```
//---------------------------------------------------------------------------
//
// Copyright (c) Microsoft Corporation 1991-1994
//
// File: shlobj.h
//
//  Definitions of IShellUI interface. Eventually this file should be merged
// into COMMUI.H.
//
// History:
//  12-30-92 SatoNa     Created.
//  01-06-93 SatoNa     Added this comment block.
//  01-13-93 SatoNa     Added DragFilesOver & DropFiles
//  01-27-93 SatoNa     Created by combining shellui.h and handler.h
//  01-28-93 SatoNa     OLE 2.0 beta 2
//  03-12-93 SatoNa     Removed IFileDropTarget (we use IDropTarget)
//
//---------------------------------------------------------------------------

#ifndef _SHLOBJ_H_
#define _SHLOBJ_H_

#include <ole2.h>
#include <prsht.h>
#include <commctrl.h>   // for LPTBBUTTON

#ifndef INITGUID
#include <shlguid.h>
#endif

#ifndef RC_INVOKED
#pragma pack(1)         /* Assume byte packing throughout */
#endif /* !RC_INVOKED */

#ifdef __cplusplus
extern "C" {            /* Assume C declarations for C++ */
#endif  /* __cplusplus */

typedef void const FAR*       LPCVOID;

//---------------------------------------------------------------------------
//
// Shell Extension API
//
//---------------------------------------------------------------------------


//
// Definitions of ItemID and IDList
//
// itemid
typedef struct _SHITEMID        // mkid
{
    USHORT      cb;             // Size of the moniker ID
    BYTE        abID[1];        // The item ID (variable length)
} SHITEMID, FAR* LPSHITEMID;
typedef const SHITEMID FAR * LPCSHITEMID;

// null-terminated itemid list
typedef struct _ITEMIDLIST      // idl
{
    SHITEMID    mkid;
} ITEMIDLIST, FAR* LPITEMIDLIST;
```

```
typedef const ITEMIDLIST FAR* LPCITEMIDLIST;

//
// Task allocator
//
//  All the shell extensions MUST use this allocator when they allocate
// or free memory objects that are passed across any shell interface
// boundary.
//
// REVIEW:
//  It would be really nice if we can guarantee that shell's task
// allocator and OLE's task allocator is always the same. it is,
// however, not so easy to do, because:
//
//  1. We don't want to load COMPOBJ unless a shell extension DLL
//     loads it. We need to be notified when COMPOBJ is loaded.
//  2. We need to register our task allocator to the COMPOBJ
//     if one of shell extension DLLs loads it into the shell
//     process.
//  3. We need to get the task allocator from the COMPOBJ, if
//     the shell dll is loaded by non-shell process that registers
//     the task allocator to the COMPOBJ.
//

LPVOID WINAPI SHAlloc(ULONG cb);
LPVOID WINAPI SHRealloc(LPVOID pv, ULONG cbNew);
ULONG  WINAPI SHGetSize(LPVOID pv);
void   WINAPI SHFree(LPVOID pv);

//
// Helper macro definitions
//
#define S_BOOL(f)   MAKE_SCODE(SEVERITY_SUCCESS, FACILITY_NULL, f)


//---------------------------------------------------------------------------
//
// Interface:   IContextMenu
//
// History:
//  02-24-93 SatoNa     Created.
//
// IContextMenu::QueryContextMenu
//    This member function may insert menuitems to the specified menu (hmenu)
//  at the specified location (indexMenu which is never -1). The IDs of those
//  menuitem should be in the specified range (idCmdFirst and idCmdLast).
//  It returns the maximum menuitem ID offset as the 'code' (low word).
//    The uFlags specify the context. CMF_DEFAULTONLY flag is passed if the
//  user is performing a default action (typically by double-clicking).
//  CMF_VERBSONLY is passed if the context menu should contain only verbs
//  to the object itself. The explorer passes this flag if it is constructing
//  a context menu for a link object.
//
// IContextMenu::InvokeCommand
//    Typically this member is called when the user is selected one of menuitems
//  that are inserted by previous QueryContextMenu member. In this case, the
//  LOWORD(idCmd) contains the menuitem ID offset relative to idCmdFirst.
//
// IContextMenu::GetCommandString
//    This member function is called by the explorer either to get the
//  canonical (language independent) command name (uFlags == 0) or the help
//  text ((uFlags & GCS_HELPTEXT) != 0) for the specified command.
