Introduction

First things first. Thank you for evaluating the Message Blaster. We are sure you will like it. If you need any help our Tech Support is easily reachable.

When most people look at an application they think of that application as a single window. This is rarely the case. Most applications are made up of a variety of windows in the form of main windows and any of a variety of controls or child windows such as buttons, edit boxes, etc.

At the heart of Windows 95 and Windows NT is a messaging system that is the basis for most of what goes on in their respective systems. Messages are the "atomic" events that programmers can respond to. For example, whenever you press a key there can be up to 3 messages generated: WM_KEYDOWN, WM_KEYUP and WM_CHAR. Other releated messages are generated for systems keys (the alt key).

These messages are passed to the appropriate window for handling. There are several ways to generate messages. Windows can create them in response to hardware events such as   keystrokes or mouse clicks or software events such as when a screen needs repainting. Programmers can create them to send information or notifications to themselves or others.

In general, the way in which programmers respond to these message events determines how a program runs. If you want to change the behavior of a particular window, you need to respond to the appropriate message and act accordingly.

The behavior of windows are controlled by a thing called a Window Procedure or WindProc. Essentially, a WindProc is a function that acts as a filter. As messages are generated and passed to a window for possible handling, the WindProc is where a programmer looks at this stream of messages and picks the ones he is interested in processing. If the message is of no interest it is passed back to Windows for default processing.

In many programming environments access to this message stream is limited or impossible. Using the ***Message Blaster*** , you can respond to just about any message in any window.

Trial Version Information

This version of the Message Blaster is designed to work for 30 days once installed. After that time it will no longer function. If you create an executable using the trial version, the executable will fail after 30 days. If you plan to use the Message Blaster in your project please contact WareWithAll, Inc. to license a registered copy. The cost is $80 per license.

To purchase licenses for the Message Blaster call 1-800-689-0747 or send a check or money order made out to WareWithAll, Inc. for $80 U.S. and mail it to:

WareWithAll, Inc.
758 N. Williams Drive
Palatine, IL 60067

How to

Using the Message Blaster is easy. Just drop a Message Blaster on a form and bring up the property sheet. From there choose the custom properties and the Property Page will be displayed.

From there you can choose the messages you are interesting in trapping. As you choose messages, you can select whether you want the message passed to the original window procedure after, before or not at all. Once you have finished selecting messages, all that remains to be done is to write one line of code that tells the Message Blaster what window you want to trap message for. This is done by setting the hWndTarget property.

That's it! Now all you have to do is write code in response to the Message event from the Message Blaster and your done.

You can also set up and use the Message Blaster completly in code. See the smallcap example for details.

If you have an object browser (like the one that comes with VB), you can browse the many constants an functions that are automatically declared for you. Almost every conceivable Windows message is listed along with a little help on what they do. Along with all the messages are constants used by Message Blaster as well as constants commonly used in wParam and lParams.

There are several examples that come with the product that demonstrate the capabilities of the product.

Technical Support

You can reach WareWithAll any number of ways. For technical support email us on the internet at edstaff@mcs.com or opearce@vdospk.com or on Compuserve at 72240,2171or 76106,1541.

You can also call us at 708-358-0484 or 901-763-4868.

Of the properties, all but two of them are standard properties. Unfortunately, standard properties are not really standard. There is no guarantee that your particular control container implements any or all of the standard properties.

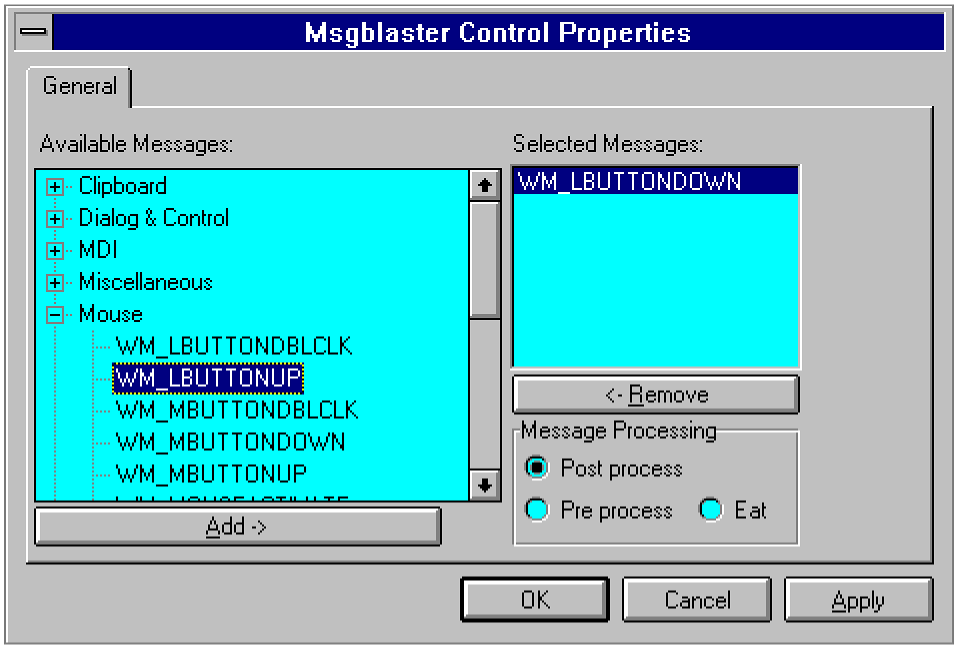
Properties

About
Property Page
Enabled*
hWndTarget
Index*
Left*
Name*
Tag*
Top*

*Standard Property

Property Page

The property page is a convenient way of selecting messages to trap at design time without having to write any code. Simply select the messages you are interested in from the tree on the left, set the processing by choosing the appropriate radio button (Post process, Pre process or Eat) and hit ok. Then all you have to do is set the hWndTarget at run-time.


