. }
        fsClassStyle:Word;          { Window class style }
        flWndStyle:LongInt;         { Default window style }
        cbCtlExtra:Word;            { # bytes alloc'd for HCTL structure }
        idBmpPalette:Word;          { BITMAP id for tool palette }
        DefCtlName: NPnt;           { Default control name prefix }
        ClassName:NPnt;             { Visual Basic class name }
        ParentClassName:NPnt;       { Parent window class if subclassed }
        proplist:NPPROPLIST;        { Property list }
        eventlist:NPEVENTLIST;      { Event list }
        nDefProp:Byte;              { Index of default property}
        nDefEvent:Byte;             { Index of default event }
        nValueProp:Byte;            { Index of control value property }
        usCtlVersion:Word;          { Identifies the current version of      }
                                    { the custom control. The values      }
                                    { 1 and 2 are reserved for custom     }
                                    { controls created with VB 1.0 and }
                                    { VB 2.0.
                                    }
    end;
```

TPROPINFO Record

```
Type
    PPROPINFO=NPnt;
    TPROPINFO=record
        npszName:NPnt;
        fl:LongInt;                 { PF_ flags }
        offsetData:Byte;            { Offset into static structure }
        infoData:Byte;              { 0 or _INFO value for bitfield }
        dataDefault:LongInt;        { 0 or _INFO value for bitfield }
        npszEnumList:NPnt;          { For type = DT_ENUM, this is          }
                                    { a near ptr to a string containing      }
                                    { all the values to be displayed         }
                                    { in the popup enumeration listbox. }
                                    { Each value is an sz, with an           }
                                    { empty sz indicated the end of list. }
        enumMax:Byte;               { Maximum legal value for enum.}
end;
```

TEVENTINFO Record

```
Type
    TEVENTINFO=record
        npszName:NPnt;
        cParms:Word;
        cwParms:Word;               { # words of parameters   }
        npParmTypes:NPnt;           { list of parameter types }
        npszParmProf:NPnt;          { event parameter profile string }
        fl:LongInt;                 { EF_ flags }
end;
```

TPIC Record

```
Type
    TPicData=record
    case Byte of
        PICType_BitMap:(BitMap:HBitMap;hpal:HPalette);
        PICType_MetaFile:(Meta:THandle;xExt,yExt:Integer);
        PICType_Icon:(Icon:HIcon);
    end;
```

```
    PPIC=^TPIC;
    TPIC=record
        picType:Byte;
        picData:TPicData;
        picReserved:array [0..3] of Byte;
end;
```

The use of each field is described below:

Data path to field	Description
picType	Enumerated type specifying the format of picture. Can be set to any of the following: PICTYPE_NONE PICTYPE_BITMAP PICTYPE_METAFILE PICTYPE_ICON
picData.bmp.hbitmap	Handle to a bitmap.
picData.bmp.hpal [2.0]	Handle to a palette.
picData.wmf.hmeta	Handle to a metafile.
picData.wmf.xExt	Width of the area to contain the picture.
picData.wmf.yExt	Height of the area to contain the picture.
picData.icon.hicon	Handle to an icon.
picData.icon.hcursor	Handle to a cursor; can be NULL if you don t wish to specify a drag cursor for the icon.