//
```

```
//---------------------------------------------------------------------------

#undef  INTERFACE
#define INTERFACE   IContextMenu

// QueryContextMenu uFlags
#define CMF_NORMAL        0x00000000
#define CMF_DEFAULTONLY   0x00000001
#define CMF_VERBSONLY     0x00000002
#define CMF_EXPLORE       0x00000004
#define CMF_RESERVED      0xffff0000     // View specific

// GetCommandString uFlags
#define GCS_VERB          0x00000000     // canonical verb
#define GCS_HELPTEXT      0x00000001     // help text (for status bar)

DECLARE_INTERFACE_(IContextMenu, IUnknown)
{
    // *** IUnknown methods ***
    STDMETHOD(QueryInterface) (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
    STDMETHOD_(ULONG,AddRef) (THIS)  PURE;
    STDMETHOD_(ULONG,Release) (THIS) PURE;

    STDMETHOD(QueryContextMenu)(THIS_
                                HMENU hmenu,
                                UINT indexMenu,
                                UINT idCmdFirst,
                                UINT idCmdLast,
                                UINT uFlags) PURE;

    STDMETHOD(InvokeCommand)(THIS_
                             HWND   hwndParent,
                             LPCSTR pszWorkingDir,
                             LPCSTR pszCmd,
                             LPCSTR pszParam,
                             int    iShowCmd) PURE;

    STDMETHOD(GetCommandString)(THIS_
                                UINT        idCmd,
                                UINT        uFlags,
                                UINT FAR *  pwReserved,
                                LPSTR       pszName,
                                UINT        cchMax) PURE;
};

typedef IContextMenu FAR*       LPCONTEXTMENU;

// GetIconLocation() input flags

#define GIL_OPENICON     0x0001      // allows containers to specify an "open" look
                                     // return FALSE to get the standard look

// GetIconLocation() return flags

#define GIL_SIMULATEDOC  0x0001      // simulate this document icon for this
#define GIL_PERINSTANCE  0x0002      // icons from this class are per instance (each
file has its own)
#define GIL_PERCLASS     0x0004      // icons from this class per class (shared for
all files of this type)


//============================================================================
// Helper macro for C programmers
```

```
#define _IOffset(class, itf)         ((UINT)&(((class *)0)->itf))
#define IToClass(class, itf, pitf)   ((class FAR *)(((LPSTR)pitf)-_IOffset(class,
itf)))
#define IToClassN(class, itf, pitf)  IToClass(class, itf, pitf)


//=============================================================================
//
// Interface: IShellExtInit
//
//  This interface is used to initialize shell extension objects.
//
//=============================================================================

#undef  INTERFACE
#define INTERFACE   IShellExtInit

DECLARE_INTERFACE_(IShellExtInit, IUnknown)
{
    // *** IUnknown methods ***
    STDMETHOD(QueryInterface) (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
    STDMETHOD_(ULONG,AddRef) (THIS)  PURE;
    STDMETHOD_(ULONG,Release) (THIS) PURE;

    // *** IShellExtInit methods ***
    STDMETHOD(Initialize)(THIS_ LPCITEMIDLIST pidlFolder,
                          LPDATAOBJECT lpdobj, HKEY hkeyProgID) PURE;
};

typedef IShellExtInit FAR*      LPSHELLEXTINIT;


//=============================================================================
//
// Interface: IShellPropSheetExt
//
//=============================================================================

#undef  INTERFACE
#define INTERFACE   IShellPropSheetExt

DECLARE_INTERFACE_(IShellPropSheetExt, IUnknown)
{
    // *** IUnknown methods ***
    STDMETHOD(QueryInterface) (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
    STDMETHOD_(ULONG,AddRef) (THIS)  PURE;
    STDMETHOD_(ULONG,Release) (THIS) PURE;

    // *** IShellPropSheetExt methods ***
    STDMETHOD(AddPages)(THIS_ LPFNADDPROPSHEETPAGE lpfnAddPage, LPARAM lParam) PURE;
};

typedef IShellPropSheetExt FAR* LPSHELLPROPSHEETEXT;


//=============================================================================
//
// IPersistFolder Interface
//
//  This interface is used by the Folder implementation of
// IMoniker::BindToObject when it is initializing a folder object.
//
```

```
//============================================================================

#undef  INTERFACE
#define INTERFACE   IPersistFolder

DECLARE_INTERFACE_(IPersistFolder, IPersist)    // fld
{
    // *** IUnknown methods ***
    STDMETHOD(QueryInterface) (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
    STDMETHOD_(ULONG,AddRef) (THIS)  PURE;
    STDMETHOD_(ULONG,Release) (THIS) PURE;

    // *** IPersist methods ***
    STDMETHOD(GetClassID) (THIS_ LPCLSID lpClassID) PURE;

    // *** IPersistFolder methods ***
    STDMETHOD(Initialize) (THIS_
                           LPCITEMIDLIST pidl) PURE;
};

typedef IPersistFolder FAR*     LPPERSISTFOLDER;


//============================================================================
//
// IExtractIcon interface
//
//  This interface is used in two different places in the shell.
