Help for MsgHook

Properties Events Methods

Getting Custom Controls Written

Description

A control which provides the ability to intercept messages sent to a window.

File Name

MSGHOOK.VBX, MSGHOO16,OCX, MSGHOO32.OCX

Object Type

VBHT.MsgHook

Compatibility

VB 2.0 and up

Remarks

MsgHook provides a way for the Visual Basic programmer to intercept and respond to messages which Visual Basic doesnt provide events for. For instance, you can use MsgHook to intercept taskbar related messages, thus enabling you to write a program which displays itself on the taskbar.

Distribution Note When you develop and distribute an application that uses MsgHook, you should install the file MSGHOOK.VBX, MSGHOO16.OCX, or MSGHOO32.OCX into the users Windows SYSTEM directory. MsgHook has version information built into it. So, during installation, you should ensure that you are not overwriting a newer version of MsgHook.

Close

MsgHook Properties

Properties that have special meaning for this control or that only apply to this control are marked with an asterisk (*).

***HwndHook** Property

Index Property

Left Property

***Message** Property

Name Property

Parent Property

Tag Property

Top Property

Close

Tips Events

Events that have special meaning for this control or that only apply to this control are marked with an asterisk (*).

***Message** Event

Close

MsgHook Methods

Methods that have special meaning for this control or that only apply to this control are marked with an asterisk (*).

***InvokeWindowProc** Method

HwndHook Property

Description

Handle of hooked window.

Usage

[*form.*][*control.*]**HwndHook**[= *long*]

Remarks

Assigning the value of a windows handle to the MsgHooks HwndHook property enables message interception for that window. In addition to assigning a window handle to HwndHook you must also enable the interception of one or more window messages by setting elements of the Message property.

You can set this property to zero to unhook a window.

Data Type

Integer (long)

Message Event

Description

Occurs when one of the messages enabled by the Message property is received..

Syntax

Sub *ctlname*_**Message(***msg* **As Long,** *wp* **As Long,** *lp* **As Long,** *result* **As Long)**

Remarks

This event Occurs when one of the messages enabled by the Message property is received..

The *msg* parameter lets you know which message has been received, and the *wp* and *lp* parameters the values passed as *wp* and *lp* to window procedures. You will need to refer to Windows API documentation to understand the values of *wp* and *lp* which are passed with the various messages.

Window procedures must return a value of some sort, thats what the *result* argument is for. Inside the Message event you must set *result* to a value appropriate for the message received. Again you will need to refer to Windows API documentation for specifics.

Message Property

Description

Used to enable firing of the Message.event when messages are received.

Usage

[*form.*][*control.*]**Message(** *index As Long* **)** [= *Boolean*]

Remarks

This property is used to enable/disable firing of the Message.event for messages received by the hooked window. Setting an element of the Message array to TRUE enables message events for the corresponding message number, setting the element to FALSE disables the event.

Data Type

Boolean

InvokeWindowProc Method

Description

Calls the default window procedure for the hooked window

Usage

Long *result* = [*form.*][*control.*]**InvokeWindowProc(***msg* **As Long,** *wp* **As Long,** *lp* **As Long)**

[*form.*][*control.*]**Version**

Remarks

When a window is hooked you have the opportunity to handle windows messages before they are seen by Visual Basic and the hooked window. If you are not handling normal processing of received messages entirely by yourself you will want to call the hooked windows message procedure at some point during the Message event routine. To do that you use the InvokeWindowProc method.

Under some circumstances you may choose to handle the message entirely within your own code and not allow the hooked window to process it. In that case you will not call InvokeWindowProc. In either case you must ensure that you return a meaningful value in result or Windows may do strange things.

Data Type

Long

Getting Custom Controls Written

If you or your organization would like to have custom controls written, you can contact me at the following:

Zane Thomas
Post Office Box 300
Indianola, WA 98342

CompuServe: 72060,3327

Internet: zthomas@acvitexpert.com

or contact James Shields at:

James Shields
Mabry Software, Inc.
Post Office Box 31926
Seattle, WA 98103-1926

Phone: 206-634-1443
Fax: 206-632-0272
BBS: WinDev BBS 206-634-0783

CompuServe: 71231,2066
Internet: mabry@halcyon.com