Msgblaster Control Properties

General

Available Messages:

- Clipboard
- Dialog & Control
- MDI
- Miscellaneous
- Mouse
  - WM_LBUTTONDBLCLK
  - WM_LBUTTONUP
  - WM_MBUTTONDBLCLK
  - WM_MBUTTONDOWN
  - WM_MBUTTONUP

Selected Messages:

WM_LBUTTONDOWN

Message Processing: [✓ Post process] Pre process Eat

See also

AddMessage

hWndTarget

Description

Sets or gets the window handle of the target window.
The target window is the window that the Message Blaster will trap messages on.

Usage

[*form* .]*ctlname.* **hWndTarget =** setting

Data Type

Long

Remarks

Set this property to the window handle (hWnd) of the window that you want to Message Blaster to trap messages on. One of the interesting things you can do with this property is to change it at run-time with the focus. For example, if you wanted to trap a particular message for all the controls on a form, all you would have to do is reset the hWndTarget to the hWnd of whatever control/window receives the focus. Although this property is available at design time, it typically is set at run-time.

Example

```
Private Sub Form_Load()
	MsgBlaster1.hWndTarget = Form1.hWnd
End Sub
```

Enabled

Description
Turn message trapping on or off.

Syntax
object.Enabled [= boolean]

Remarks
By default, the enabled property is set to True.

Name

Description

Returns the name used in code to identify an object.

Syntax

object.Name

Remarks

The default name for new objects is the kind of object plus a unique integer.   For example, the first new Message Blaster control is MsgBlaster. An object's Name property must start with a letter and can be a maximum of 40 characters.   It can include numbers and underlined (_) characters but can't include punctuation or spaces.

The most important property!

Index

Description

Returns or sets the number that uniquely identifies a control in a control array. Available only if the control is part of a control array.

Syntax

object[(number)].Index

The Index property syntax has these parts:

Remarks

See the Visual Basic reference for more information

Tag

Description

Returns or sets any extra data needed for your program.   Unlike other properties, the value of the Tag property isn't used by Visual Basic; you can use this property to identify objects.

Syntax

object.Tag [= expression]

Remarks

See the Visual Basic reference for more information.

Left

Description

Returns or sets the distance between the internal left edge of an object and the left edge of its container.

Syntax

object.Left [= number]

Remarks

See the Visual Basic reference for more information.

Top

Description

Returns or sets the distance between the internal top edge of an object and the top edge of its container.

Syntax

object.top [= number]

Remarks

See the Visual Basic reference for more information.

Methods
AddMessage
ClearMessageList

AddMessage

Syntax

object.AddMessage Message, Processing

The AddMessage method syntax has these parts:

Part	Data Type	Description
Message	Long	Value of message to trap
Processing	Integer	How to process message

Remarks

Adds a message to the list of messages that the Message Blaster will look for. You can add as many messages as you want, including user defined messages.

Example

```
Private Sub Form_Load()
        MsgBlaster1.AddMessage _
                WM_LBUTTONDOWN, POSTPROCESS
        MsgBlaster1.AddMessage _
                WM_USER + 1, EATMESSAGE
End Sub
```

See Also

Property Page, hWndTarget

ClearMessageList

Syntax

object.ClearMessageList

Remarks

Use the ClearMessageList to reset the list of messages to be trapped by Message Blaster. Normally, when the hWndTarget is changed at run-time the message list is not reset.

Example

```
Private Sub Command1_GotFocus()
	' This example resets the hWndTarget
	' and message list when this object gets the focus
	MsgBlaster1.hWndTarget = Command1.hWnd
	MsgBlaster1.ClearMessageList
	MsgBlaster1.AddMessage _
		BN_CLICK,   POSTPROCESS
End Sub
```

There is only one event associated with the Message Blaster.

Events
Message

Message

Occurs whenever the Message Blaster see a message destined for the hWndTarget that is in the message list.

Syntax

```
Private Sub MsgBlaster1_Message (_
        ByVal hWnd as Long, _
        ByVal Msg As Long, _
        wParam as Long, _
        lParam As Long, _
        nPassage As Integer, _
        lReturnValue As Long)
```

The Message Event has these parameters:

Parameter	**Data Type**	**Description**
hWnd	Long	Handle of window receiving a message
Msg	Long	Value of message received
wParam	Long	wParam of Message
lParam	Long	lParam of Message
nPassage	Integer	How Message Blaster will process the message
lReturnValue	Long	Value Message Blaster should return to Windows

Remarks

The Message event is where to respond to any messages that Message Blaster traps for you.

Example

```
Private Sub MsgBlaster1_Message(ByVal hWnd As Long, _
        ByVal Msg As Long,_
        wParam As Long, _
        Param As Long,_
        nPassage As Integer,_
        lReturnValue As Long)

        ' This example shows how you might handle
        ' adding text to a status bar based on a menu
        ' selection
        Select Case Msg
                Case WM_MENUSELECT
                        Select Case wParam
                                Case 2
                                        ' Set status bar
                Case 3
                    ' Set status bar
                Case Else
                    ' Set status bar
            End Select
    End Select

End Sub
```

Constant	Value	Description
POSTPROCESS	1	Message is passed to target window after Message Blaster message event
EATMESSAGE	0	Message event will be fired, but target window will not receive message
PREPROCESS	-1	Rare. Message is sent to target window first, then message event is fired