//
//  First, it is used by the explorer to get the "icon location" of
// sub-folders from each shell folders. When the user expand a folder
// in the scope pane of the explorer, the explorer does following:
//  (1) binds to the folder,
//  (2) enumerates its sub-folders by calling its EnumObjects member,
//  (3) calls its GetUIObjectOf member to get IExtractIcon interface
//      for each sub-folders.
//  In this case, the explorer uses only IExtractIcon::GetIconLocation
// member to get the location of the appropriate icon. An icon location
// is always a file name and either an icon resource or an icon index.
//
//  Second, it is used by the shell when it extracts an icon image
// from a file. When the shell is extracting an icon from a file,
// it does following:
//  (1) creates the icon extraction handler object (by getting its CLSID
//      under the {ProgID}\shell\ExtractIconHanler key and calling
//      CoCreateInstance requesting for IExtractIcon interface).
//  (2) Calls IExtractIcon::GetIconLocation.
//  (4) If (3) returns S_FALSE, it calls IExtractIcon::ExtractIcon.
//  (5) If (5) returns S_OK, it recursively calls this logic with new location.
//
//  From extension programmer's point of view, there are only two cases
// where they provide implementations of IExtractIcon:
//  Case-1) providing explorer extensions (i.e., IShellFolder).
//  Case-2) providing per-instance icons for some types of files.
//
// Because Case-1 is described above, we'll explain only Case-2 here.
//
// When the shell is about display an icon for a file, it does following:
//  (1) Finds its ProgID and ClassID.
//  (2) If the file has a ClassID, it gets the icon location string from the
//      "DefaultIcon" key under it. The string indicates either per-class
//      icon (e.g., "FOOBAR.DLL,2") or per-instance icon (e.g., "%1,1").
//  (3) If a per-instance icon is specified, the shell creates an icon
```

```
//    extraction handler object for it, and extracts the icon from it
//    (which is described above).
//
//  It is important to note that the shell calls IExtractIcon::GetIconLocation
// first, then calls IExtractIcon::ExtractIcon. Most application programs
// (which support per-instance icons) will probably store an icon location
// (DLL/EXE name and index/id) rather than an icon image in each file.
// In those cases, programmer need to implement only the GetIconLocation member
// and does not need to implement the ExtractIcon member. They need to implement
// ExtractIcon member only if they decided to store the icon images within
// files themselved.
//
// IExtractIcon::GetIconLocation(uFlags, szIconFile, cchMax, piIndex, pwFlags)
//  This function returns an icon location.
// Parameters:
//  uFlags     [in]  --
//  szIconFile [out] -- Specifies the icon file
//  cchMax     [in]  -- Specifies the size of szIconFile (almost always MAX_PATH)
//  piIndex    [out] --
//  pwFlags    [out] --
// Notes:
//   When the explorer is getting an icon location, szIconFile and piIndex
//  are purely out parameters. When IExtractIcon::GetIconLocation returns
//  S_OK, it fills szIconFile and piIndex with an (another) icon location.
//
// IExtractIcon::ExtractIcon(pszFile, nIconIndex, phiconLarge, phiconSmall, nIcons)
//  This function extracts an icon image from a specified file.
// Parameters:
//  pszFile [in] -- Specifies the icon file.
//  nIconIndex [in] -- Specifies the icon index.
//  phiconLarge [out] --
//  phiconSmall [out] --
//  nIcons [in] --
//============================================================================

#undef  INTERFACE
#define INTERFACE   IExtractIcon

DECLARE_INTERFACE_(IExtractIcon, IUnknown)      // exic
{
    // *** IUnknown methods ***
    STDMETHOD(QueryInterface) (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
    STDMETHOD_(ULONG,AddRef) (THIS)  PURE;
    STDMETHOD_(ULONG,Release) (THIS) PURE;

    // *** IExtractIcon methods ***
    STDMETHOD(GetIconLocation)(THIS_
                          UINT   uFlags,
                          LPSTR  szIconFile,
                          UINT   cchMax,
                          int  FAR * piIndex,
                          UINT FAR * pwFlags) PURE;

    STDMETHOD(ExtractIcon)(THIS_
                            LPCSTR pszFile,
                            UINT   nIconIndex,
                            HICON  FAR *phiconLarge,
