µ §

Desktop Intercept 1.0

Paul Butcher

Introduction

Recently a number of Microsoft Windows programs have appeared that, in one way or another, enhance the behaviour of Windows' "desktop window". All such techiques rely on intercepting messages intended for the desktop window before the desktop itself receives them. A number of mechanisms may be used to achieve this, all of which have advantages and disadvantages. Most work adequately as long as only one desktop enhancer is active at any one time. When more than one is active simultaneously, however, conflicts can emerge. Desktop Intercept is an attempt to coordinate these various applications in such a way that they can work together without getting in each other's way. In addition, Desktop Intercept provides a (relatively) painless way for new programs of this type to be constructed.

Desktop Intercept consists of two parts, a Dynamic-Link Library (DLL) that applications wishing to intercept desktop messages may register with and a control panel applet through which the user may configure their system. Applications don't interfere with each other's execution because all interaction with the desktop window is coordinated through the Desktop Intercept DLL. Any conflicts between applications may be resolved by the user via the control panel.

Desktop Intercept is *not* in the public domain, the author retains full copyright, but no charge is made for its use. Unmodified versions of the "dskint.dll" and "di_cpl.cpl" files may be distributed with applications at no charge (although I would appreciate being told about any applications that make use of Desktop Intecept). The source code is available on request. The author can be contacted at the address below (please use e-mail if at all possible - I'm far more likely to get back to you in a timely manner if you do).

Paul Butcher
c/o Harlequin Ltd.
Barrington Hall
Barrington
Cambridgeshire
CB2 5RG

e-mail: paulb@harlequin.co.uk

Tel: (0223) 872522 or +44 223 872522 (international)
Fax: (0223) 872519 or +44 223 872519

Important Note: Please be aware that, apart from being my employer and therefore

keeping a roof over my head, Harlequin has nothing whatsoever to do with Desktop Intercept. Please do not telephone Harlequin technical support demanding help with Desktop Intercept — they won't understand what the hell you're talking about!

Installing Desktop Intercept

Desktop Intercept should be installed by copying the dskint.dll and di_cpl.cpl files into the Windows "system" directory (e.g. C:\WINDOWS\SYSTEM). When installing you should check for an existing version of the DLL, and only overwrite the existing file if it is an earlier version than the one being installed.

Writing a Desktop Intercept client

All Desktop Interface clients should call DiInitialise before calling any other functions.

A desktop filter is installed by passing a callback function to DiRegisterFilter, and removed by calling DiUnregisterFilter. The desktop may be marked as accepting files dropped from the Windows File Manager by calling DiUnregisterFilter.

After a filter has been installed, Desktop Intercept will call that filter with each message received by the desktop window. If the filter returns FALSE, then that message will be passed to the next filter in the chain. If the filter returns TRUE, then the message is consumed and no other filters are called. If no filter returns TRUE, then the message is passed to the desktop window as usual.

The filter function should be contained within a DLL *which must be compiled with real-mode prologue and epilogue code*. Similar restrictions apply to any functions called by the filter. With Microsoft compilers, real-mode prolog and epilog code is generated by giving the /Gw switch on the command-line instead of /GD.

Filter functions should be kept as short and efficient as they can introduce significant overhead. The recommended architecture is to have the filter function simply determine whether a message is of interest to the application that registered it. If so, then the filter function calls PostMessage to forward the message to a window in that application. This has two benefits; the rest of the application need not reside in a DLL or be compiled with /Gw and the filter function is kept simple.

Desktop Intercept clients should bear in mind that they may not be the only applications intercepting messages from the desktop and should attempt to cooperate with other clients. All messages that don't have to be processed by the client should be forwarded. The user should be given options to allow conflicts to be resolved (e.g. if an application displays a menu when the mouse is clicked on the desktop, it should allow the user to select which button and whether the shift, control or alt keys should be used).

Functions

DWORD WINAPI DiGetVer(VOID)

Return Value

This function returns the major and minor version numbers of Desktop Intercept.

Comments

The low order word contains the minor version number, the high order word the major version number.

VOID DiInitialise(VOID)

Comments

This function must be called by all Desktop Interface clients, and should be called before all other functions (with the exception of DiGetVer).

HANDLE WINAPI DiRegisterFilter(LPDIFILTER lpfnFilter, LPSTR lpszAppName)

Parameters

lpfnFilter Pointer to a function to which all desktop window messages are sent.

lpszAppName Pointer to a string which gives the name of the application. This is the name that will appear in the “Clients” list of the Desktop Intercept control panel.

Return Value

If successful, returns a handle which may be passed as an argument to DiUnRegisterFilter, otherwise NULL.

Comments

This function registers the application with the Desktop Intercept DLL. When the application no longer wishes to be notified of messages from the desktop, it should call DiUnRegisterFilter.

VOID WINAPI DiUnregisterFilter(HANDLE h)

Parameters

h A handle previously returned from DiRegisterFilter.

Comments

After calling this function the application will no longer be notified of messages sent to the desktop window.

BOOL (WINAPI *LPDIFILTER)(UINT uMsg, WPARAM wParam, LPARAM lParam,
LRESULT *pLResult)

Parameters

uMsg	As for WindowProc.
wParam	As for WindowProc.
lParam	As for WindowProc.
pLResult	A pointer to a the value to be returned from this message.

Return value

FALSE if this message should be passed to the next application in the Desktop Intercept chain, TRUE otherwise.

Comments

If this function returns TRUE, the value placed in pLResult is returned from the message, and no other functions in the Desktop Intercept chain are called. If the function returns FALSE, the next function in the chain is called. If there is no other function in the chain, the default desktop window procedure is called.

VOID WINAPI DiAcceptFiles(HANDLE h, BOOL fAccept)

Parameters

h	A handle previously returned from DiRegister
fAccept	TRUE to allow the desktop to be a drop-site, FALSE otherwise.

Comments

If one or more programs have called DiAcceptFiles with fAccept set to TRUE, the desktop will allow files to be dropped, otherwise the “no entry” cursor will be displayed over the desktop window when files are being dragged.

Interception Methods

Desktop Intercept uses two message interception methods, “subclass” and “hook”. In the future it’s intended for a further method, “overlay” to be supported as well.

Subclass: The subclass method involves subclassing the desktop window (i.e. replacing the standard desktop window procedure with an alternative).

Hook: The hook method involves installing a system-wide windows hook of type

WH_GETMESSAGE. Unfortunately a system-wide hook has to be used as (for reasons I don't understand) installing a task-specific hook for the desktop window task doesn't seem to work. This means that this method involves slightly more overhead than the subclass method as all messages to all windows are filtered. This method is probably preferable, however, as it is somewhat "cleaner" than subclassing the desktop.

Overlay: The overlay method involves creating a large transparent window immediately over the desktop window. Messages are not "intercepted" as such,. As this transparent window completely covers the desktop window, it receives all messages caused by mouse clicks, etc. over the desktop.

Future plans

- Implement the "overlay" technique.
- Windows/NT port (and possibly Win32s, although this is unlikely).

Change History

Version 0.1: Preliminary release to alpha testers.

Version 1.0 (BETA): Released on 27th Sept. 1993.

- HINSTANCE parameter removed from DiRegisterFilter.
- Support for .ini file.
- Control Panel Applet.

Version 1.0: Released on 1st Oct. 1993.

- Debugging removed.
- Documentation corrections.