TDATASTRUCT Record (PropertyArrays)

```
Type
    TIndex=record
        datatype:Word;    { Type of nth index (Currently always DT_SHORT) }
        data:LongInt;     { Value of nth index}
    end;
    TDataStruct=record
        data:LongInt;     { Data for Get/Set }
        cindex:Word;      { Number of indecies }
       index:array [0..0] of TIndex;
    end;
```

TDRAGINFO Record

```
Type
    TDRAGINFO=record
        Control:HCTL;
        pt:TPOINT;
        state:Word;     { Enter, Over, Exit; only used for VBM_DRAGOVER}
    end;
```

TVBLinkData Record

Data xfer structure for VBM_LINKGETDATA or VBM_LINKSETDATA

```
Type
    LPLinkData=^TVBLinkData;
    TVBLinkData=record
        wReserved:Word;
        cb:LongInt;
        HData:THandle;
        dwReserved:LongInt;
    end;
```

TModelInfo Record

```
Type
    LPModelInfo=^TModelInfo;
    TModelInfo=record
        usVersion:Word;
        lplpmodel:^LPMODEL;
    end;
```

TDataAccess Record

DA structure used for conversations with the data control

```
Type
    LPDataAccess=^TDataAccess;
    TDataAccess=record
        usVersion:Word;                 { VB version of structure filled in }
                                        { when structure is created          }
        sAction:Integer;                { on VBM_DATA_GET/SET                }
                                        { specifies what to get/set on       }
                                        { VBM_DATA_AVAILABLE/REQUEST }
                                        { tells why                                  }
        hctlData:HCtl;                  { The data control providing data }
        hctlBound:HCtl;                 { The bound control receiving data }
        hszDataField:HSZ;               { The name of the field to get value of }
        sDataFieldIndex:Integer;        { the field index used when FieldName is null }
        usDataType:Word;                { the property datatype to convert data to   }
        hlstrBookMark:HLStr;            { used when getting multirow data }
        fs:Word;                        { Bitfield structure }
        lData:LongInt;                  { The data }
        ulChunkOffset:LongInt;          { The offset to start at for GetChunk }
        ulChunkNumBytes:LongInt;   { The number of bytes for GetChunk/SetChunk }
end;
```

VBApi Functions

The VBAPI.TPW file provides access to a subset of the functions residing in VB.EXE and VBRUN100.DLL. These functions include many of the same functions used internally to support standard controls. The functions are listed below alphabetically:

Function	*Description*
VBAllocPic	Allocate HPIC structure.
function VBAllocPic(PntPic:PPIC):HPIC;	
VBAllocPicEx [2.0]	Extended VBAllocPic.
function VBAllocPicEx(PntPic:PPIC;usVersion:Word):HPic;	
VBArrayBounds [2.0]	Returns upper and lower bounds.
function VBArrayBounds(VBArray:HAD;index:Integer):LongInt;	
VBArrayElement [2.0]	Returns a pointer to the value of a Basic array element.
function VBArrayElement(VBArray:HAD;cIndex:Integer;var lpi:Integer):Pointer;	
VBArrayElemSize [2.0]	Returns the size of a single Basic array element.
function VBArrayElemSize(VBArray:HAD):Word;	
VBArrayFirstElem [2.0]	Returns a ponter to the firdt element of a Basic array.
function VBArrayFirstElem(VBArray:HAD):Pointer;	
VBArrayIndexCount [2.0]	Returns the number of indexes for a Basic array.
function VBArrayIndexCount(VBArray:HAD):Integer;	
VBCbSaveFPState [2.0]	Saves the current floating-point state.
function VBCbSaveFPState(pb:Pointer;cb:Word):Word;	
VBClientToScreen [2.0]	Converts the given point to creen coordinates.
procedure VBClientToScreen(Control:HCtl;var Point:Point);	
VBCoerceVariant [2.0]	Cpies the value of a variant to a memory location
function VBCoerceVariant(Variant:PVariant;vtype:Integer;lpData:Pointer):Word;	
VBCreateHlstr	Allocate language string.
function VBCreateHlstr(pb:Pointer;cbLen:Word):HLStr;	
VBCreateHsz	Allocate HSZ string.
function VBCreateHsz(Control:HCtl;Str:LPStr):HSZ;	
VBCreateTempHlstr [2.0]	Creates temp Basic string.
function VBCreateTempHlstr(pb:Pointer;cbLen:Word):HLStr;	
VBDefControlProc	Default message processing.
function VBDefControlProc(Control:HCtl;Wnd:HWnd;Msg:Word;WParam:Word;LParam:LongInt):LongInt;	
VBDerefControl	Get pointer to programmer-defined structure.
function VBDerefControl(Control:HCtl):LongInt;	
VBDerefHlstr	Get pointer to string data.
function VBDerefHlstr(HStr:HLStr):LPStr;	
VBDerefHlstrLen [2.0]	
function VBDerefHlstrLen(HStr:HLStr;var pCbLen:Word):PChar;	
VBDerefHsz	Get pointer to string data.
function VBDerefHsz(HSZStr:HSZ):LPStr;	
VBDerefZeroTermHlstr [2.0]	Returns the pointer the string data.