                            HICON  FAR *phiconSmall,
                            UINT   nIcons) PURE;
};

typedef IExtractIcon FAR*       LPEXTRACTICON;
```

```
//
// IShellLink Interface
//

// IShellLink::Resolve fFlags
#define SLR_NO_UI       0x0001
#define SLR_ANY_MATCH   0x0002

#undef  INTERFACE
#define INTERFACE   IShellLink

DECLARE_INTERFACE_(IShellLink, IUnknown)        // sl
{
    // *** IUnknown methods ***
    STDMETHOD(QueryInterface) (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
    STDMETHOD_(ULONG,AddRef) (THIS)  PURE;
    STDMETHOD_(ULONG,Release) (THIS) PURE;

    STDMETHOD(GetPath)(THIS_ LPSTR pszFile, int cchMaxPath, WIN32_FIND_DATA *pfd)
PURE;
    STDMETHOD(SetPath)(THIS_ LPCSTR pszFile, const WIN32_FIND_DATA *pfd, LPCSTR
pszRelPath, ULONG fFlags) PURE;

    STDMETHOD(GetIDList)(THIS_ LPCITEMIDLIST * ppidl) PURE;
    STDMETHOD(SetIDList)(THIS_ LPCITEMIDLIST pidl) PURE;

    STDMETHOD(GetDescription)(THIS_ LPSTR pszName, int cchMaxName) PURE;
    STDMETHOD(SetDescription)(THIS_ LPCSTR pszName) PURE;

    STDMETHOD(GetWorkingDirectory)(THIS_ LPSTR pszDir, int cchMaxPath) PURE;
    STDMETHOD(SetWorkingDirectory)(THIS_ LPCSTR pszDir) PURE;

    STDMETHOD(GetArguments)(THIS_ LPSTR pszArgs, int cchMaxPath) PURE;
    STDMETHOD(SetArguments)(THIS_ LPCSTR pszArgs) PURE;

    STDMETHOD(GetHotkey)(THIS_ WORD *pwHotkey) PURE;
    STDMETHOD(SetHotkey)(THIS_ WORD wHotkey) PURE;

    STDMETHOD(GetShowCmd)(THIS_ int *piShowCmd) PURE;
    STDMETHOD(SetShowCmd)(THIS_ int iShowCmd) PURE;

    STDMETHOD(GetIconLocation)(THIS_ LPSTR pszIconPath, int cchIconPath, int
*piIcon) PURE;
    STDMETHOD(SetIconLocation)(THIS_ LPCSTR pszIconPath, int iIcon) PURE;

    STDMETHOD(Resolve)(THIS_ HWND hwnd, UINT fFlags) PURE;

    STDMETHOD(Update)(THIS_ LPCSTR pszPathRel, UINT fFlags) PURE;
};


//
// ICopyHook Interface
//

#undef  INTERFACE
#define INTERFACE   ICopyHook

#ifndef FO_MOVE //these need to be kept in sync with the ones in shell.h

#define FO_MOVE           0x0001
#define FO_COPY           0x0002
```

```
#define FO_DELETE         0x0003
#define FO_RENAME         0x0004

#define FOF_CREATEPROGRESSDLG      1
#define FOF_CONFIRMMOUSE           2
#define FOF_SILENT                 4  // don't create progress/report
#define FOF_RENAMEONCOLLISION      8
#define FOF_NOCONFIRMATION        16  // Don't prompt the user.
#define FOF_WANTMAPPINGHANDLE     32  // Fill in SHFILEOPSTRUCT.hNameMappings
                                      // Must be freed using SHFreeNameMappings

typedef WORD FILEOP_FLAGS;

#define PO_DELETE       0x0001
#define PO_RENAME       0x0002

typedef WORD PRINTEROP_FLAGS;

#endif // FO_MOVE

DECLARE_INTERFACE_(ICopyHook, IUnknown) // sl
{
    // *** IUnknown methods ***
    STDMETHOD(QueryInterface) (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
    STDMETHOD_(ULONG,AddRef) (THIS)  PURE;
    STDMETHOD_(ULONG,Release) (THIS) PURE;

    STDMETHOD_(UINT,CopyCallback) (THIS_ HWND hwnd, WORD wFunc, WORD wFlags, LPCSTR
pszSrcFile, DWORD dwSrcAttribs,
                                   LPCSTR pszDestFile, DWORD dwDestAttribs) PURE;
};

typedef ICopyHook FAR*  LPCOPYHOOK;


//=========================================================================
//
// IFileViewer Interface, 3/4/94 kraigb
//
// Implemented in a FileViewer component object.  Used to tell a
// FileViewer to PrintTo or to view, the latter happening though
// ShowInitialize and Show.  The filename is always given to the
// viewer through IPersistFile.
//
//=========================================================================

#undef  INTERFACE
#define INTERFACE   IFileViewer

DECLARE_INTERFACE(IFileViewer)
    {
    STDMETHOD(QueryInterface) (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
    STDMETHOD_(ULONG,AddRef) (THIS)  PURE;
    STDMETHOD_(ULONG,Release) (THIS) PURE;

    STDMETHOD(ShowInitialize) (THIS) PURE;
    STDMETHOD(Show) (THIS_ int nCmdShow) PURE;
    STDMETHOD(PrintTo) (THIS_ LPSTR pszDriver, BOOL fSuppressUI) PURE;
    };

typedef IFileViewer FAR* LPFILEVIEWER;
```

```
//============================================================================
// The IShellBrowser/IShellView interface.
//============================================================================
//
// Defines the range of command IDs used by view windows.
