Quick Tour of Instant File Access

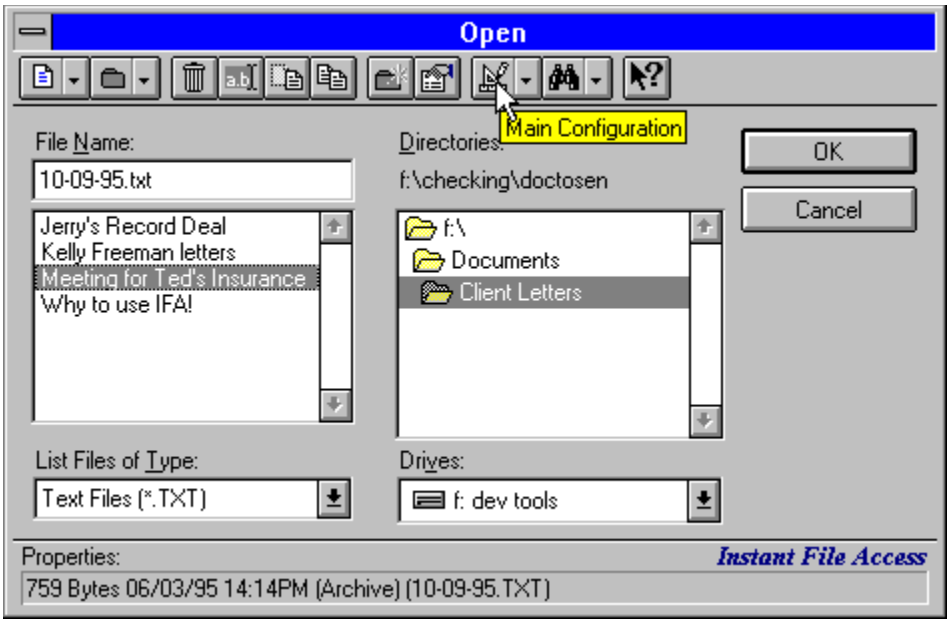
To view this tour, simply click the "**>>**" and "**<<**" buttons to move through the pages of the tour. Or you can click on the topics below. When finished, simply close the tour application.

What does IFA do?

Instant File Access (or **IFA** for short) enhances the File Dialogs of Windows applications by adding features such as ...

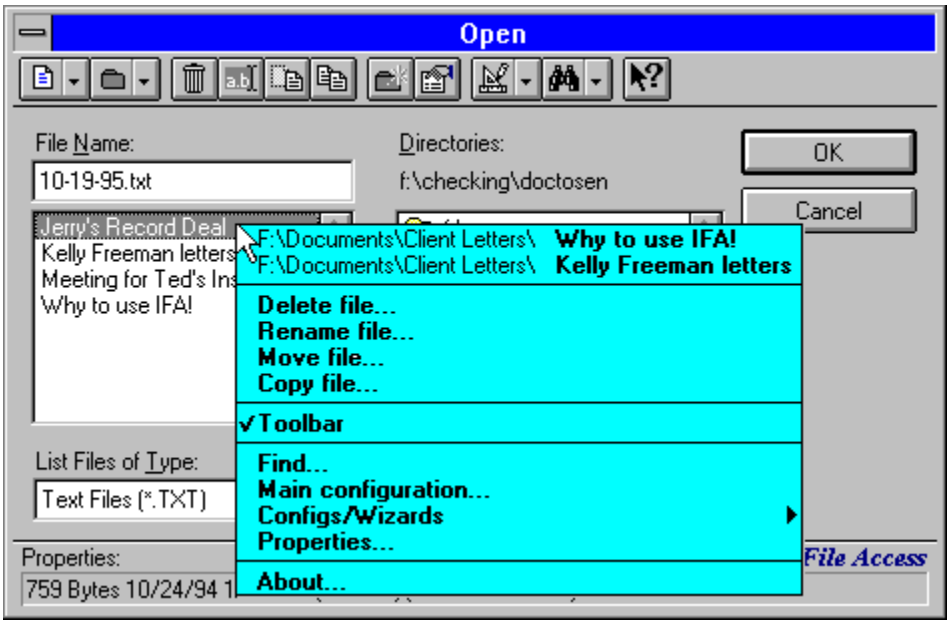
- Quick Access to previously opened files
- Long Filename support (supports 4DOS description format)
- File/Text search (and replace)
- File/directory functions such as copy/move/rename, etc.

The picture below is a file dialog with IFA attached. You can click on various parts of the picture to display help on the item.