function VBDerefZeroTermHlstr(HStr:HLStr):PChar;	
VBDestroyHlstr	Remove language string.
procedure VBDestroyHlstr(HStr:HLStr);	
VBDestroyHsz	Remove string.
function VBDestroyHsz(HSZStr:HSZ):HSZ;	
VBDialogBoxParam	Create a pop-up dialog box.
function VBDialogBoxParam(Instance:THandle;TemplateName:LPStr;DialogFunc:TFARPROC;lp:LongInt):Integer;	
VBDirtyForm [2.0]	Indicate property change.
procedure VBDirtyForm(Control:HCtl);	
VBFireEvent	Execute event procedure.
function VBFireEvent(Control:HCtl;IdEvent:Word;LPParams:Pointer):ERR;	
VBFormat [2.0]	converts a string to a format.
function VBFormat(vtype:Integer;lpData:Pointer;lpszFmt:PChar;pb:Pointer;cb:Word):Integer;	
VBFreePic	Decrement reference count and delete HPIC if count is zero.
procedure VBFreePic(Pic:HPic);	
VBGetAppTitle	Get application title.
procedure VBGetAppTitle(Str:LPStr;cbMax:Word);	
VBGetCapture [2.0]	Retrieves a handle to the control that has mouse capture.
function VBGetCapture:HCtl;	
VBGetClientRect [2.0]	Copies the dimensions of the client area to a TRect.
procedure VBGetClientRect(Control:HCtl;var Rect:TRect);	
VBGetControl [2.0]	Searches for a control.

function VBGetControl(Control:HCtl;gc:WORD):HCtl;
VBGetControlHwnd Get handle to window.
function VBGetControlHwnd(Control:HCtl):HWnd;
VBGetControlModel Get model structure.
function VBGetControlModel(Control:HCtl):LPModel;
VBGetControlProperty Get property value.
function VBGetControlProperty(Control:HCtl;IdProp:Word;PData:Pointer):ERR;
VBGetControlName Get property name.
function VBGetControlName(Control:HCtl;lpszName:LPStr):LPStr;
VBGetControlRect [2.0] Copies dimensions of bounding rect of the control.
procedure VBGetControlRect(Control:HCtl;var Rect:TRect);
VBGetDataSourceControl [3.0] Retrieve data control currently bound to.
function VBGetDataSourceControl(Control:HCTL;blsRegistered:Bool):HCtl;
VBGetHInstance Get handle to current instance of Visual Basic.
function VBGetHInstance:THandle;
VBGetHlstr [2.0] Copies the string data to the dest buffer.
function VBGetHlstr(HStr:HLStr;pb:Pointer;cbLen:Word):Word;
VBGetHlstrLen Get string length.
function VBGetHlstrLen(HStr:HLStr):Word;
VBGetHwndControl Get handle to control.
function VBGetHwndControl(Wnd:HWnd):HCtl;
VBGetMode Determine whether in design, run, or break mode.
function VBGetMode:Word;
VBGetPic Dereference picture data.
function VBGetPic(Pic:HPic;PntPic:PPic):HPic;
VBGetPicEx [2.0] Extended VBGetPic function.
function VBGetPicEx(Pic:HPic;PntPic:PPIC;usVersion:Word):HPic;
VBGetRectInContainer [2.0]
procedure VBGetRectInContainer(Control:HCtl;var Rect:TRect);
VBGetVariantType [2.0] Returns the Varaint data type.
function VBGetVariantType(Variant:PVariant):Integer;
VBGetVariantValue [2.0] Gets the value of the given variant.
function VBGetVariantValue(Variant:PVariant;Value:PValue):Integer;
VBGetVersion [2.0] Returns the version of the host environment.
function VBGetVersion:Word;
VBInvalidateRect [2.0] Adds a rectangle to the controls update region.
procedure VBInvalidateRect(Control:HCtl;var Rect:TRect;fEraseBkGnd:Boolean);
VBIsControlEnabled [2.0] Determines if control is Enabled.
function VBIsControlEnabled(Control:HCtl):BOOL;
VBIsControlVisible [2.0] Determines if control is Visible.
function VBIsControlVisible(Control:HCtl):BOOL;
VBLinkMakeItemName [2.0] Create item name that contains control array information.