//
//  FCIDM_SHVIEWFIRST/LAST      for the right pane (IShellView)
//  FCIDM_BROWSERFIRST/LAST     for the explorer frame (IShellBrowser)
//  FCIDM_GLOBAL/LAST           for the explorer's submenu IDs
//
#define FCIDM_SHVIEWFIRST           0x0000
#define FCIDM_SHVIEWLAST            0x7fff
#define FCIDM_BROWSERFIRST          0xa000
#define FCIDM_BROWSERLAST           0xbf00
#define FCIDM_GLOBALFIRST           0x8000
#define FCIDM_GLOBALLAST            0x9fff

//
// Global submenu IDs
//
#define FCIDM_MENU_FILE         (FCIDM_GLOBALFIRST+0x0000)
#define FCIDM_MENU_EDIT         (FCIDM_GLOBALFIRST+0x0040)
#define FCIDM_MENU_VIEW         (FCIDM_GLOBALFIRST+0x0080)
#define FCIDM_MENU_TOOLS        (FCIDM_GLOBALFIRST+0x00c0)
#define FCIDM_MENU_HELP         (FCIDM_GLOBALFIRST+0x0100)

//----------------------------------------------------------------------------
// Common to most viewers. ie These are the things that you would like
// to be global to all viewers.

typedef LPBYTE LPVIEWSETTINGS;

// NB Bitfields.
typedef enum
    {
    FWF_AUTOARRANGE =       0x0001,
    FWF_ABBREVIATEDNAMES =  0x0002,
    FWF_SNAPTOGRID =        0x0004,
    // FWF_UNUSED =         0x0008,
    FWF_BESTFITWINDOW =     0x0010,
    FWF_DESKTOP =           0x0020,
        FWF_SINGLESEL =         0x0040,
    } FOLDERFLAGS;

typedef enum
    {
    FVM_ICON =              1,
    FVM_SMALLICON =         2,
    FVM_LIST =              3,
    FVM_DETAILS =           4,
    } FOLDERVIEWMODE;

typedef struct
    {
    UINT ViewMode;      // View mode (FOLDERVIEWMODE values)
    UINT fFlags;        // View options (FOLDERFLAGS bits)
    } FOLDERSETTINGS, NEAR *PFOLDERSETTINGS, FAR *LPFOLDERSETTINGS;

typedef FOLDERSETTINGS const FAR* LPCFOLDERSETTINGS;

//
// Interface:   IShellBrowser
//
```

```
// History:
//  01-08-93 GeorgeP     Created.
//

// Values for uWindow parameter of GetWindow/GetWindowRect member function.
//
#define FCW_STATUS      0x0001
#define FCW_TOOLBAR     0x0002

// SetToolbarItems flags
#define FCT_MERGE       0x0001
#define FCT_CONFIGABLE  0x0002
#define FCT_ADDTOEND    0x0004

// Values for wFlags parameter of IShellBrowser::SetPath() member
#define SBSP_DEFBROWSER  0x0000
#define SBSP_SAMEBROWSER 0x0001
#define SBSP_NEWBROWSER  0x0002
#define SBSP_ABSOLUTE    0x00000000
#define SBSP_RELATIVE    0x10000000
#define SBSP_PARENT      0x20000000


//---------------------------------------------------------------------------
// IShellBrowser interface
//
//  IShellBrowser interface is the interface that is provided by the shell
// explorer/folder frame window. When it creates the "contents pane" of
// a shell folder (which provider IShellFolder interface), it calls its
// CreateViewObject member function to create an IShellView object. Then,
// it calls its CreateViewWindow member to create the "contents pane"
// window. The pointer to the IShellBrowser interface is passed to
// the IShellView object as a parameter to this CreateViewWindow member
// function call.
//
//    +--------------------------+  <-- Explorer window
//    | [] Explorer              |
//    |--------------------------+       IShellBrowser
//    | File View ..             |
//    |--------------------------|
//    |         |                |
//    |         |              <-------- Content pane
//    |         |                |
//    |         |                |       IShellView
//    |         |                |
//    |         |                |
//    +--------------------------+
//
//
// Member functions:
//
//  IShellBrowser::GetWindow(phwnd)
//    Inherited from IOleWindow::GetWindow.
//
//  IShellBrowser::ContextSensitiveHelp(fEnterMode)
//    Inherited from IOleWindow::ContextSensitiveHelp.
//
//  IShellBrowser::InsertMenus(hmenuShared, lpMenuWidths)
//    Similar to the IOleInPlaceFrame::InsertMenus. The explorer will put
//   "File" and "Edit" pulldown in the File menu group, "View" and "Tools"
//    in the Container menu group and "Help" in the Window menu group. Each
//    pulldown menu will have a uniqu ID, FCIDM_MENU_FILE/EDIT/VIEW/TOOLS/HELP.