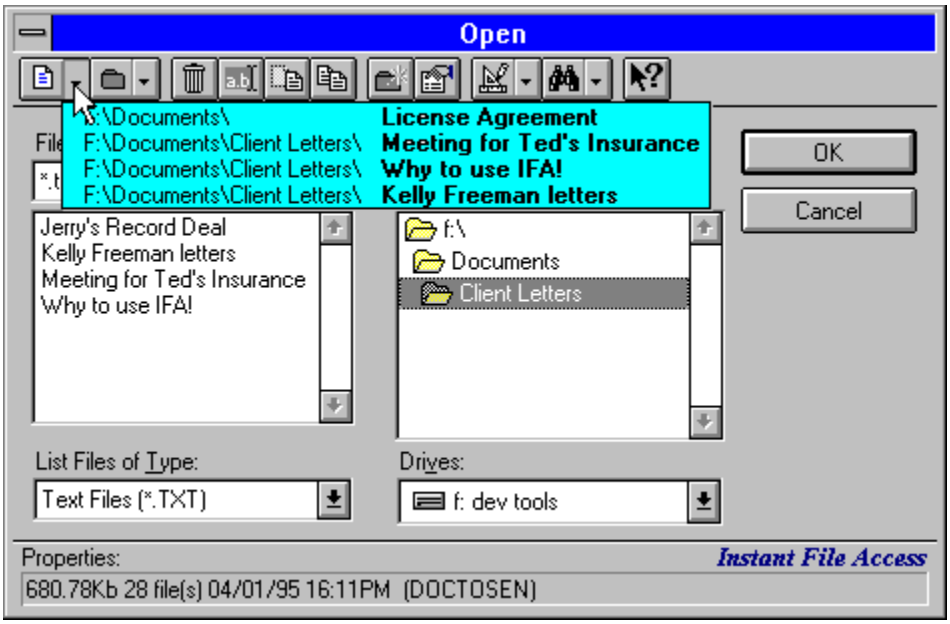
Popup Menu

Click the secondary mouse button (the Right Button for most) over the File Dialog to activate a popup menu of functions.



File History

The left few buttons of the toolbar are used to display the files and directories previously opened by the application owning the file dialog. As you open more files, this list grows until the limit is reached. This limit is set in the Main Configuration dialog.



Long File Names

IFA also lets you assign **long meaningful names** to your files. Instead of naming the minutes of your business meeting held on 09/95...

BUSM0995.DOC

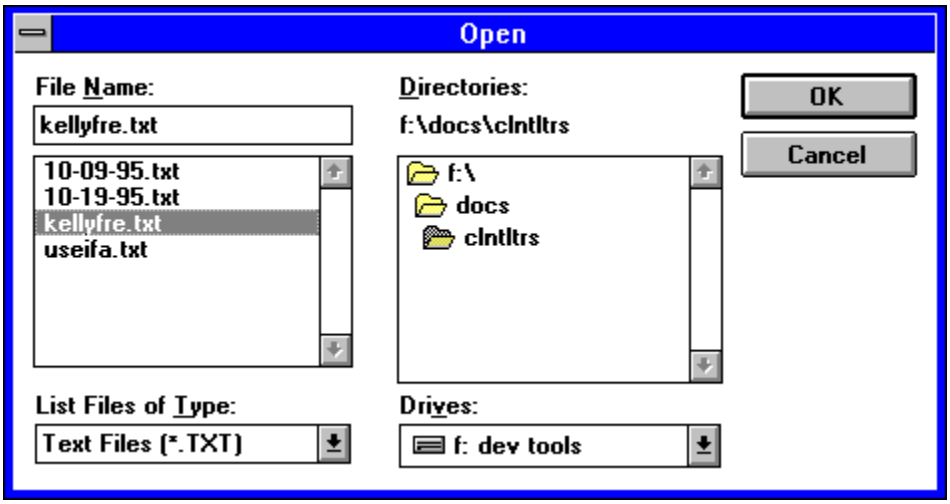
You can name it...

Business Meeting of September, 1995

To save a file with a long name, you can simply type the long name in the **Filename** field. Or you can assign a regular DOS name to the file, then rename it.