procedure VBLinkMakeItemName(Control:HCTL;lpszBuf:PChar);
VBLinkPostAdvise [2.0] Sends notification to the DDE client that the link data has been changed.
function VBLinkPostAdvise(Control:HCtl):Word;
VBLockHsz Get pointer to data and prevent string from moving.
function VBLockHsz(HSZStr:HSZ):LPStr;
VBMoveControl [2.0] Changes position and dimensions of a control.
procedure VBMoveControl(Control:HCtl;var Rect:TRect;fRepaint:BOOL);
VBPaletteChanged [2.0] Notifies Visual Basic that the pallette has changed.
procedure VBPaletteChanged(Control:HCtl);
VBPasteLinkOk [2.0]
function VBPasteLinkOk(var phTriplet:THANDLE;Control:HCtl):BOOL;
VBPicFromCF Get picture from Clipboard.
function VBPicFromCF(PntHPic:Pointer;HData:THandle;WFormat:Word):ERR;
VBReadBasicFile Read data file.
function VBReadBasicFile(UsFileNo:Word;pb:Pointer;cb:Word):ERR;
VBReadFormFile Read property value from disk during a load.
function VBReadFormFile(FormFile:HFormFile;pb:Pointer;cb:Word):ERR;
VBRecreateControlHwnd Destroy and recreate window, to enable new window styles.
function VBRecreateControlHwnd(Control:HCtl):ERR;
VBRefPic Increment reference count.
function VBRefPic(Pic:HPic):HPic;
VBRegisterModel Register a control class.
function VBRegisterModel(HMod:THandle;var Model:TModel):Boolean;
VBRelSeekBasicFile Move position in data file a relative distance.

function VBRelSeekBasicFile(usFileNo:Word;offset:LongInt):LongInt;
VBRelSeekFormFile
function VBRelSeekFormFile(FormFile:HFormFile;OffSet:LongInt):LongInt;
VBResizeHlstr [2.0] Reallocates the size of a given string.
function VBResizeHlstr(HStr:HLStr;newCbLen:Word):Word;
VBRestoreFPState [2.0] Restores the floating point state.
procedure VBRestoreFPState(pb:Pointer);
VBRuntimeError [2.0] Generates a runtime error.
procedure VBRuntimeError(err:Word);
VBScreenToClient [2.0] Converts points.
procedure VBScreenToClient(Control:HCtl;var Point:TPoint);
VBSeekBasicFile [3.0] Move position in data file.
function VBSeekBasicFile(usFileNo:Word;offset:LongInt):LongInt;
VBSeekFormFile Moves the current position within a form file.
function VBSeekFormFile(FormFile:HFormFile;OffSet:LongInt):LongInt;
VBSendControlMsg Send message to a control.
function VBSendControlMsg(Control:HCtl;Msg,WParam:Word;LParam:LongInt):LongInt;
VBSetControlFlags [2.0] Set and return the characteristics of a control.
function VBSetControlFlags(Control:HCtl;mask:LongInt;value:LongInt):LongInt;
VBSetControlProperty Set property value.
function VBSetControlProperty(Control:HCtl;IdProp:Word;Data:LongInt):Err;
VBSetErrorMessage Set text of next error message.
function VBSetErrorMessage(error:ERR;Str:LPStr):ERR;
VBSetHlstr Assign new string data.
function VBSetHlstr(PHStr:Pointer;pb:Pointer;cbLen:Word):ERR;
VBSetVariantValue [2.0] Sets a varaint to a given value.
function VBSetVariantValue(Variant:PVariant;vtype:Integer;lpData:Pointer):Word;
VBSuperControlProc Call parent class directly.
function VBSuperControlProc(Control:HCtl;Msg,WParam:Word;LParam:LongInt):LongInt;
VBTranslateColor [2.0] Converts a VB color to an RGB color.
function VBTranslateColor(Control:HCtl;Color:LongInt):LongInt;
VBUnlockHsz Unlock string address.
function VBUnlockHsz(HSZStr:HSZ):LPStr;
VBUpdateControl [2.0] Updates a control.
procedure VBUpdateControl(Control:HCtl);
VBWriteBasicFile Write to data file.
