VB Language Manager Help

Contents

Getting Started with VBLM

Thank you for trying VB Language Manager!

There's two or three things you should do before you dive in:

- Check out the overview to get a quick sense of how VBLM works;
- Check out about documentation to get a quick sense of how the online documentation works.
- If you are using a non-English version of Visual Basic® or Windows®, you probably need to reconfigure the VB Control Interface.

Five minutes spent reading these topics will be time well spent. Then crank up VBLM and go conquer world markets with your translated Visual Basic® applications!

Again, thank you for your business!

Ben Whipple
President & Programmer

WhippleWare

Getting Started with VBLM Pro

Thank you for trying the Professional Edition of VB Language Manager!

There's two or three things you should do before you dive in:

- Check out the overview to get a quick sense of how VBLM Pro works;
- Check out about documentation to get a quick sense of how the online documentation works.
- If you are using a non-English version of Visual Basic® or Windows®, you probably need to reconfigure the VB Control Interface.

Five minutes spent reading these topics will be time well spent. Then crank up VBLM Pro and go conquer world markets with your translated Visual Basic® applications!

Again, thank you for your business!

Ben Whipple
President & Programmer

WhippleWare

VB Language Manager Options

VBLM has many options that allow you to control and configure it to your preferences. All are accessible from the main window's option menu, and most are also accessible elsewhere. The options fall in several categories:

- General Options control a handful of high-level behaviors, such as screen painting and interaction with the help system.
- String Extraction Options allow you to control which strings VBLM will extract from your VB project for translation.
- Language Table Editor Options allows you to control the LTE's display format, sort order, and print format.
- Build Options control the build of new language versions of the parent VB project.

The Professional Edition of VBLM has yet more options, including the ability to work with third party translation programs, to width-check language tables, build runtime switchable multi-lingual applications, and more.

Note: All current option settings are preserved on exit in VBLM's initialization file and restored on entry.

VB Language Manager Windows

VBLM has two dominant windows:

The Main Window that appears when you start the program; and

The Language Table Editor that pops up when you add or edit a language table.

In the course of using VBLM, however, you will also encounter many others. Most are simply for setting options and are self-explanatory, but there are a few others. Click the window for which you'd like more information:

The About Box

The Status Window

The VB Control Window

The VB Error Window

The Width Check Viewer (Pro Only)

VBLM's Menu System

VBLM's Main Window and the Language Table Editor each have their own menu system. Click the menu for which you'd like more information:

Main Menus

File Menu

Options Menu

Help Menu

LTE Menus

File Menu

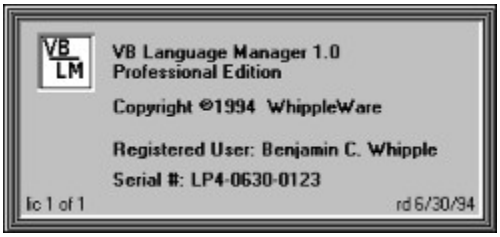
Edit Menu

Options Menu

Search Menu

Help Menu

VBLM's About Box



The About Box displays 5 useful tidbits of information:

1) The version number and the edition.

2) The registered user.

3) The serial number.

4) The current license and the number of licenses for simultaneous use (lower left).

5) The release date of the version (lower right).

Note: Our philosophy is to continuously improve our products and to fix all bugs as soon as they are reported. Rather than changing the version number everytime we make a fix, we change the release date. So this date is important information.

Build Options

VBLM's Build Options, set on the Build Options window and accessible from the main options menu and at build time, control the build of new language versions of the parent VB project. They include the following:

Language Table(s) to Use

The language table(s) that will be used during the build; see Language Table to Use.

Project Name

The name VBLM will give the new project's MAK file. By default, it is identical to the parent. Enabled only at build time.

Target Directory

The directory where VBLM will build the new version. See Target Directory.

Interaction with VB

The Interaction with VB option controls VBLM's interaction with VB when it completes a build.

Project Files Without Strings

This option controls VBLM's behavior when it encounters files that contain no strings. See Project Files Without Strings.

Build Runtime Switchable (RTS) Multi-Lingual Version (Pro Only)

If you have VBLM Pro, this checkbox enables the build of RTS multi-lingual applications. See Building Multi-Lingual Projects.

VB Directory

If the directory where VB.EXE is stored on your PC is in your command search path, leave this blank. If VB.EXE is not in your path, specify its directory here so VBLM can find, start, and control it.

Language Table(s) to Use

The **Language Table(s) to Use** option, set on the Build Options window and enabled only at build time, tells VBLM which language table(s) to use during the build it is about to perform.

If you have the Standard Edition of VBLM, you may select only one table.

If you have the Professional Edition of VBLM, you can check the **Build RTS** box, select as many tables as you want, and have VBLM Pro build an RTSMLV. See Building RTS Multi-Lingual Projects for more information.

The Target Directory

The **Target Directory**, set on the Build Options window and enabled only at build time, tells VBLM where to build the translated version of your VB project.

By default, the target directory is a subdirectory of the parent project directory named for the currently selected language table. VBLM will advise you and create it if it does not exist.

The target directory **must be different from the parent project directory**, as otherwise VBLM would overwrite your original source files. VBLM will not let you procede if it is the same.

The target directory option is enabled only at build time.

Project Files Without Strings

The **Project Files Without Strings** option, set on the Build Options window and accessible from the main options menu and at build time, controls VBLM's behavior when it encounters files in the parent project that contain no extracted strings (CONSTANT.TXT, for example). It has two possible settings:

Make Copies

VBLM makes a new copy of the file in the target directory. Although unnecessary because the file is unchanged, this setting has the effect of creating a wholly new, self-contained project in the target directory.

Reference Originals

VBLM inserts the full path and file name of the original file in the new MAK file. This saves on disk space and eases maintenance. Until you select a "save file as" in VB, however, VB won't tell you that the file is in a different directory.

Building Runtime Switchable Multi-Lingual Projects (Pro Only)

VBLM Pro allows you to build runtime switchable multi-lingual versions of your VB applications. An RTSMLV stores strings in a separate database and either learns from the command line or queries the user at startup about which language to use.

To have VBLM build an RTSMLV, simply check **Build RTS** on the Build Options window, then select as many languages on the list as you want included. When you click **OK**, VBLM will ask if you wish to include the design-time language as well, and if so, its name.

When VBLM builds an RTSMLV, it does several things differently than when it builds a single-language version. The best way to thoroughly understand how it all works is to build an RTSMLV and explore the code. But here's an overview:

The Multi-Language Database

VBLM creates a language database file in the target directory. The file is named LANGUAGE.DAT, and it contains all of the strings in all of the selected languages. See LANGUAGE.DAT: The RTS Language Database for information on file format.

RTS Support Files

VBLM copies two supplied support files, one form and one module, from its own directory to the target directory and adds them to the RTSMLV MAK file. The module contains the all-important **VBLM_RTString()** function, which takes one argument (a string index) and returns a string from the database. The first time it is called, VBLM_RTString() also initializes the database, using the form to prompt a language choice if one is not specified on the command line. See RTS Support Files.

RTS Code Modifications

When VBLM builds a single language version, the only changes it makes when rewriting your project code in the target directory are string swaps. When building an RTSMLV, however, VBLM adds, eliminates, and modifies code in various ways to support RTS. The changes made depend on the file type and the section within the file. Click any of the following for details:

Definition Section, Form Files

Declaration Section, All Files

Code Section, Form Files

Code Section, All files

Note: All of the support for runtime switching is implemented in VB code that is added and/or modified during the build. This means that you can easily modify the code yourself in anyway you see fit, whether to spruce up the language selection form, to retrieve the language selection from an INI file, or whatever you want.

RTS Code Modifications: Form Definition Sections (Pro Only)

As VBLM rewrites the definition section of form files for an RTSMLV, it replaces all RTS property strings with null strings.

There is no point in keeping the originals because they will never be used and simply add overhead.

RTS Code Modifications: Declaration Section, All Files (Pro Only)

If you have translated any global, form, or module scoped string constants, VBLM eliminates them by commenting out the lines where they are declared. This is because it is impossible to switch constant values at runtime.

If VBLM finds any of these constants, after commenting them out on the first pass it then takes a second pass through the code to locate and replace all references to them with appropriately indexed calls to **VBLM_RTString()**.

RTS Code Modifications: Code Section, Form Files (Pro Only)

If any of the string properties of a form or its controls have been translated, when building an RTSMLV VBLM appends a **VBLM_SetProperties** procedure to its code section, and inserts a call to this procedure at the top of form's load procedure.

If the form has no load event handler, VBLM creates one.

The **VBLM_SetProperties** procedure simply calls **VBLM_RTString()** for each RTS property. For example:

```
Sub VBLM_SetProperties ()
  frmMain.Caption = VBLM_RTString(860)
  cmd(0).Caption = VBLM_RTString(269)
  cmd(1).Caption = VBLM_RTString(127)
  cmd(2).Caption = VBLM_RTString(163)
  cmd(3).Caption = VBLM_RTString(147)
  lblErr(4).Caption = VBLM_RTString(885)
End Sub
```

RTS Code Modifications: Code Section, All Files (Pro Only)

As VBLM reads through the code section of all files when building an RTSMLV, it replaces all embedded strings found in the language table with calls to **VBLM_RTString()**.

For example, assuming that you extracted and translated "Are you sure you want to trash your hard disk?", the line of code

```
TrashMe = MsgBox("Are you sure you want to trash your hard disk?", 36) = IDYES
```

will be rewritten as

```
TrashMe = MsgBox(VBLM_RTString(113), 36) = IDYES
```

where 113 is the string's index in the language table.

If the line of code references global or file level string constants that have been translated, as discussed elsewhere, these references are also replaced with calls to **VBLM_RTString()**.

Runtime Switchable Multi-Lingual Version (Pro Only)

RTSMLV is shorthand for "runtime switchable multi-lingual version," a version of your VB application created by VBLM Pro that stores its strings in a separate database, contains code to support runtime switching, and either learns from the command line or queries the user at startup about which language to use.

LANGUAGE.DAT: The RTS Language Database (Pro Only)

The LANGUAGE.DAT file created when you build an RTSMLV is a minimalist database that contains a list of included languages and all of the translated strings.

After much experimentation to get optimal performance, we opted to use VB's built-in support for variable length strings. Thus the basic record in the file is a simple user-defined data type with a single varaible length string element The data type, named "tagVBLM_VS," is declared in the RTS support module.

The precise format of LANGUAGE.DAT is as follows:

Item[Data Type] (Explanation)

NumLanguages[Integer] (# of language tables in the file)

(for i= 1 to NumLanguages)

Language(i)[tagVBLM_VS] (Language(i).String = Language Name)

Offset(i)[Long] (File position where Language(i) string data begins)

(next)

NumStrings[Long] (# of strings in a table)

(for i= 1 to NumLanguages) (for j= 1 to NumStrings)

Language(i) String(j) [tagVBLM_VS] (Language(i) String(j) String = String)

(next) (next)

RTS Support Files (Pro Only)

When VBLM builds an RTSMLV, it copies two support files into the target directory and adds them to the MAK file. The source files, VBLM_RTS.FRM and VBLM_RTS.BAS, were installed in VBLM's directory during setup, and VBLM expects to find them there.

VBLM_RTS.FRM

This file defines a simple form named **frmVBLM_RTS**. The form, which displays a list of language choices and an OK button, prompts the user on startup for a language choice. It is not displayed if the RTSMLV was executed with "/L=Language" on the command line, where "Language" matches one of the language names in the database. See VBLM_RTString(), the function that loads it, for more detail.

Note: Since **frmVBLM_RTS** often will be the first part of your application seen by the user, you will doubtless want to customize the copy of it in the target directory.

VBLM_RTS.BAS

This module file contains only two items, a type definition and a function:

Type **tagVBLM_VS** has a single variable length string element named "String." It defines the record type for the language database. See The RTS Language Database for more information.