//    The view is allowed to insert menuitems into those sub-menus by those
```

```
//    IDs must be between FCIDM_SHVIEWFIRST and FCIDM_SHVIEWLAST.
//
//   IShellBrowser::SetMenu(hmenuShared, holemenu, hwndActiveObject)
//     Similar to the IOleInPlaceFrame::SetMenu. The explorer ignores the
//    holemenu parameters and performs menu-dispatch based on the menuitem
//    IDs. It is important to note that the explorer will add different
//    set of menuitems depending on whether the view has a focus or not.
//    Therefore, it is very important to call OnViewActivate whenever
//    the view window (or its children) gets the focus.
//
//   IShellBrowser::RemoveMenus(hmenuShared)
//     Same as the IOleInPlaceFrame::RemoveMenus.
//
//   IShellBrowser::SetStatusText(lpszStatusText)
//     Same as the IOleInPlaceFrame::SetStatusText. It is also possible to
//    send messages directly to the status window via SendControlMsg.
//
//   IShellBrowser::EnableModeless(fEnable)
//     Same as the IOleInPlaceFrame::EnableModeless.
//
//   IShellBrowser::TranslateAccelerator(lpmsg, wID)
//     Same as the IOleInPlaceFrame::TranslateAccelerator, but will be
//    never called because we don't support EXEs. This member function
//    is defined here for possible future enhancement.
//
//   IShellBrowser::BrowseObject(pmk, wFlags)
//     The view calls this member to let shell explorer browse to another
//    folder. "pmk" is an absolute moniker (from the desktop), which specifies
//    the folder to be browsed. "wFlags" is not used at this point.
//
//   IShellBrowser::GetViewStateStream(grfMode, ppstm)
//     The browser returns an IStream interface for the view state information.
//    "grfMode" specifies the read/write access (STGM_READ, STGM_WRITE or
//    STGM_READWRITE). Note that it might return NULL in *ppstm, which means
//    no need to save/read view information.
//
//   IShellBrowser::GetControlWindow(id, phwnd)
//     The shell view may call this member function to get the window handle
//    of Explorer controls (toolbar or status winodw -- FCW_TOOLBAR or
//    FCW_STATUS).
//
//   IShellBrowser::SendControlMsg(id, uMsg, wParam, lParam, pret)
//     The shell view calls this member function to send control messages to
//    one of Explorer controls (toolbar or status window -- FCW_TOOLBAR or
//    FCW_STATUS).
//
//   IShellBrowser::QueryActiveShellView(IShellView FAR* ppshv)
//     This member returns currently activated (displayed) shellview object.
//    A shellview never need to call this member function.
//
//   IShellBrowser::OnViewWindowActive(pshv)
//     The shell view window calls this member function when the view window
//    (or one of its children) got the focus. It MUST call this member before
//    calling IShellBrowser::InsertMenus, because it will insert different
//    set of menu items depending on whether the view has the focus or not.
//
//   IShellBrowser::AddViewPropertySheetPages(dwReserved, lpfn, lParam)
//     Currently not used.
//
//   IShellBrowser::SetToolbarItems(lpButtons, nButtons, uFlags)
//     The view calls this function to add toolbar items to the exporer's
//    toolbar. "lpButtons" and "nButtons" specifies the array of toolbar
//    items. "uFlags" must be one of FCT_MERGE, FCT_CONFIGABLE, FCT_ADDTOEND.
```

```
//
//--------------------------------------------------------------------------

#undef  INTERFACE
#define INTERFACE   IShellBrowser

DECLARE_INTERFACE_(IShellBrowser, IOleWindow)
{
    // *** IUnknown methods ***
    STDMETHOD(QueryInterface) (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
    STDMETHOD_(ULONG,AddRef) (THIS)  PURE;
    STDMETHOD_(ULONG,Release) (THIS) PURE;

    // *** IOleWindow methods ***
    STDMETHOD(GetWindow) (THIS_ HWND FAR* lphwnd) PURE;
    STDMETHOD(ContextSensitiveHelp) (THIS_ BOOL fEnterMode) PURE;

    // *** IShellBrowser methods *** (same as IOleInPlaceFrame)
    STDMETHOD(InsertMenus) (THIS_ HMENU hmenuShared, LPOLEMENUGROUPWIDTHS
lpMenuWidths) PURE;
    STDMETHOD(SetMenu) (THIS_ HMENU hmenuShared, HOLEMENU holemenu, HWND
hwndActiveObject) PURE;
    STDMETHOD(RemoveMenus) (THIS_ HMENU hmenuShared) PURE;
    STDMETHOD(SetStatusText) (THIS_ LPCOLESTR lpszStatusText) PURE;
    STDMETHOD(EnableModeless) (THIS_ BOOL fEnable) PURE;
    STDMETHOD(TranslateAccelerator) (THIS_ LPMSG lpmsg, WORD wID) PURE;

    // *** IShellBrowser methods ***
    STDMETHOD(BrowseObject)(THIS_ LPMONIKER pmk, UINT wFlags) PURE;
    STDMETHOD(GetViewStateStream)(THIS_ DWORD grfMode, LPSTREAM FAR *ppStrm) PURE;