function VBWriteBasicFile(UsFileNo:Word;pb:Pointer;cb:Word):ERR;
VBWriteFormFile Write property value to disk during a save.
function VBWriteFormFile(FormFile:HFormFile;pb:Pointer;cb:Word):ERR;
VBXPixelsToTwips Convert X units to twips.
function VBYPixelsToTwips(Pixels:Integer):LongInt;
VBXTwipsToPixels Convert X units to pixels.
function VBXTwipsToPixels(Twips:LongInt):Integer;
VBYPixelsToTwips Convert Y units to twips.
function VBXPixelsToTwips(Pixels:Integer):LongInt;
VBYTwipsToPixels Convert Y units to pixels.
function VBYTwipsToPixels(Twips:LongInt):Integer;
VBZOrder [2.0] Alters the drawing order of the control.
procedure VBZOrder(Control:HCtl;zorder:WORD);

VBApi Properties

Standard property constant	*Comments*
PPROPINFO_STD_ALIGN [2.0]	Forces control to align with form.
PPROPINFO_STD_BACKCOLOR	BackColor. You must send a WM_CTLCOLOR message to get a brush that contains the background color.
PPROPINFO_STD_BORDERSTYLEON	BorderStyle: set to True by default.
PPROPINFO_STD_BORDERSTYLEOFF	BorderStyle: set to False by default.
PPROPINFO_STD_CAPTION	Caption. Requires that you respond to WM_SETTEXT, WM_GETTEXT, and WM_GETTEXTLENGTH messages.
PPROPINFO_STD_CLIPCONTROLS [2.0]	Sets or clears the windows child flag.
PPROPINFO_STD_CTLNAME	CtlName. This property is required, and should be placed first in the list..
PPROPINFO_STD_DATACHANGED [3.0]	Indicates the data in the bound control has changed by something other than getting the current record .
PPROPINFO_STD_DATAFIELD [3.0]	Allows the control to bind to a database field.
PPROPINFO_STD_DATACHANGED [3.0]	Allows the control to bind to a specific data control.
PPROPINFO_STD_DRAGICON	DragIcon
PPROPINFO_STD_DRAGMODE	DragMode
PPROPINFO_STD_ENABLED	Enabled
PPROPINFO_STD_FONTNAME	FontName. Requires that you respond to WM_SETFONT and WM_GETFONT messages. Responding to these messages provides the support necessary for all the font properties. All font properties are saved and loaded together with FontName.
PPROPINFO_STD_FONTBOLD	FontBold
PPROPINFO_STD_FONTITALIC	FontItalic
PPROPINFO_STD_FONTSIZE	FontSize
PPROPINFO_STD_FONTSTRIKE	FontStrike
PPROPINFO_STD_FONTUNDER	FontUnder
PPROPINFO_STD_FORECOLOR	ForeColor. You must also send a WM_CTLCOLOR message to use this property.
PPROPINFO_STD_HEIGHT	Height
PPROPINFO_STD_HELPCONTEXTID [2.0]	User defind help context id.
PPROPINFO_STD_HWND	The controls runtime window handle.
PPROPINFO_STD_IMEMODE [2.0]	See VB control Ref.
PPROPINFO_STD_INDEX	Index. This property is required, and should be placed second in the list.
PPROPINFO_STD_LAST [2.0]	Indicates the max value of a standard property.
PPROPINFO_STD_LEFT	Left. Top, Width, and Height are all saved and loaded together with Left.
PPROPINFO_STD_LEFTNORUN [2.0]	Use this prop instead of left for an invisible control.
PPROPINFO_STD_LINKITEM [2.0]	DDE property.
PPROPINFO_STD_LINKMODE [2.0]	DDE property.
PPROPINFO_STD_LINKTIMEOUT [2.0]	DDE property.
PPROPINFO_STD_LINKTOPIC [2.0]	DDE property.
PPROPINFO_STD_MOUSEPOINTER	Mousepointer.
PPROPINFO_STD_NAME [2.0]	Replaces CTLNAME.
PPROPINFO_STD_NONE [2.0]	Place holder for removed properties.
PPROPINFO_STD_PARENT	Parent
PPROPINFO_STD_TABINDEX	TabIndex. Should be supported if fFocusOk or fMnemonic flag is set.