The **VBLM_RTString()** function is the core of runtime switching. It takes one argument (a string index) and returns a string from the database. When VBLM builds an RTSMLV, it inserts VBLM_RTString() function calls in place of all translated strings. See VBLM_RTString().

Note: Copies of both files are included in this help file, and can be copied out using the clipboard: See VBLM_RTS.FRM source code and VBLM_RTS.BAS source code.

RTS Support Code: VBLM_RTString() (Pro Only)

The **VBLM_RTString()** function, defined in the VBLM_RTS.BAS support module, is the core of runtime switching support. It takes one argument (a string index) and returns a string from the database. When VBLM builds an RTSMLV, it inserts VBLM_RTString() function calls in place of all translated strings.

The first time VBLM_RTString() is called, it initializes the database. The source code is fully documented and included in this help file. You'll get the best understanding of this process by reading it, but here's what happens in brief:

1) VBLM opens the database file and retrieves the number of language tables it contains and their names and locations within the file.

2) If the command line does not contain "/L=ValidLanguageName," the function modally displays frmVBLM_RTS and waits for the user to select a language.

3) By default, VBLM_RTString() then loads the strings in the selected language table into a static array of type tagVBLM_VS and closes the file. This optimizes performance for speed. You can easily reconfigure the function to optimize for memory use, however, by changing the OPTIMIZATION constant in the source code. If you do so, VBLM_RTString() builds an array of file pointers and leaves the file open. The Optimization method is fully explained in the code.

The Optimization method selected has an obvious effect on the source of the string that VBLM_RTString(x) returns. When optimized for speed, the string is in memory as element x of the static string array, and the function returns it without further ado. When optimized for memory, the string is in the disk file at the location pointed to by element x of the static pointer array, and the function has to seek and fetch. If the host system has a disk cache, however, the majority of calls will not actually result in a trip out to disk; the data will be in the cache.

RTS Support Code: VBLM_RTS.FRM (Pro Only)

'This topic includes the contents of the VBLM_RTS.FRM support file.

```
VERSION 2.00
Begin Form frmVBLM_RTS
   BackColor       =   &H00C0C0C0&
   BorderStyle     =   3  'Fixed Double
   ClientHeight    =   2850
   ClientLeft      =   4065
   ClientTop       =   2295
   ClientWidth     =   3465
   ControlBox      =   0   'False
   Height          =   3255
   Left            =   4005
   MaxButton       =   0   'False
   MinButton       =   0   'False
   ScaleHeight     =   2850
   ScaleWidth      =   3465
   Top             =   1950
   Width           =   3585
   Begin CommandButton cmdOK
      Caption         =   "OK"
      Default         =   -1  'True
      Height          =   375
      Left            =   1020
      TabIndex        =   1
      Top             =   1740
      Width           =   1455
   End
   Begin ListBox lstLanguages
      Height          =   1005
      Left            =   240
      TabIndex        =   0
      Top             =   540
      Width           =   2895
   End
   Begin Label lbl
      Alignment       =   2  'Center
      BackStyle       =   0  'Transparent
      Caption         =   "RT Multi-Language by VB Language Manager from WhippleWare©"
      FontBold        =   0   'False
      FontItalic      =   0   'False
      FontName        =   "MS Sans Serif"
      FontSize        =   8.25
      FontStrikethru  =   0   'False
      FontUnderline   =   0   'False
      Height          =   435
      Index           =   1
      Left            =   120
      TabIndex        =   3
      Top             =   2280
      Width           =   3255
   End
   Begin Label lbl
      Alignment       =   2  'Center
      BackStyle       =   0  'Transparent
      Caption         =   "Please Select a Language"
      Height          =   195
      Index           =   0
      Left            =   240
      TabIndex        =   2
      Top             =   180
      Width           =   2835
   End
End
Option Explicit

Sub cmdOK_Click ()
```

```
    If lstLanguages.ListIndex > True Then
        Me.Hide
    Else Beep
    End If

End Sub

Sub lstLanguages_DblClick ()
    cmdOK_Click
End Sub
```

RTS Support Code: VBLM_RTS.BAS (Pro Only)

'This topic includes the entire contents of the VBLM_RTS.BAS support module.

```
'VBLM_RTS.BAS
'VB Language Manager Runtime Language Switching Support Module
'Copyright 1994 by WhippleWare

'================================================================
'DECLARATIONS
'================================================================

Option Explicit
DefInt A-Z

'the tagVBLM_VS data type defines the language database record

Type tagVBLM_VS
    String As String
End Type

'================================================================
' END OF DECLARATIONS
'================================================================

'The VBLM_RTString function is the core of runtime switching (RTS)
'
'All translated strings and properties have been replaced with
'calls to VBLM_RTString
'
'The function is passed an index and returns a string
'The first call initializes the database
'
'================================================================
'================================================================
'
Function VBLM_RTString (Index As Long) As String

'================================================================
'================================================================
'STOCK VB CONSTANTS USED FOR CLARITY
'THESE CAN BE DELETED IF THEY ARE ALREADY DECLARED IN THIS PROJECT
'WITH GLOBAL SCOPE

Const MB_STOP = 16
Const MB_ABORTRETRYIGNORE = 2
Const MB_ICONEXCLAMATION = 48
Const IDABORT = 3
Const IDRETRY = 4
Const IDIGNORE = 5

'================================================================
' LOCAL DECLARATIONS
'================================================================

'RTS_FILE is the name of the database file created by VBLM
'VBLM_RTString expects to find it in the application directory

    Const RTS_FILE = "LANGUAGE.DAT"

'================================================================
'OPTIMIZATION METHOD

'VBLM_RTString allows you to optimize its performance for either memory or speed.
'When optimized for speed (the default), it only goes to disk the first time
'it is called, and loads the entire language table into an array in memory.
'Subsequent calls are very fast, and since the Strings() array consists of
'user-defined types, it does not intrude on local string space.

'If your application has a very large language table, however, this method
```

```
'might cause memory problems.  If so, redefine the OPTIMIZATION constant below
'from OPTIMIZE_FOR_SPEED to OPTIMIZE_FOR_MEMORY.

'When optimized for memory, VBLM_RTString initializes by loading the Ptrs() array
'with each string's offset in the file, which are then used on subsequent calls
'to fetch strings "from disk."  I use the quotes here because if the host
'system is using a disk cache, which it probably is, fewer than 1 in 10 calls
'are apt to cause an actual read; the other 9 will be in the cache

    Const OPTIMIZE_FOR_MEMORY = 0
    Const OPTIMIZE_FOR_SPEED = 1
    Const OPTIMIZATION = OPTIMIZE_FOR_SPEED

'=====================================================================
' STATIC VARIABLES

'Handle is the database file handle
'It is also used as the initialization flag
    Static Handle As Integer

'Ptrs() hold string location data when optimized for memory
    Static Ptrs() As Long

'Strings() hold actual strings when optimized for speed
    Static Strings() As tagVBLM_VS

'=====================================================================
' TRANSIENT VARIABLES USED ONLY ON FIRST CALL (INITIALIZATION)
'
'NumLanguages = number of languages in the database
    Dim NumLanguages As Integer

'NumStrings = number of entries in each language table
    Dim NumStrings As Long

'i = for-next counter variable
    Dim i As Long

'PreviousMousePointer = MousePointer Cache Variable
    Dim PreviousMousePointer As Integer

'SelectedLanguage = Language Selected by user or command line
    Dim SelectedLanguage As Integer

'FileName = Full path and filename of language database file
    Dim FileName As String

'Offsets() = location in file of beginning of each language table
    ReDim Offsets(0) As Long

'Languages() = Names of Languages in the the database
    ReDim Languages(0) As tagVBLM_VS

'=====================================================================
' TRANSIENT VARIABLE USED ON ALL CALLS WHEN OPTIMIZED FOR MEMORY

'vsTmp = tmp var-length string data type, used to read from disk
    Dim vsTmp As tagVBLM_VS

'=====================================================================
' EXECUTABLE CODE BEGINS HERE
'=====================================================================
'INITIALIZATION CODE: EXECUTES ONLY ON FIRST CALL
'=====================================================================

'Handle is used as the initialization flag
    If Handle = False Then

'Default Error handling
        On Error GoTo RTS_Error
```

```
'cache the current cursor
        PreviousMousePointer = Screen.MousePointer

'opening the file is in a sub in case we need to call it again
        GoSub OpenDataBaseFile

'get the number of languages and redim name and offset arrays
        Get #Handle, , NumLanguages
        ReDim Languages(NumLanguages), Offsets(NumLanguages)

'get the name and offset of each language table
'while iterating, check for a command line match, flag = "/L="
        For i = 1 To NumLanguages
            Get #Handle, , Languages(i)
            Get #Handle, , Offsets(i)
            If InStr(1, Command$, "/L=" & Languages(i).String, 1) Then SelectedLanguage = i
        Next

'if language not specified on command line, query the user
        If SelectedLanguage = False Then

'load the rts support form, and fill in the list of language choices
            Load frmVBLM_RTS
            For i = 1 To NumLanguages
                frmVBLM_RTS.lstLanguages.AddItem Languages(i).String
            Next

'center it on the screen, set an arrow cursor, show it modally
            frmVBLM_RTS.Move (Screen.Width - frmVBLM_RTS.Width) \ 2, (Screen.Height -
frmVBLM_RTS.Height) \ 2
            Screen.MousePointer = 1
            frmVBLM_RTS.Show 1

'get the selected language and unload
            SelectedLanguage = frmVBLM_RTS.lstLanguages.ListIndex + 1
            Unload frmVBLM_RTS

        End If

'look busy
        Screen.MousePointer = 11

'get the number of strings in a language table
        Get #Handle, , NumStrings

'and, depending on optimization method, make room either for strings or pointers
        If OPTIMIZATION = OPTIMIZE_FOR_SPEED Then
            ReDim Strings(NumStrings)
        ElseIf OPTIMIZATION = OPTIMIZE_FOR_MEMORY Then ReDim Ptrs(NumStrings)
        End If

'seek to the beginning of the selected table
        Seek Handle, Offsets(SelectedLanguage)

'and for each string
'either retrieve its value into Strings() or its location into Ptrs()
        For i = 1 To NumStrings
            If OPTIMIZATION = OPTIMIZE_FOR_MEMORY Then Ptrs(i) = Seek(Handle)
            Get #Handle, , vsTmp
            If OPTIMIZATION = OPTIMIZE_FOR_SPEED Then Strings(i) = vsTmp
        Next

'if we've read and saved the strings, close the file
'otherwise we need to keep it open
        If OPTIMIZATION = OPTIMIZE_FOR_SPEED Then Close Handle

'restore the original cursor state
        Screen.MousePointer = PreviousMousePointer

    End If
```

```
'====================================================================
' END OF INITIALIZATION CODE
' FOLLOWING CODE EXECUTES ON ALL CALLS TO RETURN THE STRING
'====================================================================

'only two likely errors, so deal with them as needed
    On Error Resume Next

    If OPTIMIZATION = OPTIMIZE_FOR_SPEED Then

'return string from array
        VBLM_RTString = Strings(Index).String

'possible error: index out of range; so indicate
        If Err = 9 Then VBLM_RTString = "Invalid Index"

    ElseIf OPTIMIZATION = OPTIMIZE_FOR_MEMORY Then

'read string from disk
        Get #Handle, Ptrs(Index), vsTmp
'possible error: bad handle, somebody's "Close" elsewhere closed our file
        If Err = 9 Then
            vsTmp.String = "Invalid Index"
        ElseIf Err = 52 Then
            Err = 0
            GoSub OpenDataBaseFile
            Get #Handle, Ptrs(Index), vsTmp
            If Err Then vsTmp.String = "Unable to retrieve string"
        End If

        VBLM_RTString = vsTmp.String

    End If

    Exit Function

'====================================================================
' END OF MAIN FUNCTION CODE
'====================================================================

'====================================================================
' opendatabasefile sub-procedure
'====================================================================

OpenDataBaseFile:

'grab a handle
        Handle = FreeFile

'look for file in application directory and open as binary
    FileName = App.Path
    If Right$(FileName, 1) <> "\" Then FileName = FileName & "\"
    FileName = FileName & RTS_FILE

'if file not found, terminate
'you can gussy this up as desired
    If Dir$(FileName) = "" Then
        MsgBox "Fatal Error: Language database file " & FileName & " not found.", MB_STOP
        End
    End If

    Open FileName For Binary As Handle
    Return

'====================================================================
' default error handler
'====================================================================

RTS_Error:
    Select Case MsgBox(Error$ & "(Code" & Str$(Err), MB_ICONEXCLAMATION + MB_ABORTRETRYIGNORE,
"VBLM_RTString()")
```

```
    Case IDABORT
        End
    Case IDRETRY
        Resume
    Case IDIGNORE
        Resume Next
    Case Else
    End Select

End Function
```

General Options

VBLM's general options, accessed from the main window's option menu, control a handful of high-level behaviors. They include the following:

Beep for Attention

If checked, VBLM beeps whenever it encounters an error or needs attention. Otherwise it waits silently.