    STDMETHOD(GetControlWindow)(THIS_ UINT id, HWND FAR* lphwnd) PURE;
    STDMETHOD(SendControlMsg)(THIS_ UINT id, UINT uMsg, WPARAM wParam, LPARAM
lParam, LRESULT FAR* pret) PURE;
    STDMETHOD(QueryActiveShellView)(THIS_ struct IShellView FAR** ppshv) PURE;
    STDMETHOD(OnViewWindowActive)(THIS_ struct IShellView FAR* ppshv) PURE;
    STDMETHOD(AddViewPropertySheetPages)(THIS_ DWORD dwReserved,
                                LPFNADDPROPSHEETPAGE lpfn, LPARAM lparam) PURE;
    STDMETHOD(SetToolbarItems)(THIS_ LPTBBUTTON lpButtons, UINT nButtons, UINT
uFlags) PURE;
};

typedef IShellBrowser FAR* LPSHELLBROWSER;

//--------------------------------------------------------------------------
// ICommDlgBrowser interface
//
//  ICommDlgBrowser interface is the interface that is provided by the new
// common dialog window to hook and modify the behavior of IShellView.  When
// a default view is created, it queries its parent IShellBrowser for the
// ICommDlgBrowser interface.  If supported, it calls out to that interface
// in several cases that need to behave differently in a dialog.
//
// Member functions:
//
//  ICommDlgBrowser::OnKeyDown(WORD wVKey, LPBOOL pfProcessed)
//    Called when a virtual key is pressed in the view.  The browser should
//   return TRUE in *pfProcessed if it processed the key itself, FALSE to let
//   the view do the default action.  Used to capture things such as the Del
//   key.
//
//  ICommDlgBrowser::OnDefaultCommand(LPBOOL pfProcessed)
//    Called when the user double-clicks in the view or presses Enter.  The
//   browser should return TRUE if it processed the action itself, or FALSE
```

```
//    to let the view perform the default action.
//
#if 0
//   ICommDlgBrowser::OnDragDropMsg(UINT iMessage, WPARAM wParam,
//                                  LPDROPSTRUCT lpds, LRESULT FAR *lplResult,
//                                  LPBOOL pfProcessed)
//     Called to validate drag/drop messages.  The browser should return TRUE
//    in *pfProcessed if it processed the message;  *lplResult should be set to
//    the value to be returned for the message.  The browser should return FALSE
//    in *pfProcessed to allow the view to perform the default action.  This
//    function is called on all standard drag/drop messages:  WM_DRAGSELECT,
//    WM_QUERYDROPOBJECT, etc.
#endif
//
//   ICommDlgBrowser::AllowBeginDrag(LPBOOL pfAllow)
//     Called to ask whether objects can be dragged out of the view.  *pfAllow
//    should be set to TRUE to allow dragging, FALSE to disallow.
//
//   ICommDlgBrowser::AllowDrop(LPBOOL pfAllow)
//     Called when the view is created to determine whether the view should be
//    a valid shell drop target.  Set *pfAllow to TRUE to allow dropping on the
//    view, FALSE to disallow.
//
//   ICommDlgBrowser::OnSelectionChange()
//     Called when the selection in the view changes.  The browser may send
//    the view a message to determine what is currently selected.  This call
//    is made after the selection has changed.
//
//   ICommDlgBrowser::IncludeObject(LPCITEMIDLIST pidl, LPBOOL pfInclude)
//     Called when the view is enumerating objects.  'pidl' is a relative
//    IDLIST.  The browser should set *pfInclude to TRUE to include the
//    object in the view, FALSE to hide it.
//
//---------------------------------------------------------------------------

#undef  INTERFACE
#define INTERFACE   ICommDlgBrowser

DECLARE_INTERFACE_(ICommDlgBrowser, IUnknown)
{
    // *** IUnknown methods ***
    STDMETHOD(QueryInterface) (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
    STDMETHOD_(ULONG,AddRef) (THIS)  PURE;
    STDMETHOD_(ULONG,Release) (THIS) PURE;

    // *** ICommDlgBrowser methods ***

    STDMETHOD(OnKeyDown) (THIS_ WORD wVKey, LPBOOL pfProcessed) PURE;
    STDMETHOD(OnDefaultCommand) (THIS_ LPBOOL pfProcessed) PURE;
    STDMETHOD(OnDragDropMsg) (THIS_ UINT iMessage, WPARAM wParam,
                              LPDROPSTRUCT lpds, LRESULT FAR *lplResult,
                                                 LPBOOL pfProcessed) PURE;
    STDMETHOD(AllowBeginDrag) (THIS_ LPBOOL pfAllow) PURE;
    STDMETHOD(AllowDrop) (THIS_ LPBOOL pfAllow) PURE;
    STDMETHOD(OnSelectionChange) (THIS) PURE;
    STDMETHOD(IncludeObject) (THIS_ LPCITEMIDLIST pidl, LPBOOL pfInclude) PURE;
};

typedef ICommDlgBrowser FAR* LPCOMMDLGBROWSER;

//============================================================================
//
// Interface:   IShellView
```

```
//
// History:
//  01-07-93 GeorgeP     Created.