PPROPINFO_STD_TABSTOP	TabStop. Should be supported if fFocusOk or fMnemonic flag is set.
PPROPINFO_STD_TAG	Tag
PPROPINFO_STD_TEXT	Text. This property is the same as Caption, except for the property name.
PPROPINFO_STD_TOP	Top
PPROPINFO_STD_TOPNORUN [2.0]	Use this prop instead of top for an invisible control.
PPROPINFO_STD_WIDTH	Width

PPROPINFO_STD_VISIBLE	Visible

VBApi Events

Standard event constant	*Comments*
PEVENTINFO_STD_CLICK	Click. Fired when the control has captured the mouse, a WM_LBUTTONUP message is received, and the button is released over the control. This event is fired after the standard MouseDown and MouseUp events.
PEVENTINFO_STD_DBLCLICK	DblClick. Similar to Click, but fired when a WM_LBUTTONDBLCLK message is received.
PEVENTINFO_STD_DRAGDROP	DragDrop. Fired after a VBM_DRAGDROP message is received.
PEVENTINFO_STD_DRAGOVER	DragOver. Fired after a VBM_DRAGOVER message is received.
PEVENTINFO_STD_GOTFOCUS	GotFous. After receiving a WM_GOTFOCUS message, the default message processor posts a VBM_FIREEVENT message. This is a mechanism that results in delayed firing of the event, which is processed after other pending messages are.
PEVENTINFO_STD_KEYDOWN	KeyDown. Fired when either a WM_KEYDOWN or a WM_SYSKEYDOWN message is received.
PEVENTINFO_STD_KEYPRESS	KeyPress. Fired when a WM_CHAR message is received.
PEVENTINFO_STD_KEYUP	KeyUp. Fired when either a WM_KEYUP or a WM_SYSKEYUP message is received
PEVENTINFO_STD_LAST [2.0]	Place holder for the last standard event.
PEVENTINFO_STD_LINKCLOSE [2.0]	DDE Event.
PEVENTINFO_STD_LINKERROR [2.0]	DDE Event.
PEVENTINFO_STD_LINKNOTIFY [2.0]	DDE Event.
PEVENTINFO_STD_LINKOPEN [2.0]	DDE Event.
PEVENTINFO_STD_LOSTFOCUS	LostFocus. Fire when a WM_LOSTFOCUS message is received. The same delayed-processing mechanism is used as for GotFocus.
PEVENTINFO_STD_MOUSEDOWN	MouseDown. Fired if any BUTTONDOWN (right, left, or middle button) message is received. The mouse is captured as a result of this event.
PEVENTINFO_STD_MOUSEMOVE	MouseMove. Fired when a WM_MOUSEMOVE message is received.
PEVENTINFO_STD_MOUSEUP	MouseUp. Fired if any BUTTONUP (right, left, or middle button) message is received. The mouse capture is released as a result of this event.
PEVENTINFO_STD_NONE [2.0]	Place holder fo r removed event.

VBApi Flags

This chapter provides a summary of each data structure used in custom control code. Understanding these structures can be useful in writing general DLL routines as well as in writing custom controls. Features that are specific to a version level of Visual Basic are denoted by the version number inside brackets. For example, version 3.0 are denoted by [3.0].

Model Flags

Flag Value	Tells Visual Basic To:
MODEL_fArrows	Send to the control keyboard messages for arrow keys. If this flag is off, arrow keys are used to move between controls.
MODEL_fFocusOk	Enable the control to get the focus at run time.
MODEL_fMnemonic	Respond to the controls mnemonic key (access key) by moving the focus to the control and sending a VBM_MNEMONIC message. The control must support the standard Caption property to have a mnemonic key.
MODEL_fChildrenOk	Enable the Visual Basic developer to draw other controls on top of this control at design time. This lets the control function as a container of other controls, as picture boxes and frames can.
MODEL_fLoadMsg	Enable the control to get VBM_CREATED and VBM_LOADED messages.
MODEL_fInitMsg	Enable the control to get VBM_INITIALIZE messages.
MODEL_fDesInteract	Enable the control to get right-mouse-button messages at design time: WM_RBUTTONDOWN, WM_RBUTTONDBLCLK, WM_RBUTTONUP.By processing these messages at design time, the control procedure can provide special mechanisms for manipulating the control and its properties.