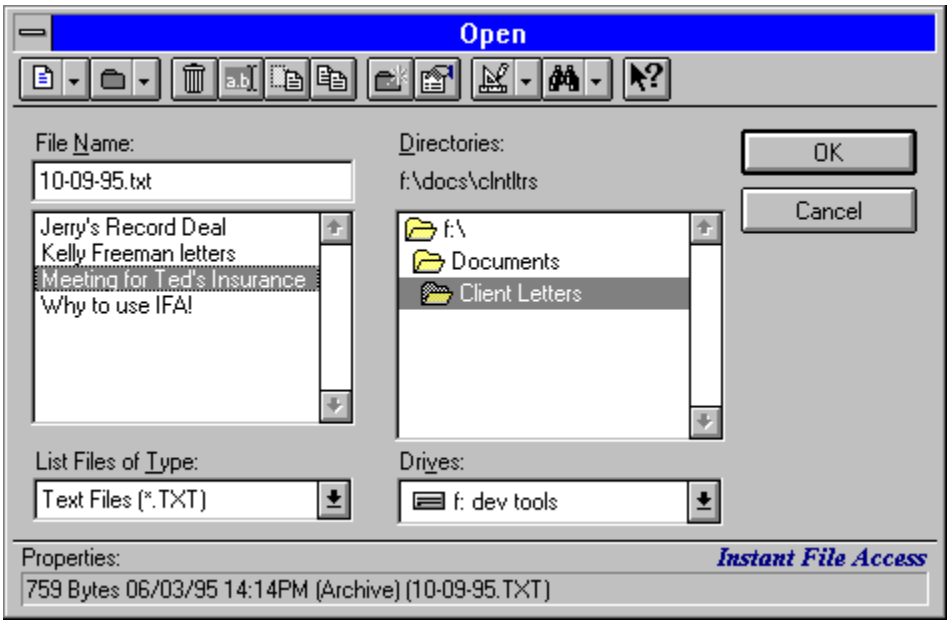
Without IFA

Without IFA, your file list looks something like this...



With IFA

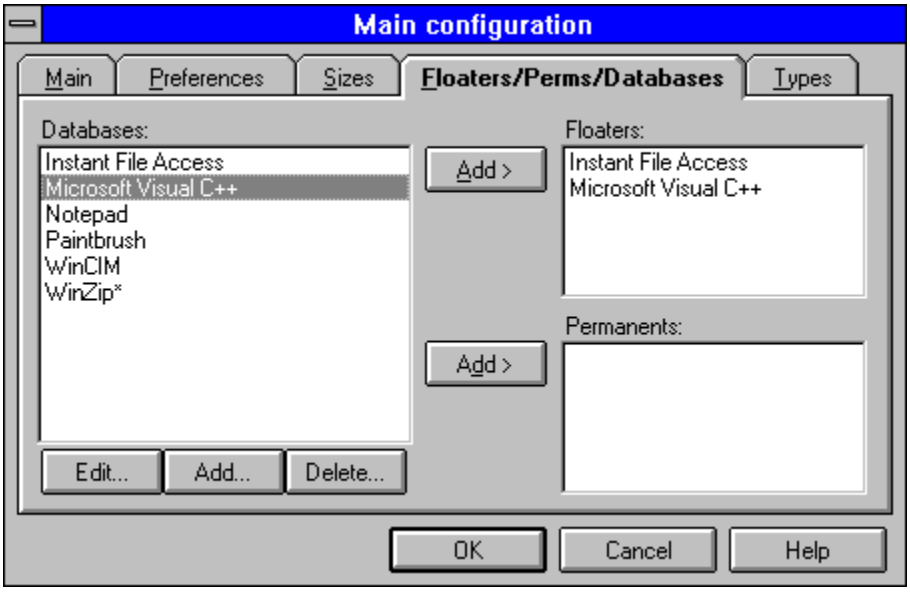
With IFA and long file names, your file list can look like this...