Log Error Info to Disk

If checked, VBLM will record information about any errors it encounters in the error log file on disk. Indispensable for resolving problems, always turn this on and replicate problems before calling for technical support.

Gray Displays

If checked, all windows are given that grey sculpted look that is all the rage these days. It also slows down screen painting, though not very much. If not checked, you get plain old flat white windows.

Animate Map

If checked, VB travels around the world on VBLM's main window until a project is loaded. If not checked, VB stays home.

Temp Directory

The directory where VBLM writes temporary files. This defaults to your TEMP directory if one is set, to the root directory if one is not.

Help System Interaction

These three options control the automatic positioning of VBLM's Help Window. See Positioning the Help Window for details.

Positioning the Help Window

VBLM can control the size and location of its Help window. The Help System Interaction options, set on the General Options window accessed from the main options menu, determine whether and how it does so.

If **Position Help Window** is checked, help requests are packaged with resizing messages that fit the Help window between the edge of the screen and the edge of the calling window.

If **Slide Calling Window to Left** is checked, the calling window is moved to the left edge of the screen before help is requested, and the Help window is placed to its right. If not checked, the calling window is not moved and Help will be placed on whichever side is furthest from the edge of the screen.

If the width available is less than the value you set for **Minimum Help Window Width**, expressed as a percentage of total screen width, the Help window will be sized to this minimum.

If **Position Help Window** is not checked, the current size and position of the help window is not changed and the other options have no effect.

VB Language Manager's Main Window

VBLM's main window displays a stylized world map until you load or create a project. If the Animate Map option is checked, VB travels around saying hello.

After a project is created or loaded, the main window displays current project information at the top and the Language Table Frame at the bottom. If the window is somewhat wider than the frame, the world map reappears to its right.

The main window is resizable (although it has a minimum size) and the information display resizes proportionately.

Main window size and position are retained when you exit VBLM and restored on entry.

You access VBLM's functions via the Main File Menu, the Main Options Menu, and the command buttons contained in the Language Table Frame

For an overview of what these functions are, see VB Language Manager Overview

Note: To minimize memory and resource use, VBLM uses graphics methods rather than persistent images on all windows. In the rare event that a window fails to repaint itself properly, double-click on it to force an immediate repaint

Language Table Editor Print Window

The LTE Print Window pops up prior to print jobs and allows you to specify what portion of the current language table you wish to print.

You may select your print range by pages, by language objects, or by setting a selection condition.

If you specify **Selected Objects**, the Select Objects to Print Window will pop up as soon as you click OK.

Whatever your selection criteria, your print job will be formatted as specified by current settings for LTE Print Options.

Language Table Editor Print Options

The LTE Print Options, accessed from the LTE File Menu, give you precise control over the format of language table print jobs.

These options are very similar to the LTE Display Options, and you may be wondering why they even exist. In this day and age of WYSIWIG, why does VBLM allow you to display and print in independent formats, *ie* to create non-WYSIWIG hardcopy?

Why Not Just WYSIWYG?

WYSIWIG is nice, and VBLM makes it readily available: just click the **WYSIWYG** button, which synchronizes print options with display options, and be done with it. When might you NOT want WYSIWYG hardcopy? Two situations:

1) Working on-screen and on-paper are two different activities, and you may find that different formats are most appropriate for each medium. For example, we found that when we were printing tables to use as worksheets, we consistently wanted a higher number for Line Spacing to give more room to scribble. On the screen, this just wasted space.

2) Not everybody in the world has a good graphics-capable laser printer yet, and if you don't, you probably don't want VBLM instructing your printer to print vertical lines or do gray shading; it will take forever and look horrible.

So that's why.

Select a specific LTE Print Option Group for more information:

LTE Print Options: Format

LTE Print Options: Miscellaneous

LTE Print Options: Margins

See Also:

Selecting a Print Range

Selecting Objects to Print

LTE Display Options,

LTE Print Options: Format

The LTE Print Format, accessed from the main or LTE options menu, sets the overall format used to print language tables.

The three choices -- **Columns**, **2 rows per entry**, and **3 rows per entry** -- allow you to maximize the number of entries on a page (columns) or the amount of space allotted to each item (3 rows per entry), or to compromise with 2 rows per entry.

When **Columns** is selected, each entry is printed on a single line and appears as

Object Name|Original String|Translated String

When **2 Rows per Entry** is selected, each entry is printed as

Object Name=Original String

[Context: Code Context]

Translation: Translated String

When **3 Rows per Entry** is selected, each entry is printed as

Object: Object Name

Orig: Original String

[Context: Code Context]

Trans: Translated String

If your format is not set to **Columns** and you have requested Code Context information, it will be printed with each item in the table.

Whatever format you select, the **Line Spacing** value sets the spacing between lines of text and can be used to add additional whitespace. The value must be >= 1.

LTE Print Options: Miscellaneous

The Miscellaneous LTE Print options, accessed from either the main or the LTE options menu, control nitty-gritty formating of language table hardcopy. The miscellany includes:

Gray Bars

If checked, LTE prints table entries with alternating white and grey background; improves readability.

Percent Gray

Specify the grey shade for printing (100%=black). The default value of 4% works well with HP Laserjets, but your printer may produce more legible output with a different setting.

Vertical Lines

If checked, LTE prints vertical lines between columns.

Horizontal Lines

If checked, LTE prints horizontal lines between entries.

Show File Names

If checked, LTE prepends form and file names to object names; see object names for more information.

Show Context

If available and checked, each item in printed language tables includes the line of code in which the language object is used; not available when using **Column** format. See code context information for more information.

Use Quotes

If checked, LTE encloses all strings in double quotes; useful for seeing leading and trailing whitespace.

LTE Print Options: Margins

The margins for LTE hardcopy, set on the LTE Print Options window and accessed from the LTE File Menu, can be adjusted from 0.25" to 2.0" in increments of 1/8", using the scroll bars.

Note: If you have VBLM Pro, the margins set for LTE printing are also applied to width-check reports.

Selecting Objects to Print

VBLM allows you to define a print range based on the contents of language object elements.

If you designate **Selected Objects** as your print range, the Select Objects to Print Window pops up after you click OK in the Print Window. This is where you set your criteria. It is an almost exact replica of the Search Window and works in similar fashion.

Three Examples:

1) To print all "Caption" properties:

   a) Type "caption" in the **Select Objects Where** combo box (without the quotes).

   b) Click "Object Name" in the **Is Found In** frame.

   c) Click OK

2) To print all language objects which have not been translated

   a) Leave the **Select Objects Where** combo box empty

   b) Click "Object Translation" in the **Is Found In** frame.

   c) Click OK

3) To print all strings used as MsgBox arguments (only possible if Code Context Information is available):

   a) Type "MsgBox" in the **Select Objects Where** combo box (without the quotes).

   b) Click "Code Context" in the **Is Found In** frame.

   c) Click OK.

If no matches are found, VBLM allows you to select again. If matches are found, VBLM tells you how many and asks if you wish to print them.

The **Whole Words Only** and **Case Sensitive** options condition the matching process as expected.

See Also: Selecting a Print Range

String Extraction Options

VBLM's String Extraction Options, accessed from the main options menu, give you a degree of control over which strings VBLM extracts from the parent VB project.

With the exception of the Binary Project Files option, the string extraction options are a set of limits that prevent VBLM from extracting every item in the project that is enclosed in double quotation marks:

The Properties to Ignore list allows you to systematically ignore properties by name.

The Strings to Ignore options allow you to systematically ignore strings by content or type

Note that these limits are a convenience, not a necessity, because you do not need to translate every string that VBLM extracts. Any string extracted but left untranslated will not be touched when VBLM builds a new version. You do not need to be concerned that VBLM extract only those strings that you want translated. The extraction options simply keep clutter out of the language tables.

Properties to Ignore

VBLM's **Properties to Ignore** list, a string extraction option accessed from the main options menu, allows you to tell VBLM to ignore properties by name.

The purpose of this list, maintained with the **Add, Edit** and **Del** commands, is to let you systematically ignore properties that have string values but don't require translation.

When you instruct VBLM to create a new project or update an existing one, any string directly assigned to a property on this list will NOT be extracted for translation. This applies to direct assignments made in code, as well as in the form definition section. See the ignored properties example.

By default, VBLM has the following (mostly database and file-related) properties on this list:

DatabaseName DataField DataSource DefaultExt FileName Filter FontName InitDir LinkItem LinkTopic PasswordChar Path Pattern RecordSource ShortCut Sort SourceDoc Tag

If you are translating into a language that uses a different alphabet, you will most likely want to remove **FontName** from this list, so that you can define your substitute fonts.

See Also

String Extraction Options

Strings to Ignore

Ignored Properties: An Example

When you instruct VBLM to create a new project or update an existing one, any string directly assigned to a property on the Properties to Ignore list will NOT be extracted for translation. This applies to direct assignments made in code, as well as in the form definition section. It does not, however, apply to strings assigned indirectly in code.

Say, for example, that you have left **FontName** on the list (it is there by default).

As VBLM reads the definition sections of your form files, it will not extract any FontName strings.

As VBLM reads code in form and module files, It will not extract any strings directly assigned to FontName; "MS Sans Serif" will not be extracted from the following line of source code:

```
Form1!Text1.FontName = "MS Sans Serif"
```

It will, however, extract "MS Sans Serif" from the following code, because the assignment is indirect:

```
Face$ = "MS Sans Serif"

Form1!Text1.FontName = Face$
```

Strings to Ignore

VBLM's Strings to Ignore options, all string extraction options accessed from the main options menu, allow you to systematically ignore strings by content or type. Select a specific option for more information:

Punctuation Only

Quoted Numbers

Format Strings

SendKey Strings

See Also

String Extraction Options

Properties to Ignore

Strings To Ignore: Punctuation Only

If the **Punctuation Only** string extraction option is checked, VBLM does not extract strings composed only of punctuation characters, "###,##" for example. VBLM uses Window's **IsCharAlphaNumeric** API function to perform the test.

For more information on the language-sensitive **IsCharAlphaNumeric** function, look it up in Windows' SDK help file.

Strings To Ignore: Quoted Numbers

If the **Quoted Numbers** string extraction option is checked, VBLM does not extract strings composed only of characters that do NOT pass Window's **IsCharAlpha** API function, *ie* strings composed solely of numeric and punctuation characters.

For more information on the language-sensitive **IsCharAlpha** function, look it up in Windows' SDK help file.

Strings To Ignore: Format Strings

If the **Format Strings** string extraction option is checked, VBLM does not extract any string that is used in source code as the direct argument to VB's Format() and Format$() functions.

Note that the **Format Strings** option works only where the string literal is directly the argument; it does not work if the literal is assigned to a string variable and this variable is then passed to Format. Hence the string "mm/dd/yy" is not extracted from this code:

```
Print Format$(Now, "mm/dd/yy")
```

but is extracted from this code:

```
Fmt$ = "mm/dd/yy"

Print Format$(Now, Fmt$)
```

Strings To Ignore: SendKey Strings

If the **SendKey Strings** string extraction option is checked, VBLM does not extract any string that is used in source code as the direct argument to VB's SendKeys procedure.