MODEL_fGraphical [2.0]	Indicate control is a graphical control.
MODEL_fInvisAtRun [2.0]	Specify that the control is invisible at run time. Visual Basic automatically handles the representation of the control on the form at design time it uses the same icon that appears in the Toolbox

Property Data Type Flags

Only one of the following flags can be selected for each property. The selected flag is then logically OR'ed with flags listed in the next section.

Flag Value	Resulting data type of:
DT_BOOL	Boolean. This is a short integer, though it can be packed into as little as 1 bit. All incoming and outgoing nonzero values are converted to -1 during a set or get operation.
DT_COLOR	A long integer that holds a color value. This value is an RGB value, or (if the most significant bit is set) the lowest 3 bytes contain a system-color number defined in WINDOWS.H. Visual Basic automatically validates numbers input for validity as color values. The value is displayed in the Properties window using

hexadecimal radix.

DT_ENUM — A short integer holding an enumerated value (0 255). Selection of this data type (as opposed to DT_SHORT) enables the npszEnumList and enumMax fields, described in the previous section. The data can be packed into a bit field from 1 to 4 bits wide.

DT_HLSTR [2.0] — String. The string data should be stored as an HLSTR type. DT_HLSTR property values are string descriptors representing strings of bytes that can contain embedded NULL characters. This is the principle advantage over DT_HSZ strings which are terminated by a NULL character. HLSTR strings that you allocate must be freed in response to WM_NCDESTROY. Refer to Chapter 9 of the Control Development Guide, "Handling Strings and Fonts," for more details on HLSTR strings.

DT_HSZ — String. The string data should be stored as an HSZ type. If the PF_fSetData flag is set, allocation and freeing of the HSZ string are handled by Visual Basic. HSZ strings that you allocate must be freed in response to WM_NCDESTROY. Refer to Chapter 9 of the Control Development Guide, "Handling Strings and Fonts," for more details on HSZ strings.

DT_LONG — 32-bit signed integer.

DT_OBJECT — Points to an iDispatch interface. Refer to Technical Note 1: Support for DT_OBJECT Properties, TN001.TXT, in the CDK directory.

DT_PICTURE — Picture structure. The picture should be stored as an HPIC handle. If the PF_fSetData flag is set, Visual Basic allocates HPIC handles as needed. The HPIC must be freed in response to WM_NCDESTROY.

DT_REAL — 4-byte real.

DT_SHORT — 16-bit signed integer. The data can be packed into a bit field from 1 to 4 bits wide.

DT_XPOS — Long integer expressing an X coordinate. The data for a get or set operation is expressed in terms of the container's scale. The data gets converted, if necessary, so that the custom control code uses units expressed in twips, with the origin (0,0) at the control's upper-left corner. Similar conversions are performed for the remaining types.

DT_XSIZE — Long integer expressing an X size in twips. Origin does not affect size conversions.

DT_YPOS — Long integer expressing a Y

	coordinate in twips.
DT_YSIZE	Long integer expressing a Y size in twips.

Property Flags

You can use any number of the following flags in the fl field of the TPROPINFO structure.

Important: When the PF_fPropArray flag is used, the following flags must also be used: PF_fSetMsg, PF_fGetMsg, and PF_fNoShow. The PF_fSaveData flag must not be used.

Flag Value	**Description:**
PF_fGetData	Indicates that Visual Basic gets the property value by directly copying data from the programmer-defined structure. The offsetData and infoData fields of the TPROPINFO table specify exactly where the data is located within the structure.
PF_fGetMsg	Causes Visual Basic to send a VBM_GETPROPERTY message when the property value is requested. Either this flag or the PF_fGetData flag should be used.
PF_fSetCheck	Causes Visual Basic to send a VBM_CHECKPROPERTY message before it sets the value of the property. This gives the custom control code a chance to check the validity of the property value. If the control procedure returns a nonzero error code in response to this message, Visual Basic reports an error to the user and will not proceed with setting the property to the new value.
PF_fSetData	Indicates that Visual Basic sets the value of the property by placing data directly in the programmer-defined structure.