Floaters and Permanents

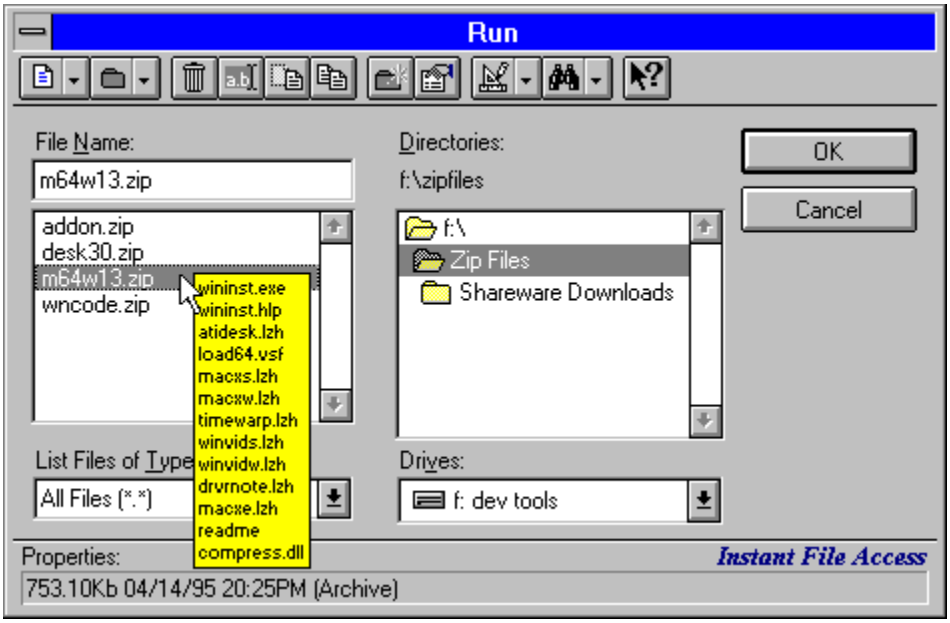
Floaters are small floating windows appearing beside the file dialog. They contain lists of your favorite or most frequently used files and directories. You can create Floaters to hold your Word templates, or Excel Time Sheet form, or whatever file you repeatedly need.

Permanents are like Floaters, only they appear at the top of the file and directory popups.
To create Floaters and Permanents, click on the Configuration Button in the toolbar and select the **Floaters/Perms/Databases** tab. You can then create new File Databases to hold your file lists. Then add that database to your Floaters and/or Permanents lists.



File Viewers

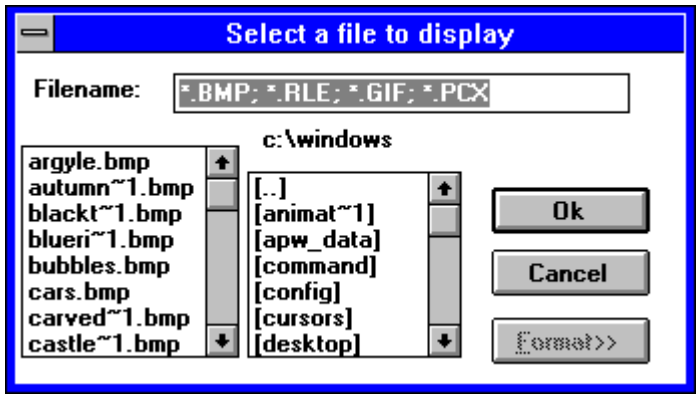
Instant File Access lets you see the file you are about to open, before you open it. Lets say you are looking for a file contained within one of the 4 ZIP files in a directory. Simply hold down the ALT key while clicking the primary mouse button on the file name, and a small window will appear displaying the contents of the ZIP file. This can be done for text files as well. The functionality of the file viewer can be easily modified and extended by editing the **VIEW_TXT.DES** file in the IFA directory.



See also...
Adding More Viewers

Not a Common Dialog

Some applications don't support the Common Dialog facility of Windows 3.1, so they design their own dialog to load and save files. The dialog below is an example of one.



A feature known as All To Common can take care of this by replacing this dialog with a Windows 3.1 Common File Dialog, which is compatible with IFA. The replaced dialog is shown on the next page...

All To Common (tm)

To replace a non-standard file dialog, simply call up the **All To Common Wizard** from IFA's main desktop icon. This wizard takes you through a few simple steps which allow you to replace the non-standard file dialog with a Common File Dialog. You will then have access to IFA's features within that program, and never have to see the original file dialog again.

As You Can See

Instant File Access *saves you time*, and that's important. It also extends your creative abilities beyond the document text, and into the file name as well.

Alexoft

507 de la Metairie
Nuns' Island, Quebec
CANADA H3E 1S4

Numbers

Compuserve:	**72154,15**
Internet:	**72154.15@compuserve.com**
MSN:	**Alexoft**
Phone:	**(514) 762-1792**

Open Last File

This button is used to open the last file opened by the application that owns this file dialog. To open other previously opened files, click the small down-arrow button to the right. To see what file will be opened, simply hold the mouse pointer over the button until the tool tip appears. The file name will be displayed there.

Open Previous File

This button displays a drop-down list of previously opened files by this application.

Go To Last Directory

This button is used to go to the directory of the last file opened by this application. To go to the directory of other previously opened files, click the small down-arrow button to the right. To see which directory you will go to, simply hold the mouse pointer over the button until the tool tip appears. The directory name will be displayed there.