Note that the **SendKey Strings** option works only where the string literal is directly the argument; it does not work if the literal is assigned to a string variable and this variable is then passed to SendKeys. Hence the string "%{F4}" is not extracted from this code:

```
SendKeys "%{F4}", True
```

but is extracted from this code:

```
Keys$ = "%{F4}"

SendKeys Keys$, True
```

Using 3rd Party Translation Applications (Pro Only)

If you have the Professional edition of VBLM and a third party translation application that can be commanded to translate the contents of the clipboard, VBLM Pro can be configured to work with it automatically (Even if your 3rd party program cannot translate the clipboard, you still may be able to use it; see Importing & Exporting).

To configure VBLM to work with your 3rd party program, select **Translation Apps** on the main window's options menu and record the appropriate information in the Interface Definition Window that pops up.

Once you have defined your application's interface, it is accessible from the Language Table Editor's menu system:

- Select **Translate** from the LTE **File** menu, and VBLM will cycle through the entire current table, calling and commanding your 3rd party application to translate every item.
- Select **Translate** from the LTE **Edit** menu, and VBLM will call your 3rd party application and translate the currently selected object.

For more information:

Translation Application Interface Definition

Translating the Clipboard

Translating the Current Table

Translating the Clipboard (Pro Only)

If you have configured VBLM Pro to work with one or more 3rd party translation programs, selecting **Translate** on the LTE Edit menu pops up the Clipboard Translation window in preparation for a translation request to your 3rd party program.

If you have defined multiple 3rd party programs, the first time this window appears you need to tell VBLM which one to use by selecting its title on the **Translation Application** list. If you wish to change the application's interface definition, click **Setup.**

The text of the current language object is assigned to the **Source Text to Translate**. You may edit this if you wish to translate something else.

The command buttons trigger the following actions:

Translate, Paste & Close

VBLM:

1) puts the source text on the clipboard;

2) starts the 3rd party app if it is not already running;

3) sends it the clipboard translation keystrokes;

4) depending on how you have defined the interface, waits either for the app's clipboard translate window to open and close, or for the specified number of seconds;

5) pastes the clipboard into the current language object's translation field;

6) and returns to the Language Table Editor.

Just Translate

VBLM performs the translation as above, but pastes the clipboard into the **Translated Text** field and leaves the Clipboard Translation Window active.

Paste & Close

VBLM pastes the contents of the **Translated Text** field into the current language object's translation field, and returns to the Language Table Editor.

Help

Starts Help at this Topic

Cancel

Returns to the Language Table Editor without making any changes

The Remove Ampersands Option

Many strings in VB projects contain ampersands (&) to create hot-key captions. If this option is checked:

1) VBLM removes all ampersands from the source text before copying it to the clipboard for translation.

2) If the original string had a leading ampersand, VBLM prepends one to the translation.

If this option is not checked, strings will be passed as is, and hot-key ampersands are almost guaranteed to confuse the translation program.

See Also:

3rd party translation programs

Translation Application Interface Definition

Translating the Current Table

Translating the Current Table (Pro Only)

If you have configured VBLM Pro to work with one or more 3rd party translation programs, selecting **Translate** on the LTE File menu pops up the Table Translation window in preparation for a cycle of translation requests to your 3rd party program.

If you have defined multiple 3rd party programs, the first time this window appears you need to tell VBLM which one to use by selecting its title on the **Translation Application** list. If you wish to change the application's interface definition, click **Setup.**

The **Source Text to Translate** text box reads "[Current Table]" and is disabled.

The 4 command buttons have the following effects:

Do Entire Table

1) VBLM starts the 3rd party app if it is not already running, and attempts to move it into the lower right quadrant of the screen (it does so by looking for a window with the application's title).

2) VBLM relocates the Table Translation Window to the upper left screen quadrant (these window relocations are made to keep VBLM's **Cancel** button accessible during the cycle that is about to begin, should you wish to cancel)

3) For each object in the current language table, VBLM:

1) copies its original value to the clipboard;

2) sends the 3rd party app the clipboard translation keystrokes;

3) depending on how you have defined the interface, waits either for the app's clipboard translate window to open and close, or for the specified number of seconds;

4) pastes the clipboard into the current language object's translation field;

5) compares the iteration # to the App Exit Frequency #, and if a match is found shuts down and restarts the 3rd party app.

4) When all objects have been translated, VBLM shuts down the 3rd party program, closes the Table Translation Window, and returns to the Language Table Editor.

Do Untranslated Only

VBLM performs the translation as above, but only translates items with no text in the translation field.

Help

Starts Help at this Topic

Cancel

If you have not begun the translation cycle, pressing cancel causes VBLM to returns to the Language Table Editor without making any changes.

If you have begun the translation cycle, VBLM interrupts it and attempts to shutdown the 3rd party program before returning to the Language Table editor. However, you may need to shut it down manually.

The Remove Ampersands Option

Many strings in VB projects contain ampersands (&) to create hot-key captions. If this option is checked:

1) VBLM removes all ampersands from source text before copying it to the clipboard for translation

2) If the original string had a leading ampersand, VBLM prepends one to the translation.

If this option is not checked, strings will be passed as is, and hot-key ampersands are almost guaranteed to confuse the translation program.

See Also:

3rd party translation programs

Translation Application Interface Definition

Translating the Clipboard

Translation Application Interface Definition (Pro Only)

If you have a third party translation application that can be commanded to translate the contents of the clipboard, VBLM Pro can be configured to work with it automatically.

The Translation Application Interface Definition Window (whew!), accessed by selecting **Translation Apps** on the main options menu, is where you do so.

The discussion of HOW to do so is best covered in two topics:

Basic Operation, which explains the general manner in which VBLM works with 3rd party programs, and;

How to Configure, which gets into the practical details with an example.

Translation Application Interface: Basic Operation (Pro Only)

Rather than hardcoding the commands of specific translation apps into VBLM Pro, we opted to program a general purpose structure that you can fill in for as many apps as you want. The only limitation is that the 3rd party apps MUST BE ABLE TO TRANSLATE WHAT'S ON THE CLIPBOARD (If your 3rd party program cannot translate the clipboard, you still may be able to use it; see Importing & Exporting).

For VBLM to control an app, it needs to know:

1) How to start it, *ie* what is the full path and filename of the executable file?

2) How to activate it, *ie* what is the app's title as it appears on Window's Task List?

3) How to command clipboard translation, *ie* what keystrokes should be sent?

4) How to determine when the translation is complete (this is more involved and is discussed below).

5) How to shut it down, *ie* what keystrokes should be sent?

Most of these items are straightforward enough to require no explanation, but #4, how to tell when the translation is complete, is tricky. VBLM offers two possible methods.

1) If the app always pops up a specific titled window while translating and closes it when done, VBLM can wait for this window to appear and disappear. This is by far the superior method, because it verifies the translation and wastes no time.

2) If the app does not create and destroy such a window, you can tell VBLM how many seconds to wait after sending the clipboard translation keystrokes. This is a kluge, because it doesn't verify the translation and you need to set the delay for the longest possible translation, which will waste time. However, it does allow VBLM to control such an app.

We also discovered the need to accommodate a quirk that appears in at least one publisher's translation apps. The versions of MicroTac's Language Assistant series that we tested eat memory over time and don't give it up; we discovered this during the automatic translation of a large table, when Windows slowed to a crawl and eventually locked up with a GPF. We found that we could avoid this problem simply by periodically shutting down and restarting the apps. Hence we added the "Exit Frequency" parameter to the interface definition. If you set this at 20, for example, VBLM will restart the app after every 20 translations.

Enough on basic operation. Check out How to Configure

Translation Application Interface: How to Configure (Pro Only)

If you have a third party translation application that can be commanded to translate the contents of the clipboard, VBLM Pro can be configured to work with it automatically.

The Translation Application Interface Definition Window (whew!), accessed by selecting **Translation Apps** on the main window's options menu, is where you do so.

If you haven't already read Basic Operation, you should do so before reading further in this topic.

To configure VBLM to work with a new 3rd party app (for example, Micro-Tac's Italian Assistant):

1) Click the **Add** button. Note that the button's caption changes to "Record" and the **Edit** button's caption changes to "Cancel Add."

2) Enter the app's title in the **Application Title** box. Aside from case, you must enter the title EXACTLY as it appears in Windows' task list when the app is running. For the Italian Assistant, enter "MicroTac Italian Assistant" (without the quotes).

3) Enter the app's full path and file name in the **Application Path & EXE Name** box. For the Italian Assistant, enter "c:\italwin\italasst.exe" (without the quotes).

4) If the app supports the titled window method (see Basic Operation), enter the window title in the **Translate Clipboard Window Title** box. For the Italian Assistant, which does, enter "Translate Clipboard" (without the quotes).

5) If the app does not support the titled window method, enter the number of seconds to wait in the **Translate Clipboard Delay** box. Otherwise, as with the Italian Assistant, enter "0" or leave the box blank. Putting any value in this field will cause VBLM to use the inferior delay method.

6) In the **Translate Clipboard Keystrokes** box, enter the keystrokes that command the app to translate the clipboard, using standard VB **SendKeys** format. For the Italian Assistant, enter "%TTE" (without the quotes).

7) Do likewise for the **Application Exit Keystrokes**. For most apps, including the Italian Assistant, enter "%{F4}" (without the quotes).

8) If you need to (see Basic Operation), enter the **Exit Frequency**. For MicroTac, 20 is a good number.

9) To save your work, click the **Record** button. To lose your work, click **Cancel Add**. You're done!

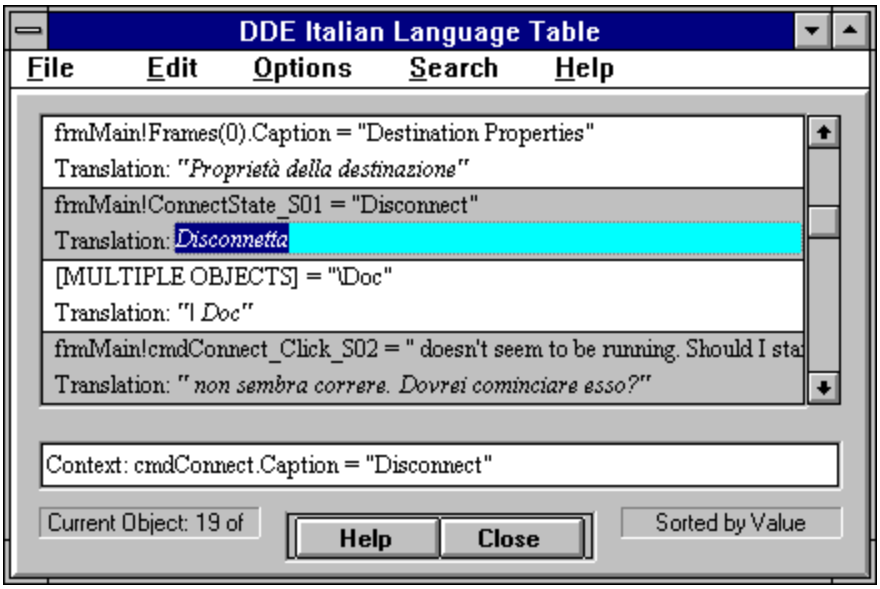
If you later want to edit the interface definition, select the app on the list and click **Edit**.

If you later want to delete the interface definition, select the app on the list and click **Delete**.

The Language Table Editor

The Language Table Editor (LTE), accessed by clicking the **Add** or **Edit Table** commands on the main window, is the interactive heart of VBLM. The LTE displays the language objects contained in the parent VB project and allows you to input translations.

Click the LTE element or topic for which you'd like more information:



The LTE File Menu

The LTE Edit Menu

The LTE Options Menu

The LTE Search Menu

The LTE Keyboard

The Language Table Display

The LTE is dominated by the language table display. The appearance of this display depends entirely on how you have set display options, accessible from the editor's option menu.