PF_fSetMsg	Causes Visual Basic to send a VBM_SETPROPERTY message when setting of the property is attempted. Either this flag or the PF_fSetData flag should be used, unless the property is read-only. If both are used, then the data is transferred before the message is sent.

PF_fSaveData	When the form is being saved to a file, Visual Basic saves the property value by getting its value and then writing the data to disk directly. Similiarly, Visual Basic loads the property by reading its value directly from the disk and then setting it.
PF_fSaveMsg	Causes Visual Basic to send a VBM_SAVEPROPERTY message to the control whenever the form is being saved to a file, and a VBM_LOADPROPERTY message when the form is loaded from a file. These messages pass a handle to file location. It is then the responsibility of the control procedure to read or write from disk (using VBReadFormFile and VBWriteFormFile).
PF_fNoShow	Prevents the property from appearing in the Properties bar.
PF_fNoRuntimeW	Indicates that the property is read-only at run time.
PF_fGetHszMsg	Causes Visual Basic to send a VBM_GETPROPERTYHSZ message to the control whenever the property value is to be displayed in the Properties bar. If this flag is not used, then Visual Basic simply gets the property value and displays it in the Settings box.
PF_fUpdateOnEdit	Causes the property value to be set every time a character is typed in the Settings box of the Properties bar. If this flag is not used, the property is not set until change is committed.
PF_fEditable	Enables the Visual Basic developer to edit text in the Settings box directly, even if a list box or pop-up window is used. If neither is used, this flag has no effect.
PF_fPreHwnd	Causes the property to be loaded before the controls window structure is created. A property should be pre-hwnd if it affects a window style. The actual

	setting of the window style (that is, the style altered by the pre-hwnd property) should be made in response to a WM_NCCREATE message.
PF_fDefVal	Causes Visual Basic to avoid saving and loading the property to disk when the value is equal to the dataDefault value in the TPROPINFO structure. This flag affects only loading and saving, not default values of newly created controls. During saving, the property value is not written to disk if equal to the dataDefault value. Then, during loading, if the property was not saved to disk, it is set to the dataDefault value. (But PF_fNoInitDef, described next, alters loading behavior.)
PF_fNoInitDef	Prevents Visual Basic from setting the property to the dataDefault value during loading of the control (though you can still initailize the property by responding to messages). If PF_fDefVal is not used, PF_fNoInitDef has no effect.
PF_fPropArray	Specifies that the property is an array. When this flag is used, you must use a TDATASTRUCT structure to implement all getting and setting of the property. Any type supported as a simple (scalar) property can also be used as the base type of a property array. Important: when this flag is used, the following flags must also be used: PF_fSetMsg, PF_fGetMsg, and PF_fNoShow. The PF_fSaveData flag must not be used.
PF_fLoadMsgOnly[2.0]	Causes Visual Basic to send VBM_LOADPROPERTY messages when the form is loaded from a

file. These messages pass a handle to file location. It is then the responsibility of the control procedure to read from disk (using the VBReadFormFile function).

This property is similar to PF_fSaveMsg, except that a VBM_SAVEPROPERTY messages are not generated. This allows custom controls to be backward compatible by loading properties in previous versions that are no longer supported in newer ones.

PF_fLoadDataOnly[2.0] Causes Visual Basic to load the property from a form file, but prevents the property from being written back out to the form file. This property is similar to PF_fSaveData, except that properties are not written out to the form file. This allows custom controls to be compatible with Visual Basic version 1.0.

PF_fNoMultiSelect[2.0] Specifies that any given custom property NOT be displayed in the Properties window when that control is part of a selected group. Currently, no properties of type DT_PICTURE are included in a multiple selection.However, if PF_fNoMultiSelect is not present, a future version of Visual Basic may include a given property of type DT_PICTURE in a multiple selection. If you must guarantee that a property is never placed in a multiple selection (even if it is DT_PICTURE), then PF_fNoMultiSelect must be set.

PF_fNoRuntimeR[2.0] Indicates that the property is write-only at run time. Any attempt to read the property generates a runtime error. When

combined with PF_fNoRuntimeW, it indicates that the property is not available at all at run time (a design-time-only property), and references to it are flagged as Visual Basic compilation errors.