Go To Previous Directory

This button displays a drop-down list of directories of previously opened files by this application.

Delete File/Directory

This button is used to delete the currently selected file or directory. To delete a file or directory, select the file/directory within either list box, click this button, then answer Yes when asked if you're sure.

Rename File/Directory

This button is used to rename the currently selected file or directory. It is also used to change the Long Name of the file. You can add a long name to a regular file by clicking this button and entering the long name in the Description field.

Move File

This button is used to Move the currently selected file. This operation will first copy the file to the destination, then delete the source.

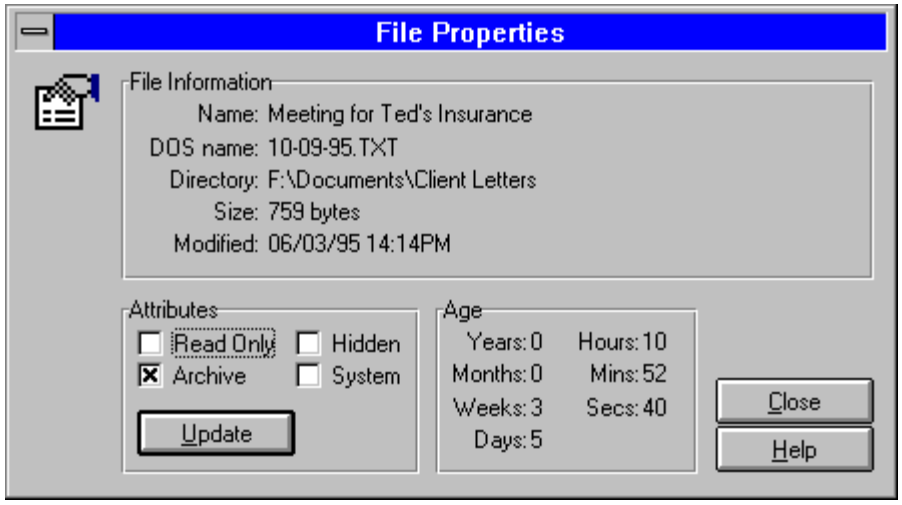
Copy File

This button is used to copy the currently selected file.

Create Directory

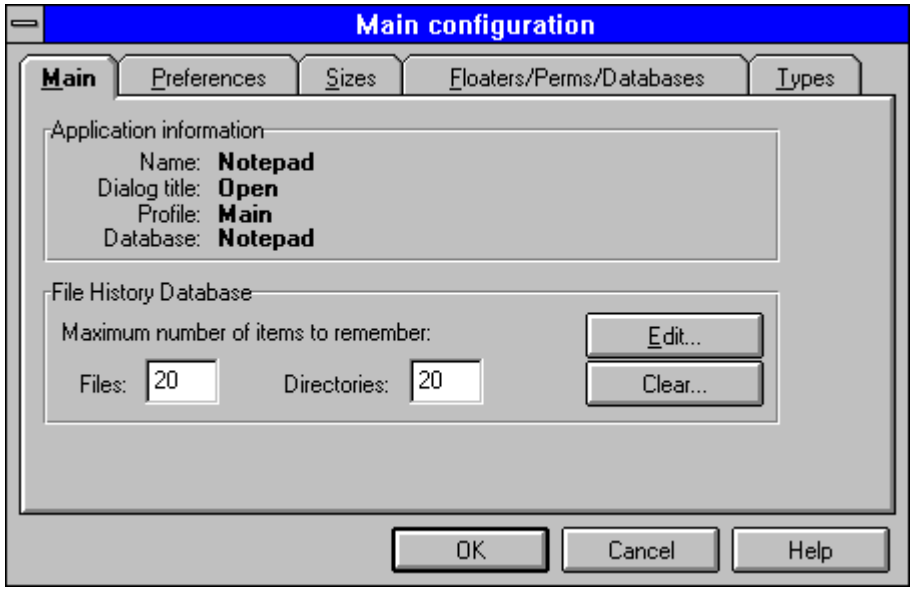
This button is used to create a new directory. You can enter an entire path if you wish, such as...
C:\Documents\Mike's Files\Lists

File Properties



Displays information about the currently selected file or directory.