If you have selected **Columns** as the display format, there will be column titles above the display. You may adjust column width by dragging the separator bars between these titles.

Whatever the display format, each object in the table contains three elements: the name of the object, the original value of the object (*ie*, the string), and the translated value of the object.

The syntax of the object name displayed depends on object type and on whether more than one object in the parent project share the same string: you need to double-click on [MULTIPLE OBJECTS] to see the individual instances of a string used more than once.

To enter translations, simply click anywhere on the item you wish to translate and type it into the floating text box.

Language Table Editor Display Options

The LTE Display Options, accessed from either the main or LTE options menu, give you precise control over the LTE's display format. While discussed below, the best way to get a feel for their effects is simply to fiddle around.

Fonts

The font settings determine the font name, size and attributes that the LTE uses to display the language table. See LTE Fonts.

Format & Line Spacing

The format options sets the overall display format. See LTE Format.

Edit Box Options

The Edit Box Color and Select Text options control the behavior of the floating text box used to input translations. See LTE Edit Box.

Miscellaneous

The miscellaneous display options are a potpourri. See Miscellaneous LTE Display & Print Options.

LTE Display Options: Miscellaneous

The Miscellaneous LTE Display options, accessed from either the main or the LTE options menu, control the nitty-gritty format used to display language tables. The miscellany includes:

Gray Bars

If checked, LTE displays table entries with alternating white and grey background; improves readability.

Vertical Lines

If checked, LTE draws vertical lines between columns.

Horizontal Lines

If checked, LTE draws horizontal lines between entries.

Show File Names

If checked, LTE prepends form and file names to object names; see object names for more information.

Show Context

If available and checked, the LTE displays the line of code in which the currently selected language object is used; see code context information for more information.

Use Quotes

If checked, LTE encloses all strings in double quotes; useful for seeing leading and trailing whitespace.

See Also:

LTE Display Options

LTE Print Options

Language Table Editor Fonts (Display & Print)

The LTE Font settings, accessed from either the main or the LTE options menu, control the fonts VBLM uses to display and print language tables. Display fonts are set in the Display Options window, print fonts in the Print Options window.

You may set two different fonts -- Main and Translation -- by clicking the appropriate command button and using the dialog box that pops up.

LTE uses the Main fonts to display/print the Object Name, the Original String, and code context information.

LTE uses the Translation font to display/print and input translations. You may set it different from the main font to translate into languages that use a different alphabet for which you have a font (Exceller Software Corporation, for example, offers Cyrillic fonts).

While the Main and Translation font *names* can be different, they must be set to the same *point size*. VBLM ignores attempts to set different point sizes, and always sets the Translation point size to match the Main font size.

LTE Display Options: Format

The LTE Display Format setting, accessed from either the main or LTE options menu, controls the overall format used to display language tables.

The three choices -- **Columns**, **2 rows per entry**, and **3 rows per entry** -- allow you to maximize the number of entries on a screen (columns) or the amount of space allotted to each item (3 rows per entry), or to compromise with 2 rows per entry.

When **Columns** is selected, column titles appear (you can adjust column width by dragging the separator bars between them), and each entry appears as

Object Name|Original String|Translated String

When **2 Rows per Entry** is selected, each entry appears as

Object Name=Original String

Translation: Translated String

When **3 Rows per Entry** is selected, each entry appears as

Object: Object Name

Orig: Original String

Trans: Translated String

Whatever format you select, the **Line Spacing** value sets the spacing between lines of text and can be used to add additional whitespace. The value must be >= 1.

See Also:

LTE Display Options

LTE Print Options

The LTE's Floating Edit Box

The Language Table Editor has a single control (a text box) in which you enter translations.

The LTE Edit Box floats around the table, anchoring on whichever item you select and resizing to fit in the space allotted for translations. This space is determined by the current LTE Format and Line Spacing options.

The Edit Box uses the Translation font (see LTE Fonts).

Other LTE display options also affect its appearance and/or behavior:

Edit Box Color sets the background color.

Select Text on Focus

If this option is checked, all text in the edit box (*ie* any translation you've entered previously) is selected when you click an item in the table. If not checked, text is not selected. In either case, the caret is positioned at the end of the text.

See Also:

LTE Display Options

Searching a Language Table

The Search Window allows you to search any or all elements in a language table for specific strings.

To Search:

1) Enter or select the string to find in the **Find What?** combo box (VBLM retains previously searched-for values during a session).

2) Select the Table elements you wish to search in the **Look In** frame. You may search Object Names, Values, Translations, and Code Context (if available).

3) Click **Search Next** to search forward, **Search Prev** to search backward

Notes:

1) Searching is inclusive, not exclusive; matches are found in any selected element when more than one element is selected.

2) A successful search enables the **Find Next** and **Find Prev** items on the Search Menu.

3) The **Case Sensitive** and **Whole Words Only** options condition searching as you would expect.

Sorting a Language Table

VBLM offers an elaborate set of options for sorting language tables. These options are set on the Language Table Sort Window, accessed either from the main or LTE options menu.

The sort options fall in three categories; select one for more information:

Sort Keys

Sort Method

Pre-Sort String Processing

The current sort keys are always displayed in lower right corner of the LTE.

Notes:

1) VBLM does not store language tables in sorted order; tables are resorted per the current specification when loaded.

2) In the current release of VBLM, sorting is done in VB code and is thus slow. A future release will have faster sort algorithms implemented in a DLL.

Language Table Sort Keys

You may nest-sort a language table alphabetically on up to three keys:

File Name (the source file in which VBLM first found the language object)

Object Type (form or control property vs. string in code)

Object Name (as it appears in the table)

Object Value (the string itself)

Object Translation (the translation you have input)

If you select **No Sort**, table order is determined by

a) the order in which the project files are listed in VB's project window, and

b) the order in which VBLM found the strings in the files.

Language Table Sort Method

The LTE Sort Method, either **Windows** or **ANSI** and Case Sensitive or not, sets the method VBLM uses to sort the table:

Windows Rules

VBLM sorts using Windows' language-sensitive **lstrcmp** (case sensitive) or **lstrcmpi** (case insensitive) API functions. For more information on these functions and the primary and secondary sort values they return, look them up and see the "International Applications Overview" in Windows' SDK help file.

ANSI Rules

VBLM sorts by ANSI code using VB's **StrComp** function, with the case flag set appropriately.

Language Table Pre-Sort Processing

The pre-sort string processing options -- **Ignore Leading Spaces, Ignore Leading Punctuation**, and **Ignore Embedded Punctuation** -- are included to compensate for the idiosyncracies of strings used to define VB properties and/or embedded in VB code.

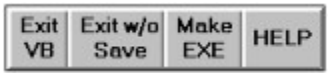
In particular, embedded strings often have leading spaces for concatenation reasons, and property strings often have leading or embedded ampersands for hot-key reasons. Checking any of these options causes VBLM to remove the ignored item(s) from the temporary copies of the strings used for sorting.

Note: Checking **Ignore Embedded Punctuation** causes VBLM to reconstruct the temporary copies of strings used for sorting character by character, and can significantly slow the already less-than-breathtaking speed of VBLM's current sort algorithms.

Status Window

The Status Window pops up to provide feedback whenever VBLM is engaged in a lengthy task. The window indicates the current task and, if possible, the percent complete.

The VB Control Window



When the **VB Control Window**, pictured above, appears on the lower left corner of your screen, it indicates that VBLM started VB under its own control but has gone to sleep and turned control over to you. This occurs when VBLM has finished processing a project and either:

a) The Interaction with VB option is set to **Load Project**, or

b) VB reports a problem trying to load or compile the translated code and you click "Take Control of VB" on the VB Error Window. In this situation, the "Make EXE" caption shown above will read "Retry EXE" instead.

The pseudo command buttons on the **VB Control Window** have the following effects:

Exit VB

VBLM instructs VB to exit, but leaves to you the responses to any VB "Save Changes?" dialog boxes.

Exit w/o Save

VBLM shuts down VB, responding **NO** to any VB "Save Changes?" dialog boxes.

Make EXE / Retry EXE

VBLM instructs VB to compile the project. If VB succeeds, and if Interaction with VB is set to **Compile & Exit**, VBLM shuts down VB and resumes execution.

HELP

Starts Help at this topic.

Main File Menu

The selections on VBLM's main File menu trigger the following actions:

New Project

VBLM pops up a file dialog set to look for VB project files (*.MAK). When you select a project, VBLM gives you an opportunity to tweak the string extraction options and then opens the VB project, extracts the strings, and pulls them into a new VBLM project.

Open Project

VBLM pops up a file dialog set to look for VBLM project files (*.LMP). When you select a project, VBLM loads it and paints the main window with Current Project Information.

Update Project

VBLM initiates a four-stage process intended to insure that VB project data stored in the VBLM project is up-to-date. See updating a VBLM project for more information.

The Update Project menu entry is enabled only when a project is loaded.

Save Project, Save Project As

VBLM saves the current project to disk. If the project has not been previously saved, or if you select "Save Project As," you are given the opportunity to name the project file. These items are enabled only when a project is loaded.

Exit

VBLM notifies you of any unsaved changes, saves all current option settings in its initialization file, and shuts down.

Recent Files...

In addition to these static menu entries, VBLM maintains the 4 most recently saved projects at the end of the file menu. Selecting a file causes VBLM to load it.

Main Options Menu

The main window's option menu gives you access to all of VBLM's various options. Click the option group for which you'd like more information:

General Options

String Extraction Options

Language Table Editor: Display Options

Language Table Editor: Sort Options

Language Table Editor: Print Options

Project Build Options

Width Checking (Pro Only)

Translation Applications (Pro Only)

The Help Menu

The selections on VBLM's main and LTE Help menus trigger the following actions:

Contents

VBLM opens its Help System at the Contents topic.

Search Help For...

VBLM opens Help and pops up the Search Help dialog box.

What Is...

Context-Sensitive: VBLM changes the cursor and gives you help on the next item you click.

Technical Support

VBLM opens its Help System at the Technical Support topic.

Using Help

VBLM opens the standard Windows Help on Help file.

About VB Language Manager

VBLM pops up its About Box

See Also: Help WIndow Positioning

Language Table Editor File Menu

Select the LTE File menu item for which you'd like more information:

Save Project, Save Project As

Import & Export

Translate (Pro Only)

Print

Print Options

Printer Setup

Close

LTE File Menu: Save Project, Save Project As

VBLM saves the current project to disk. If the project has not been previously saved, or if you select "Save Project As," you are given the opportunity to name the project file.

LTE File Menu: Translate Table(Pro Only)

If you have configured VBLM to work with one or more 3rd Party Translation Applications, selecting Translate on the LTE File menu pops up the Table Translation Window in preparation to call the program and translate the current language table.

LTE File Menu: Print

VBLM pops up a dialog box in which you specify what portion of the current language table you wish to print, and then sends it to the printer.

LTE File Menu: Print Options

VBLM pops up the Print Options dialog box in which you specify the format for printed output.

LTE File Menu: Printer Setup

VBLM pops up a standard Printer Setup dialog box.

LTE File Menu: Close

VBLM closes the LTE and returns to the main window. Note that you are not prompted to save any changes, because they are always preserved until you exit VBLM altogether.

Language Table Editor Edit Menu

The top section of the LTE Edit menu provides standard commands for editing text and working with the Windows clipboard: **Cut, Copy, Paste** and **Delete**.

Translate (Pro Only)

If you have configured VBLM to work with one or more 3rd party translation programs (see Using 3rd Party Translation Applications), selecting Translate on the LTE Edit menu pops up the Clipboard Translation Window in preparation to call the program and translate the currently selected object (or any text you wish to type).

Language Table Editor Options Menu

The LTE Options menu allows you to control table display format, sort order, and print format. Click the option group for which you'd like more information:

LTE Display Options
LTE Sort Options
LTE Print Options

Language Table Editor Search Menu

The LTE Search Menu gives you access to VBLM's table searching procedures:

Find...