Configuration



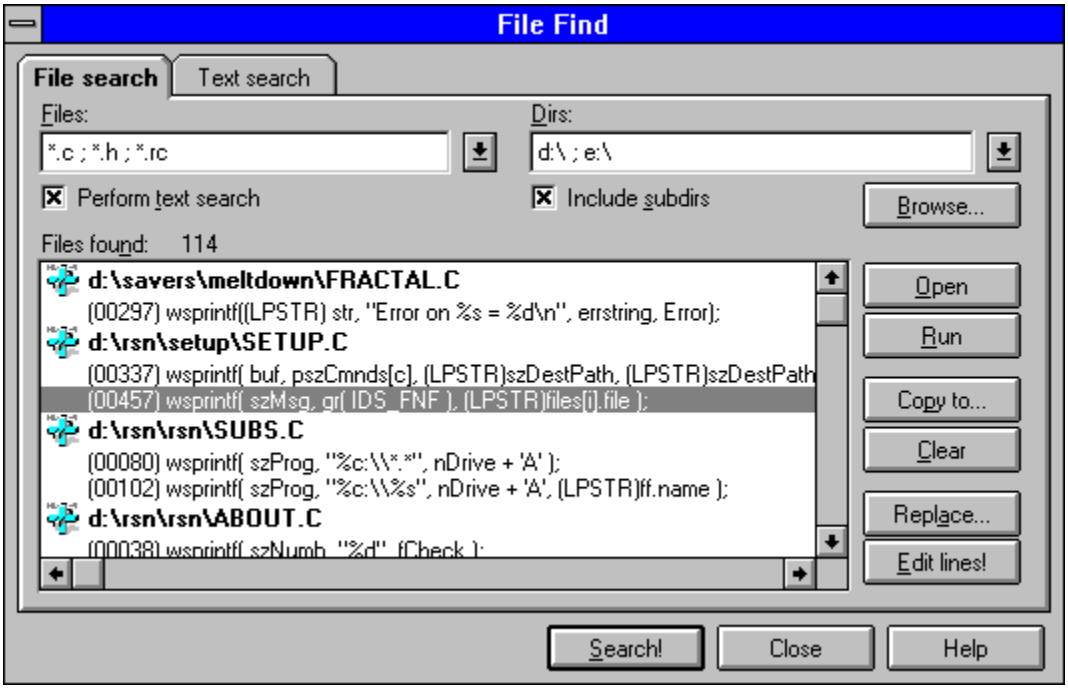
Lets you setup how IFA behaves with each application. You can setup your Floaters and Permanents, setup if the long filenames will appear in the list boxes or in the status bar, and how you want your file lists to appear.

Configurations & Wizards

This button displays a menu of other available configurations and Wizards.

Find File/Text

Find one or more files, in one or more places, and even find a particular text or phrase within those files. This is done here, and found files can be opened directly from this dialog.



Files Found

This button will display the list of files found in the previous file search. This is only if you selected the Copy... option in the File Find dialog.

Context Help

Press this button to enter the Context Help mode. The button will turn into a flashing yellow hand, which is what you must click to exit the Context Help mode. Context Help is also available from within any dialog box of IFA. Once in Context Help mode, the cursor changes shape to indicate fields with help attached. Click a field and a popup help topic appears explaining the purpose of the field. Once finished, click the flashing indicator to exit Context Help mode.

Status Bar

This is where the file details are shown.

If you have selected a file, you will see the Date and Time of file creation or last modification, the file attributes, the size of the file, and the DOS file name if it has a Long Name.

If you have selected a directory, you will see the Date and Time the directory was created, the directory attributes, the total size of all files within that directory, and the DOS name of the directory if it has a Long Name.

If you have selected the root directory such as C:\, you will see the total size of that disk drive and the amount of space free.

Tool Tips

If you hold the mouse pointer over a toolbar button for more than 1/4 second, this little yellow friend will popup to tell you what that item is.

Adding More Viewers

A file viewer is simply a Dynamic Link Library (DLL) which exports a function named **Viewer_Show**. This function is called when a file of the type associated with this viewer is clicked on in the Files listbox while the ALT key is pressed. To associate a file type, the following syntax within the **VIEW_TXT.DES** file is used...

{library name.DLL}, {file type 1}, {file type 2}, {...}

The exported function must be declared as follows ...

BOOL WINAPI __export Viewer_Show(LPCSTR lpszFile);

Where:

BOOL	Return TRUE if the file was viewed. Return FALSE if there was an error.
lpszFile	Pointer to a full path name of the file to view.

The viewer should format the file for display, create a window of WM_POPUP style, then enter a message loop until the user pressed a key or a mouse button. The window is then destroyed and the activation set back to the original window.