Pops up the Table Search Window where you specify what to search for and where to look.

Find Next / Find Prev

Enabled only after a successful search, these items repeat the last search in the specified direction.

Goto...

Pops up a dialog box in which you input the object number to go to.

Current Project Information

When VBLM has a project loaded, the top half of the main window displays information about the project, and, if present on the host PC, about the parent VB project on which the VBLM project is based. Select an item for details:

Current Project

Date Created

Last Saved

Last VB Update

Parent VB Project

Newest File

Newest File Date

Update Status

Save Status

The lower half of the main window displays related information in the Language Table Frame.

Current Project

The name of the current project file. Unless you override the default naming conventions, the project file name will consist of the name of the parent followed by LMP, VBLM's default project file extension.

Date Created

The date and time when the project was created (Note: VBLM displays all date and time info using the formats defined by the international settings in your WIN.INI file).

Last Saved

The date and time the VBLM project file was last saved to disk.

Last VB Update

The date and time VBLM last examined the parent VB project to see if anything has changed since the project was created. If you have never run an update, this will be the same as the creation date.

Parent VB Project

The full path and file name of the parent VB project on which the current VBLM project is based.

Newest File

If VBLM can locate the parent project on the host PC, the name of the most recently saved file referenced in the parent VB project. If VBLM cannot locate the VB project file or any of the files referenced in it, a message to that effect is displayed instead.

Newest File Date

If VBLM found the parent project, the date and time of the most recently saved file in it. Otherwise a "Not Available" message.

Update Status

The result of comparing the date of the last VB Update and the newest file date:

- If the newest file in the parent VB project is more recent than the last update, the Update Status will read "Out of Date."
- If the last VB Update is the same or more recent than the newest file, the Update Status will read "Up-to-Date."
- If VBLM cannot locate the parent project, the Update Status will read "Not Available."

Save Status

Save Status informs you whether you have made any changes to the current project that have not been saved to disk (VBLM will, of course, prompt you on exit to save any such changes).

Language Table Frame

When a project is loaded into VBLM, the Language Table Frame is displayed in the lower half of the main window, beneath Current Project Information.

The Language Table Frame contains a list of language tables in the project, and a set of command buttons for working with them:

Add New Table

VBLM queries you for a table name, then creates the table, opens the Language Table Editor, and loads the table.

Note: To edit the name of an existing table, select and then right click it.

Edit Table

Open the Language Table Editor and load the currently selected table. Double-clicking a table name on the list has the same effect.

Delete Table

Delete the currently selected table

Width-Check Table (Pro Only)

Perform a width-check on the table: see width checking for more information.

Build Using Table

Open the Build Options dialog box in preparation to build a translated version of the parent VB project.

Help

Context sensitive help; click the special cursor on the item for which you want help.

Exit

VBLM notifies you of any unsaved changes, saves all current option settings in its initialization file, and shuts down. Same as selecting Exit from the main File menu.

VB Language Manager Overview

Welcome to VB Language Manager! VBLM manages the process of translating Visual Basic® applications into different spoken languages. You provide the translations and VBLM does the rest.

Using VBLM is as simple as 1-2-3:

Step 1: String Extraction

Step 2: Translation

Step 3: New Language Build

That 's all there is to using VBLM.

Well, at least in theory. For practical reasons there is a fourth step -- **Update** -- because a VBLM project file contains a lot of data about its parent VB project, and some of this data is almost certain to change over time. See Updating a VBLM Project.

VBLM has many options that you can set to configure and control it, and the Professional Edition has additional features. But these four steps pretty well sum up what it does and how you use it.

See Also: About Documentation

Step 1: String Extraction

During string extraction, VBLM examines your VB project and extracts all of the embedded strings that require translation. To initiate string extraction:

a) click **New Project** on the main file menu;

b) select the VB project you wish to work with;

c) adjust the String Extraction Options as desired;

d) click **OK**

After calling VB to create ASCII copies of any binary files, VBLM reads the files, extracts the strings, and stores them, along with other project data, as a new in-memory VBLM project. When the main window fills with current project information, you're ready to translate.

Step 2: Translation

Once VBLM has extracted the strings from your VB project, you translate them into as many languages as you wish by creating new language tables. You do so using VBLM's customizable Language Table Editor (LTE):

a) click the **Add New Table** button on the main window

b) give the new table a name;

c) when the LTE pops up, customize it to your preferences via the LTE options menu;

d) enter your translations.

If you have VBLM Pro and a 3rd party translation program that supports clipboard translation, you can even translate the table automatically ; see Using 3rd Party Translation Applications.

Step 3: New Language Build

Once you have created a new language table, instruct VBLM to use it to build a translated version of the VB project:

a) click the **Build Using Table** button on the main window;

b) make any desired adjustments in the Build Options dialog box;

c) click **OK**.

If you have VBLM Pro, you can also build runtime switchable multi-lingual versions of your applications.

And **before** doing the build, VBLM Pro can width check your language tables to see if any translated strings are too long to fit in their containers.

Step 4: Updating

During the update process, VBLM re-examines a VBLM project's parent VB project to find out if data stored in the VBLM project is out-of-date. If so, VBLM refreshes the data while preserving the contents of any language tables you have created.

You can manually command an update by selecting **Update** on the main file menu. You will seldom need to, however, because VBLM handles updating automatically. See Updating a VBLM Project.

Updating a VBLM Project

A VBLM project file contains a lot of data about its parent VB project, some of which is almost certain to change over time. Almost any change in the VB source files, for example, obsoletes the string location data that the LTE uses to show you the line of code in which a string is used (see code context information). Hence VBLM allows you (in fact advises you) to update projects as often as necessary.

Updating is a four step process:

Step 1: Date Check

VBLM compares the dates of all VB project files with the VBLM project's Last VB Update date to test for the possibility that data is obsolete. VBLM performs this step whenever you load a project, and advises you if the other steps are necessary.

Step 2: Data Check

VBLM re-examines the parent VB project to see if data has changed.

Step 3: Update

VBLM updates the data stored in the VBLM project.

Step 4: RematchTranslations

Having rebuilt the language data, VBLM compares all new strings with all old strings and, whenever a match is found, reattaches any translations that exist.

When VBLM completes the update, it stores the current date and time in the project's Last VB Update field.

Note that while you can manually run an update from the main window's file menu, you generally won't need to because VBLM handles it automatically.

See Also:

Current Project Information

The Error Log File

When VBLM's **Error Logging** option is checked, error information is recorded in a file named VBLM_ERR.LOG.

If you have a single-user version of VBLM, the error log file is written in the directory where VBLM is installed.

If you have a multi-user network version, the file is written in the local windows directory.

VBLM always appends to the Error Log file, so it grows continuously until manually deleted.

Ordering Information

To order VB LANGUAGE MANAGER, please contact:

QBS Software

10 Barley Mow Passage

London W4 4PH

ENGLAND

Tel: +44 181 994 4842

FAX: +44 181 994 3441

BBS: +44 181 747 1979

Binary Project Files

VB allows you to save your form and module files in ASCII or binary format. VBLM, however, needs ASCII code files to do its work, and thus always begins the string extraction process by firing up VB and instructing it to resave any binary files in ASCII format.

The **Binary Project Files** option, set on the String Extraction Options Window, determines whether these ASCII versions are written to another directory or, if converted in place, whether files thus converted are restored to binary format when processing is complete. It has 3 possible settings:

Convert Copies (default)

VBLM instructs VB to create the ASCII versions in the Temp directory (see General Options), leaving the original files untouched.

Convert & Restore

VBLM converts the original files in place, but ends the processing cycle by instructing VB to restore all converted files to their original binary format.

Convert & Leave

VBLM converts the original files in place and leaves them in ASCII format.

Note: The Language Table Editor will be unable to display code context information for language objects extracted from files left in binary format.

The VB Error Window

The **VB Error Window** appears on your screen if VB has trouble with the translated code that VBLM has just generated and instructed VB to load and compile. The window describes the error, the line of code, and the token that VB highlighted while reporting it (if error logging is active, this information is also logged to disk). The window also asks you to decide what to do next:

Attempt to Continue

VBLM ignores the error and attempts to continue. Unless the error is an old file format alert or other innocuous load-time message, however, VB is apt to continue reporting the error and VBLM will pop the window up again.

Abort VB

VBLM gives up, shuts down VB, completes processing any report you have selected, and returns to ready state.

Take Control of VB

VBLM pops up the VB Control Window, drops into the background, and waits for you to either exit VB manually or reactivate VBLM via the VB Control window.

Help

VBLM starts help at this topic.

Code Context Information

Code context information about any given string consists of the original line of VB source code in which the string is used. During the translation process, this information can be quite useful, so *conditions permitting*, VBLM makes it available within the Language Table Editor:

- To have the line of code for the currently selected object appears at the bottom of the Editor window, check **Code Context** in the Miscellaneous LTE Display options.
- To have the line of code for each string printed along with the string in hardcopy, check **Code Context** and set the style to either 2- or 3-rows per entry in the LTE Print options.

Note the "conditions permitting" caveat, because there are several circumstances in which code context is not available. The reason is that the VBLM project does not actually contain the code; that would violate security and consume too much space. Instead, when VBLM extracts strings it stores the length and offset of the line of code within the file. To retrieve the information, VBLM opens the file in binary mode, seeks (offset) and fetches (length) bytes. For this scheme to work, the following conditions must be met:

1) The parent VB project must be present on the host PC (there is no reason otherwise why VBLM can't be used simply for translation purposes on a different machine).

2) The VBLM project must be up-to-date, as reported on the main window's display of Current Project Information.

3) The relevant VB source file must be stored in ASCII format ("as text").

If any of these conditions are not met, VBLM will inform you that code context info is not available

Object Names

The name VBLM gives an object can take one of several forms:

If the object is a control property set in a form definition section, the name will be *ControlName.PropertyName* or *ControlName(IndexNumber).PropertyName*.

If the object is a string found in a declarations section (*ie* a constant definition), the name will be *Declarations_S##*, where ## is a number indicating the sequence in which it was defined (the first string found is 1, and so forth).

If the object is a string found in a sub or function procedure, the name will be *ProcedureName_S##*, where ##, as above, is a number indicating the sequence in which it was defined.

If the Show File Names Display option is checked, all of these names will begin with the name of the object's source file.

Multiple Objects

VBLM maintains only one copy of any given string, regardless of how many times it occurs in the parent VB project, so you only have to translate it once. VBLM does, however, track each occurrance.

When a string occurs more than once, the Language Table Editor displays "[MULTIPLE OBJECTS]" in place of the object name.

To Display the individual instances, either double-click on "[MULTIPLE OBJECTS]" or type ALT-M, and VBLM will pop up a dropdown list of them.

If Code Context is active, it updates as you scroll through the list.

The multiple object list disappears as soon as you hit Enter or Escape or set the focus elsewhere.

Note: Strings must match exactly, including case, to be considered identical.

Language Table Editor Keyboard & Mouse

Most of the keystroking in the LTE is pretty standard stuff.

The Up and Down arrows move the LTE Edit Box up and down in the table.

Home and End move the insertion point to the beginning and end of the edit box text.

Shift-Home and Shift-End move the edit box to the first and last entries on the screen.

Control-Home and Control-End move the edit box to the first and last entries in the table.

When the current entry has "Multiple Objects," ALT-M opens a dropdown list box of object names. The list box disappears when a) you hit ENTER or ESCAPE, or b) you move the focus elsewhere. You may also open the box by double clicking on "Multiple Objects."

The Vertical Scroll Bar scrolls the table beneath the Edit Box, either one entry (small change) or one screen (large change).

Interaction with VB

The Interaction with VB option, set on the Build Options Window, controls VBLM's interaction with VB when it completes a build. It has 4 possible settings:

None

VBLM returns to the main window without calling VB.

Load Project

VBLM start VB, loads the translated project, pops the VB Control Window into the lower left corner of the screen, and goes to sleep until you exit VB.