See Also...
Sample Text Viewer

Copy

Close

```
////////////////////////////////////////////////////////////////////////
// Text File Viewer for Instant File Access
// Exported call: Viewer_Show
//
// Copyright (C) 1995, Alexoft
//
#include <windows.h>
#include <windowsx.h>

#include <stdlib.h>
#include <stdio.h>
#include <string.h>
#include <malloc.h>
#include <memory.h>
#include <ctype.h>

#define VIEW_BUF 16000     // Buffer to hold viewed text
#define MAXLINES 15        // Maximum number of lines to view

HINSTANCE hInst;
BOOL fRegistered;
char szClassName[] = "IFATextViewClass";
LOGFONT lfTiny =
{
      -9, 0, 0, 0, FW_NORMAL, 0, 0, 0,
      DEFAULT_CHARSET, OUT_CHARACTER_PRECIS,
      CLIP_DEFAULT_PRECIS,
      PROOF_QUALITY, DEFAULT_PITCH|FF_DONTCARE,
      "Small Fonts"
};

typedef struct tagINFO
{
      BOOL   fInView;
      HFONT hFont;
      LPSTR pszBuf;
      char   szFile[_MAX_PATH+1];
}
INFO;

LRESULT CALLBACK fnViewerProc( HWND hwnd, UINT msg, WPARAM wParam,
                                    LPARAM lParam );

void _cdecl _setargv() {};
void _cdecl _setenvp() {};
void _cdecl _nullcheck() {};

int CALLBACK __export LibMain( HINSTANCE hInstance, WORD wDataSeg,
                                    WORD cbHeapSize, LPSTR lpszCmdLine )
{
      WNDCLASS wc;

      hInst = hInstance;

      if( wDataSeg )
            UnlockData( 0 );

      memset( &wc, 0, sizeof( WNDCLASS ) );
```

```
	wc.style		= CS_SAVEBITS;
	wc.hCursor		= LoadCursor( NULL, IDC_ARROW );
	wc.hInstance		= hInst;
	wc.cbWndExtra		= sizeof( INFO * );
	wc.lpfnWndProc		= fnViewerProc;
	wc.lpszClassName = szClassName;

	if( !RegisterClass( &wc ) )
		return 0;

	return 1;
}

int CALLBACK __export WEP( int bSystemExit )
{
	UnregisterClass( szClassName, hInst );
	return 1;
}

BOOL WINAPI __export Viewer_Show( LPCSTR lpszFile )
{
	int iPos;
	INFO *i;
	HWND hwnd;
	POINT pt;
	FILE *fp;

	GetCursorPos( &pt );

	hwnd = CreateWindow(
			szClassName,
			NULL,
			WS_POPUP|WS_BORDER,
			pt.x, pt.y, 0, 0,
			NULL, (HMENU)0, hInst,
			NULL );

	if( hwnd == NULL )
		return FALSE;

	i = (INFO *)GetWindowLong( hwnd, 0 );
	lstrcpy( i->szFile, lpszFile );

	fp = fopen( i->szFile, "r" );
	if( fp )
	{
		int iLine;
		char *psz = malloc( 1024 );

		while( fgets( psz, 1024, fp ) )
		{
			if( psz[0] != \n )
				break;
		}

		for( iPos = iLine = 0; iLine < MAXLINES; iLine++ )
		{
			int c;
			int s;

			for( c = s = 0; psz[c]; c++ )
```

```
                {
                    if( !isspace( psz[c] ) )
                        s = c+1;
                }

                psz[s] = 0;

                if( iPos + lstrlen( psz ) + 2 > VIEW_BUF )
                    break;

                lstrcpy( i->pszBuf + iPos, psz );
                iPos += lstrlen( i->pszBuf + iPos );

                fgets( psz, 1024, fp );
                if( feof( fp ) )
                    break;

                lstrcat( i->pszBuf + iPos, "\n" );
                iPos++;
            }

            fclose( fp );
            realloc( i->pszBuf, iPos + 4 );
        }

        SetCapture( hwnd );
        ShowWindow( hwnd, SW_SHOW );

        i->fInView = TRUE;

        while( i->fInView )
        {
            MSG msg;