Compile Project

VBLM starts VB, loads the project, makes the EXE, pops the VB Control Window into the lower left corner of the screen, and goes to sleep until you exit VB.

Compile & Exit

VBLM starts VB, loads the project, makes the EXE, exits VB, and restores the main window.

Note: Whatever the setting of this option, any problems VBLM encounters while controlling VB causes it to pop up the VB Error Window and ask what you want to do.

The Standard & Professional Editions: What's the Diff?

VB Language Manager comes in two editions, Standard and Professional. The Professional Edition (VBLM Pro) has all features of the Standard Edition, plus:

- The ability to interface with third party translation programs that can translate the clipboard, and command them to automatically translate the strings it extracts. See Using 3rd party translation programs.
- Enhanced import/export capabilities that enable you to work with 3rd party programs that cannot translate the clipboard. See Importing and Exporting
- The ability to automatically width-check language tables to detect appearance and functionality problems caused by variations in string length. See Width Checking.
- The ability to generate run-time switchable multi-lingual versions of VB applications, in which the user selects a language at start-up. See Building Multi-Lingual Projects.

This help file covers both editions, so that owners of the Standard Edition know what they're missing!

About Documentation

Hey WhippleWare! Where's the #*&#?! manual?

There isn't one. Why not? Because we want to provide the most accessible, up-to-date, and generally most useful documentation possible, and we don't think that hardcopy measures up. Instead of writing shelfware, we've put considerable effort into creating an online help system that we believe is superior to hard copy in all respects. OK, all but one; it's harder to read in the bathroom.

Beyond the usual good stuff (you'll find it comprehensive and very context-sensitive), the help system is managed by VBLM to co-operate onscreen, obscuring as little as possible so that items of interest and information about them are visible simultaneously. See Positioning Help for details.

You will work most productively with VBLM by first familiarizing yourself with the operation and contents of the help system. We are pleased to provide support to any customer who has consulted help and still cannot answer their question or solve their problem (see Technical Support). But we might get a little testy with folks who call and obviously haven't checked help.

Don't look for a README file, because the documentation is 100% up to date. But hardcore hardcopy fans will find VBLMHELP.WRI, a Windows Write file with the contents of the help system.

Legal Stuff

VBLM.INI: The Initialization File

VBLM uses the standard Windows profile string functions to read and write all option settings and recent files in VBLM.INI.

If you have the single-user version of VBLM, by default it reads and writes VBLM.INI in the directory where VBLM is installed. If you have a multi-user network version of VBLM, it reads and writes VBLM.INI in the local Windows directory instead. You can override this behavior if you wish: see the /INI command line switch for details.

If you have the Professional Edition, VBLM.INI also contains information about any 3rd party translation programs you have defined.

VBLM.INI also contains a section defining the keystrokes needed to interface with Visual Basic®, which can be manually edited, if necessary, to maintain compatibility with future versions of VB and with international versions of Windows. See Configuring the VB Interface for more information.

Command Line Options

VBLM recognizes and reacts to these command line switches:

Loading a Project on Startup: The FileName Switch

If you start VBLM with the name of a valid VBLM project file on the command line, the project will be loaded on startup.

Adjusting VBLM for Processing Large VB Projects: The /MIM Switch

When VBLM is extracting the strings from a VB project (either during creation or updating), it stores as much data as possible in memory. For projects with string data of ~45K or less, all data fits in memory without problem. For projects with more than ~45K of string data, however, VBLM starts swapping stuff out to disk to avoid out-of-string-space and out-of-memory errors.

There is nothing particularly theoretical about this 45K number; in tests, it just seems to work well. But there are any number of variables that affect the truly optimal point at which VBLM should go to disk, and hence the /MIM switch. MIM stands for "maximum in memory," and this switch allows you to override the 45K default with a value of your own.

The only time you should consider using the /MIM switch is if VBLM blows up or locks up while extracting strings from a large VB project.

When you use the switch, you should only use it to REDUCE the threshhold, not increase it. Try lowering the value by 5K or so.

Syntax: /MIM=40000

Using Alternate INI Files: The /INI Switch

By default, VBC stores and retrieves option settings using the initialization file located in the directory where it is installed (single user version) or in the local windows directory (multi-user).

The **/INI** switch, however, lets you override this behavior by telling VBLM to use a different file and/or directory. To use it, start VBLM with /INI NewPathandFileName on the command line, and it will use that file instead of the default.

This option is most useful if you are installing a single-user version on a network and want individual users to be able to maintain their own configurations.

Note: When using the /INI switch, the file name given MUST include a full path.

Width Checking: Overview (Pro Only)

Different languages vary dramatically in the number of characters required to communicate a given word, phrase, or sentence. English is relatively concise, so translated versions of English text are usually longer. The "International Overview" in WinSDK estimates that length increases by 30% to 200%, depending on original string length: the shorter the string, the higher the % increase.

Whatever your source and target languages, you need to worry about the effect of this variation on the appearance and functionality of your translated application. At minimum, you need to identify all instances where translated captions overflow their containers and become illegible.

Complicating the task is that string length (in characters) is a lousy measure for doing so, because proportional fonts are the rule rather than the exception in Windows, and the width of a string displayed in such a font cannot be predicted simply from its length. In our tests, we found that length changes tended to overestimate width changes in some languages (*eg*Italian) but to underestimate it in others (*eg*German).

Hence the only accurate ways to detect problems are either to run the translated version and find them visually (ugh), or to use WinAPI calls to measure the width of translated strings in the font face and point size in which they will be displayed.

VBLM Pro will do this for you!

1) Select a language table from the list on the main window's Language Table Frame.

2) Click the **Width Check** command button.

3) Make any desired adjustments in the Width Checking Options dialog box that pops up, then click **OK**.

4) View the report in the Width Check Report Viewer.

5) Finally, if you want hardcopy select Print Summary or Print Detail on the viewer's **File** menu.

See Also

Width Checking: Options

Width Checking: Viewing Reports

Width Checking: Printing Reports

Width Checking: Options (Pro Only)

The four Width Checking Options, set in the dialog box that pops up whenever you request a Width Check, control the contents of the report that VBLM Pro will generate when you click **OK**.

The first three options are threshold values for ratios that will be calculated for each translated property (ratio #1) or embedded string (ratios #2 & #3) in the project. The item will be flagged and included in the report whenever the calculated value matches or exceeds the threshold value:

Translated_Property_TextWidth / Control_Width

This ratio, which can be varied from 0.5 to 2.5 and has a default value of 0.9, compares the textwidth of translated control property strings with the width of the control. Width measurements are made using the ScaleMode of the control's container and the control's font, if set. If the control definition does not include a font, the measurement is made using the default font that VB assigns to the control. When set to 1, VBLM reports overflows but not crowding (see diagnosis). Only calculated for control properties, not embedded strings (or menu captions, because menus have no width property).

Translated_String_TextWidth / Original_String_TextWidth

This ratio, which can be varied from 1.0 to 5.0 and has a default value of 1.5, compares the textwidth of translated strings with the textwidth of the original. Width is measured in twips using VB's default font. The ratio is calculated for properties as well as for embedded strings, but exceeding the threshold triggers inclusion in the report only for embedded strings and menu captions.

Translated_String_Length / Original_String_Length

This ratio, which can be varied from 1.0 to 5.0 and has a default value of 1.5, compares the length (in characters) of translated strings with the length of the original. This ratio is calculated for properties as well as for embedded strings, but exceeding the threshold triggers inclusion in the report only for strings and menu captions.

The fourth option **Check Properties / Embedded Strings** is a bit more obvious: VBLM only width checks and reports on the type(s) of strings -- properties or embedded strings -- that you tell it to check.

See Also

Width Checking: Overview

Width Checking: Viewing Reports

Width Checking: Printing Reports

Width Checking: Viewing Reports (Pro Only)

VBLM Pro's Width Check Report Viewer is a simple window with File and Font menus, a list of flagged items, a textbox that fills with detail about selected items, and three command buttons: Rerun, Help, and Close.

File Menu

Use the File menu to print reports and to exit the viewer.

Font Menu

Use the Font menu to select fonts for displaying and printing width check information.

Flagged Item List

The Flagged Item List contains all translated items (control properties and/or embedded strings) that met the criteria specified in the width check options dialog box when the report was created.

Flagged Item Detail

When you click an item on the list, the Detail box fills up with (self-explanatory) information about the item. If the item is a control property, the information includes a diagnosis.

Rerun

The Rerun button pops up the Options dialog, allowing you to rerun the report with different option settings.

See Also

Width Checking: Overview

Width Checking: Options

Width Checking: Printing Reports

Width Checking: Printing Reports (Pro Only)

Aside from the printer font set on the Width Check Viewer's Font menu, VBLM offers only two choices for hardcopy of width check reports: **Summary** or **Detail**. You print a width check report by selecting one of these options from the Viewer's File menu.

Summary Width Check Report

When you request a summary report, VBLM prints out two lines of information for each item on the current Flagged Item list. The first line contains the name of the item and the second reports VBLM's diagnosis for translated properties and the length and width ratios for embedded strings. The summary report provides all of the width check information you really need.

Detailed Width Check Report

When you request a detailed report, VBLM prints out as many lines for each item as it takes to report all of the information that appears in the Viewer's Detail box. If you enjoy stacks of paper, this option is for you.

Note: VBLM prints width check reports using the margin settings specified for printing language tables.

See Also

Width Checking: Overview

Width Checking: Options

Width Checking: Viewing Reports

Width Checking: Diagnosis (Pro Only)

When VBLM Pro width checks a translated control property, it provides a shorthand diagnosis of its conclusion:

OVERFLOW

VBLM diagnoses an overflow when the textwidth of the translation is wider than the control, the textwidth of the original string is narrower than the control, and if a label, AutoSize = FALSE.

POSSIBLE OVERFLOW

VBLM diagnoses a possible overflow when the textwidth of the translation is wider than the control but either the original is also, indicating that wrapping is OK, or if a label, AutoSize = TRUE.

CROWDED

VBLM diagnoses crowding when the textwidth of the translation is 90-99% as wide as the control and the textwidth of the original string is less than 90% as wide.

POSSIBLY CROWDED

VBLM diagnoses possible crowding when the textwidth of the translation is 90-99% as wide as the control and so is the textwidth of the original string.

OK

VBLM diagnoses OK when the textwidth of the translation is less than 90% as wide as the control.

For embedded strings and translated properties of controls without a width property (*eg*menu captions), VBLM reports the Translated/Original textwidth ratio in place of a diagnosis.

LTE Import & Export: Overview

VBLM's LTE import and export commands, accessed via the LTE file menu, allow you to transfer string data between VBLM's binary project files (*.LMP) and ASCII import/export files (*.LMX). LMX files, which you can edit with any text editor, are useful for two purposes:

1) They provide VBLM with a rudimentary dictionary capability. While translations stored in LMP files are tied directly to the parent VB project, those exported to an LMX file can be imported later into any LMP file.

2) They extend VBLM Pro's ability to work with 3rd party programs that do not support clipboard translation. By giving you control over LMX file format, VBLM Pro's export options allow you to create files that, with minimal editing on your part, can be loaded into these programs, translated, and then imported back into the LMP file. For details, see Using Pro's Export Options: An Example.

While the details of exporting and importing are discussed in separate topics, the basic principle of operation is simple. When you export, VBLM writes string data in pairs to the LMX file: the original string and the translation in the current language table. When you import, VBLM reads each pair and scans all original strings in the current language table; if an exact match is found, it imports the translation.

For more information:

LTE Import & Export: Standard Edition

LTE Import & Export: Professional Edition

Using Pro's Export Options: An Example.

LTE Import & Export: Standard Edition

The import /export options provide VBLM with a rudimentary dictionary capability. For both choices, VBLM pops up a file dialog set to look for VBLM import/export files (*.lmx). LMX files are simple ASCII files with the following format:

Entry 1 original string
Entry 1 translated string
blank line
Entry 2 original
Entry 2 translation
blank line
...
Entry n original
Entry n translation
blank line

When you select **Export**, VBLM creates an LMX file with entries for each translated item in the current language table.

When you select **Import**, VBLM first asks if you want to confirm each import, then reads such a file and for every item in it 1) searches the current table for a match on the original string, and 2) imports the translation when a match is found.

See Also:

LTE Import & Export: Overview

LTE Import & Export: Professional Edition

LTE Import & Export: Options (Pro Only)

VBLM Pro's export options, set in the window that pops up when you select **Export** from the LTE file menu, control the format, content, order, and spacing of the export file (*.LMX) that VBLM will create.

The primary purpose of these options are to extend VBLM Pro's ability to work with 3rd party programs that do not support clipboard translation. By giving you control over LMX file format, the options allow you to create files that, with minimal editing on your part, can be loaded into these programs, translated, and then imported back into a language table. For details, see Using Pro's Export Options: An Example.

Format

Your choices are **Pairs** and **Sections**. If you select Pairs, VBLM exports original and translated strings in pairs, as follows

Entry 1 original string
Entry 1 translated string
blank line
Entry 2 original
Entry 2 translation
blank line
etc.

If you select Sections, VBLM exports all originals, then all translations:

[Originals]
Entry 1 original string
Entry 2 original
etc
blank line
[Translations]
Entry 1 translated string
Entry 2 translation
etc

Because it keeps all originals together in a block of text, Section format is better suited for working with 3rd party programs.

Content

Your choices are **All**, **Translated**, and **Untranslated**. If you select Translated, VBLM exports only those items which have been translated; most useful for dictionary purposes. If you select Untranslated, VBLM exports only those items which have not been translated; most useful for working with 3rd party programs. If you select All, VBLM exports all entries, translated or not.

Order

If the language table you are exporting from is sorted (see sorting a language table), this option determines whether VBLM exports in sorted or unsorted order.

Spacing

Your choices are **Single** or **Double**. If you select Double, VBLM inserts an additional blank line between each item in the file. This can be useful if the 3rd party program you are exporting to expects blank lines between paragraphs.

See Also:

LTE Import & Export: Overview

LTE Import & Export: Standard Edition

Using Pro's Export Options: An Example.

Using VBLM Pro's Export Options: An Example

VBLM Pro's export options extend its ability to work with 3rd party programs that do not support clipboard translation. By giving you control over LMX file format, the options allow you to create files that, with minimal editing on your part, can be loaded into these programs, translated, and then imported back into a language table. This topic steps through an example.

Let's say we want to create a German language table using Translate It! English/German from TimeWorks International. Translate It! cannot translate the clipboard; instead, it has an input edit box that you either load a file into or paste text into from the clipboard. When you command it to translate, it fills an output edit box with the translated contents of the input text box. You can then save this text as a file, or copy it to the clipboard. In the following example, we'll use the clipboard.

First, extract the strings from the VB project you wish to translate and instruct VBLM to create a German language table. When the LTE pops up, select **Export** from the file menu and you will be presented with the export options window:

Set **Format** to **Sections**, so that all original strings are together in a block in the LMX file.

Set **Content** to **All**, so that all original strings are included in the block.

Set **Order** to whatever you want; it doesn't matter.

Set **Spacing** to **Double**, because experience with Translate It! shows that it sometimes has trouble finding the end of distinct sentences.

When the options are set, click **OK**, give VBLM a name for the export file, and VBLM will perform the export.

Now that you've got the file:

Start up your favorite Windows text editor;

Load the export file and find the [Originals] section below the [ExportFile] header;

Consider removing hot-key ampersands from the strings, to facilitate translation. If you do, however, DO NOT SAVE THE CHANGES TO THE LMX FILE;doing so will make the strings no longer match those in the language table and VBLM will not be able to import the translations.

Select the entire block of original strings; and

Copy it to the clipboard.

Now that all strings, minus ampersands, are on the clipboard:

Start Translate It!

Select New File / English-German on the file menu to open up the pair of edit boxes;

Select the input edit box, and paste in the strings;

Select Translate All, wait for Translate It! to do its thing; and

Select the entire contents of the output edit box and copy it to the clipboard.

Now that all translated strings are on the clipboard:

Return to your text editor;

If you removed ampersands, re-load the file from disk;

Select from the first line in the [Translations] section to the end of the file (there will be as many blank

lines as there are lines in the [Originals] section);

Paste the translations;

If necessary, insert carriage returns after each translation so that each one is exactly the same number of lines from [Translations] as the original is from [Originals]. This is VERY IMPORTANT, because this is how VBLM links them. Failure to do so will result in severe problems when you try to import the file.

Finally, save the file to disk.

And now that the LMX file is complete:

Return to the German language table in VBLM;

Select Import from the file menu, and

Select the LMX file you just finished.

You're done!

About Bugs

WhippleWare NEVER ships software with known bugs. We maintain zero inventory, continually test and improve our source code, fix all problems as soon as they are reported, and ship only the latest version. This philosophy has worked so well with our **VB COMPRESS** product that our tech support line hardly ever rings, and we want to keep it that way.

VB Language Manager has been extensively beta-tested and ships with no known bugs. It is, however, Version 1.0, entering the real world for the first time. If the history of the software business is any guide, somebody out there -- maybe you -- is going to find a bug or two.

If you do find a bug, we want to know about it: please call us at **(617) 242 2511** and report it ASAP. We will fix it as fast as we possibly can and as soon as we do, we'll ship you a free update. If you report a bug within the 30-day guarantee period, your guarantee period will restart on the date you receive the fix.

Configuring the VB Control Interface

VBLM controls and interacts with VB. Because VB was not designed to be controlled (by DDE, for example), we have implemented this capability -- call it the VB Control Interface -- as a combination of WinAPI calls and SendKeys. The VB Control Interface is configured by settings in the [VBKeys] section of the INI file.

If you are using a non-English version of VB or Windows, you need to reconfigure the VB Control Interface. If you do not, VBLM will look for the wrong windows, send improper keystrokes, and generally beep and make a mess instead of controlling VB. You need to manually edit the [VBKeys] section and replace the default English values with those needed for your system.

Each entry in [VBKeys], listed below, defines a window or control caption or a keystroke sequence in standard SendKeys format. Click them for details.

AppName=Microsoft Visual Basic
LoadProjectName=%FO%N
SaveTextName=%FT%N
SaveFileAsName=%FA%N
SaveAsText_CID=4216
RestoreMin=% R
TopOfProjWin%WR{HOME}
LoadTextName=%FL%N
Replace=%R
OpenCodeWindow={F7}
CloseCodeWindow=%-C
MakeExeWindow=%FK
MakeExeName=%N
VBExit=%FX
MinimizeWin=% N
DialogYes=Y
DialogNo=N
SaveAsTextCaption=Save As Text
SaveFileWinTitle=Save File As

AppName=Microsoft Visual Basic

The "AppName" VB Control Interface key defines VB's application title, *ie* the string that appears on the task list and that you would use with the "AppActivate" procedure .

The default English sequence is "Microsoft Visual Basic"

LoadProjectName=%FO%N

The "LoadProjectName" VB Control Interface key defines the keystrokes to send to have VB load a project. After receiving this sequence, VB should be ready to receive "projectname~".

The default English sequence is "%FO%N"

SaveTextName=%FT%N

The "SaveTextName" VB Control Interface key defines the keystrokes to send to have VB save the text of a file. After receiving this sequence, VB should be ready to receive "filename~".

The default English sequence is "%FT%N"

SaveFileAsName=%FA%N

The "SaveFileAsName" VB Control Interface key defines the keystrokes to send to have VB save a file with a new name. After receiving this sequence, VB should be ready to receive "filename~".

The default English sequence is "%FA%N"

SaveAsText_CID=4216

The "SaveAsText_CID" VB Control Interface key is the dialog ID of the Save As Text checkbox on VB's "Save File As" window. You shouldn't have to screw around with this.

The default English value is "4216"

RestoreMin=% R

The "RestoreMin" VB Control Interface key defines the keystrokes to send to a minimized window to restore it.

The default English sequence is "% R" (there's a space in the middle).

TopOfProjWin=%WR{HOME}

The "TopOfProjWin" VB Control Interface key defines the keystrokes to send to set the focus within VB on the project window, and to highlight the top file in the project list.

The default English sequence is "%WR{HOME}"

LoadTextName=%FL%N

The "LoadTextName" VB Control Interface key defines the keystrokes to send to have VB load text into a file. After receiving this sequence, VB should be ready to receive "filename~".

The default English sequence is "%FL%N"

Replace=%R

The "Replace" VB Control Interface key defines the keystrokes to send to have VB replace the code in a file when loading text (this sequence is sent to the "load text" window).

The default English sequence is "%R"

OpenCodeWindow={F7}

The "OpenCodeWindow" VB Control Interface key defines the keystrokes to send to have VB open the code window for the currently selected project file.

The default English sequence is "{F7}"

CloseCodeWindow=%-C

The "CloseCodeWindow" VB Control Interface key defines the keystrokes to send to have VB close the current code window.

The default English sequence is "%-C"

MakeExeWindow=%FK

The "MakeExeWindow" VB Control Interface key defines the keystrokes to send to have VB pop up the Make EXE window.

The default English sequence is "%FK"

MakeExeName=%N

The "MakeExeName" VB Control Interface key defines the keystrokes to send to have VB set the focus to the Name box in the already open Make EXE window.

The default English sequence is "%N"

VBExit=%FX

The "VBExit" VB Control Interface key defines the keystrokes to send to close VB .

The default English sequence is "%FX"

MinimizeWin=% N

The "MinimizeWin" VB Control Interface key defines the keystrokes to send to minimize a window .

The default English sequence is "% N" (there's a space in the middle).

DialogYes=Y

The "DialogYes" VB Control Interface key defines the keystrokes to send a dialog box an affirmative response.

The default English sequence is "Y" (for Yes)

DialogNo=N

The "DialogNo" VB Control Interface key defines the keystrokes to send a dialog box an negative response.

The default English sequence is "N" (for No)

SaveAsTextCaption=Save as Text

The "SaveAsTextCaption" VB Control Interface key defines the caption of the SaveAsText checkbox that appears on VB's Save File As window .

The default English sequence is "Save as Text"

SaveFileWinTitle=Save File As

The "SaveFileWinTitle" VB Control Interface key defines the title of the window that pops up when you select "Save As" on VB's file menu.

The default English sequence is "Save File As"

About WhippleWare

In 1991, MIT student Ben Whipple used VB to write software for his doctoral research. Ben was delighted with VB but frustrated by the time it took to make apps run efficiently. He created VB Compress, the first and only VB optimization utility. VB Compress has so far received both Readers' and Editors' Choice Awards from the Visual Basic Programmer's Journal, has been recommended by speakers at VBITS and Tech-Ed conferences, and is used daily by thousands of very satisfied customers. With VB Language Manager, Ben now re-applies both his VB code-processing expertise and the principles of doing business that made VB Compress a hit:

- Our customers are not only always right, they give us our best ideas. So when you call WhippleWare, you'll probably talk to the guy who writes the code. And he'll listen to you.
- We maintain no inventory, continually test and improve our products, and fix all problems when reported. We don't ship software with known bugs. Our tech support line hardly ever rings, and we want to keep it that way.
- We trust our customers and unconditionally guarantee their satisfaction. None of this "unless you open the envelope" or "if it doesn't do what we say it does" nonsense. If you aren't 100% satisfied for any reason at all, simply uninstall and return the product within 30 days and we'll refund everything except shipping and handling. The only question we'll ask: Why weren't you satisfied?

Revision History

October 6, 1994: Version 1.01

1) Configuration file processing internationalized; uses sDecimal and sList characters from WIN.INI instead of hard-coded period and comma. Fixes configuration problems encountered by users with sDecimal set (symptom was "Invalid Property" in fGenOpt and frmPrintOpt load events).

2) Map graphics and fonts now aware of TwipsPerPixel. Fixes problems encountered by users with TwipsPerPixel <> 15 (1024x768 and higher resolution).

August 10, 1994: Version 1.00

First Release: No Revisions