            if( PeekMessage( &msg, NULL, 0, 0, PM_REMOVE ) )
            {
                TranslateMessage( &msg );
                DispatchMessage( &msg );
            }
        }

        ReleaseCapture();
        DestroyWindow( hwnd );

        return TRUE;
    }

    BOOL Viewer_OnCreate( HWND hwnd, CREATESTRUCT FAR* lpCreateStruct )
    {
        INFO *i = calloc( 1, sizeof( INFO ) );
        if( i == NULL )
            return FALSE;

        i->pszBuf = calloc( 1, VIEW_BUF + 1 );
        if( i->pszBuf == NULL )
        {
            free( i );
            return FALSE;
        }

        SetWindowLong( hwnd, 0, (LONG)i );
        i->hFont = CreateFontIndirect( &lfTiny );
```

```
	return TRUE;
}

void Viewer_OnNCDestroy( HWND hwnd )
{
	INFO *i = (INFO *)GetWindowLong( hwnd, 0 );
	DeleteFont( i->hFont );
	free( i->pszBuf );
	free( i );
}

void Viewer_OnPaint( HWND hwnd )
{
	INFO *i = (INFO *)GetWindowLong( hwnd, 0 );

	HDC hdc;
	RECT rc;
	HFONT fontOld;
	PAINTSTRUCT ps;

	hdc = BeginPaint( hwnd, &ps );
	fontOld = SelectFont( hdc, i->hFont );
	SetBkColor( ps.hdc, RGB( 0xFF, 0xFF, 0x00 ) );
	SetBkMode( hdc, TRANSPARENT );
	GetClientRect( hwnd, &rc );
	ExtTextOut( hdc, 0, 0, ETO_OPAQUE, &rc, "", 0, NULL );
	OffsetRect( &rc, 1, 1 );
	DrawText( hdc, i->pszBuf, -1, &rc, DT_LEFT|DT_NOPREFIX|DT_WORDBREAK );
	SelectFont( hdc, fontOld );
	EndPaint( hwnd, &ps );
}

void Viewer_OnShowWindow( HWND hwnd, BOOL fShow, UINT status )
{
	INFO *i = (INFO *)GetWindowLong( hwnd, 0 );

	if( fShow )
	{
		HDC hdc;
		RECT rc;
		HFONT fontOld;

		SetRect( &rc, 0, 0, GetSystemMetrics( SM_CXSCREEN ) / 4, 0 );

		hdc = GetDC( NULL );
		fontOld = SelectFont( hdc, i->hFont );
		DrawText( hdc, i->pszBuf, -1, &rc,
			DT_LEFT|DT_NOPREFIX|DT_WORDBREAK|DT_CALCRECT );
		SelectFont( hdc, fontOld );
		ReleaseDC( NULL, hdc );

		AdjustWindowRect( &rc, WS_BORDER|WS_POPUP, FALSE );
		InflateRect( &rc, 2, 2 );
		SetWindowPos( hwnd, NULL, 0, 0, rc.right - rc.left,
				rc.bottom - rc.top,
				SWP_NOZORDER|SWP_NOMOVE|SWP_NOACTIVATE );
	}
}

void Viewer_OnLButtonDown( HWND hwnd, BOOL fDoubleClick, int x, int y,
				UINT keyFlags )
```

```
{
	INFO *i = (INFO *)GetWindowLong( hwnd, 0 );
	i->fInView = FALSE;
}

void Viewer_OnKeyDown( HWND hwnd, UINT vk, BOOL fDown, int cRepeat,
				UINT flags )
{
	INFO *i = (INFO *)GetWindowLong( hwnd, 0 );
	i->fInView = FALSE;
}

void Viewer_OnLButtonUp( HWND hwnd, int x, int y, UINT keyFlags )
{
	INFO *i = (INFO *)GetWindowLong( hwnd, 0 );
	i->fInView = FALSE;
}

LRESULT CALLBACK __export fnViewerProc( HWND hwnd, UINT msg,
					WPARAM wParam, LPARAM lParam )
{
	switch( msg )
	{
		HANDLE_MSG( hwnd, WM_CREATE,	Viewer_OnCreate );
		HANDLE_MSG( hwnd, WM_KEYDOWN,	Viewer_OnKeyDown );
		HANDLE_MSG( hwnd, WM_NCDESTROY,	Viewer_OnNCDestroy );
		HANDLE_MSG( hwnd, WM_SHOWWINDOW,	Viewer_OnShowWindow );
		HANDLE_MSG( hwnd, WM_RBUTTONDOWN, Viewer_OnLButtonDown );
		HANDLE_MSG( hwnd, WM_LBUTTONDOWN, Viewer_OnLButtonDown );
		HANDLE_MSG( hwnd, WM_RBUTTONUP,	Viewer_OnLButtonUp );
		HANDLE_MSG( hwnd, WM_LBUTTONUP,	Viewer_OnLButtonUp );
		HANDLE_MSG( hwnd, WM_PAINT,	Viewer_OnPaint );
	}

	return DefWindowProc( hwnd, msg, wParam, lParam );
}
```