//
//============================================================================

// shellview select item flags
#define SVSI_DESELECT   0x0000
#define SVSI_SELECT     0x0001
#define SVSI_EDIT       0x0003  // includes select
#define SVSI_DESELECTOTHERS 0x0004
#define SVSI_ENSUREVISIBLE  0x0008

//----------------------------------------------------------------------------
// IShellView interface
//
// IShellView::GetWindow(phwnd)
//   Inherited from IOleWindow::GetWindow.
//
// IShellView::ContextSensitiveHelp(fEnterMode)
//   Inherited from IOleWindow::ContextSensitiveHelp.
//
// IShellView::TranslateAccelerator(lpmsg)
//   Similar to IOleInPlaceActiveObject::TranlateAccelerator. The explorer
//  calls this function BEFORE any other translation. Returning S_OK
//  indicates that the message was translated (eaten) and should not be
//  translated or dispatched by the explorer.
//
// IShellView::EnableModeless(fEnable)
//   Similar to IOleInPlaceActiveObject::EnableModeless.
//
// IShellView::UIActivate(uState)
//   The explorer calls this member function whenever the activation
//  state of the view window is changed by a certain event that is
//  NOT caused by the shell view itself. The SVUIA_DEACTIVATE will be
//  passed when the explorer is about to destroy the shell view window;
//  the shell view is supposed to remove all the UIs (typically merged
//  menu and modeless popup windows). The SVUIA_ACTIVATE_NOFOCUS will be
//  passsed when the shell view is losing the input focus or the shell
//  view has been just created without the input focus; the shell view
//  is supposed to set menuitems appropriate for non-focused state.
//  The SVUIA_ACTIVATE_FOCUS will be passed when the explorer has just
//  created the view window with the input focus; the shell view is
//  supposed to set menuitems appropriate for focused state.
//   The shell view should not change focus within this member function.
//  The shell view should not hook the WM_KILLFOCUS message to remerge
//  menuitems. However, the shell view typically hook the WM_SETFOCUS
//  message, and re-merge the menu after calling IShellBrowser::
//  OnViewWindowActivated.
//
// IShellView::SaveViewState()
//   The explorer calls this member when the shell view is supposed to
//  store its view settings. The shell view is supposed to get a view
//  stream by calling IShellBrowser::GetViewStateStream and store the
//  current view state into the stream.
//
// IShellView::SelectItem(lpvID, uFlags)
//   The explorer calls this member to change the selection state of
//  object(s) within the shell view.
//----------------------------------------------------------------------------

#undef  INTERFACE
#define INTERFACE   IShellView
```

```
DECLARE_INTERFACE_(IShellView, IOleWindow)
{
    // *** IUnknown methods ***
    STDMETHOD(QueryInterface) (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
    STDMETHOD_(ULONG,AddRef) (THIS)  PURE;
    STDMETHOD_(ULONG,Release) (THIS) PURE;

    // *** IOleWindow methods ***
    STDMETHOD(GetWindow) (THIS_ HWND FAR* lphwnd) PURE;
    STDMETHOD(ContextSensitiveHelp) (THIS_ BOOL fEnterMode) PURE;

    // *** IShellView methods ***
    STDMETHOD(TranslateAccelerator) (THIS_ LPMSG lpmsg) PURE;
    STDMETHOD(EnableModeless) (THIS_ BOOL fEnable) PURE;
    STDMETHOD(UIActivate) (THIS_ UINT uState) PURE;
    STDMETHOD(Refresh) (THIS) PURE;

    STDMETHOD(CreateViewWindow)(THIS_ IShellView FAR *lpPrevView,
        LPCFOLDERSETTINGS lpfs, IShellBrowser FAR * psb,
        RECT FAR* prcView, HWND FAR *phWnd) PURE;
    STDMETHOD(DestroyViewWindow)(THIS) PURE;
    STDMETHOD(GetCurrentInfo)(THIS_ LPFOLDERSETTINGS lpfs) PURE;
    STDMETHOD(ForwardControlMsg)(THIS_ UINT id, UINT uMsg, WPARAM wParam, LPARAM
lParam,
                      LRESULT FAR* pret) PURE;
    STDMETHOD(AddPropertySheetPages)(THIS_ DWORD dwReserved,
                                LPFNADDPROPSHEETPAGE lpfn, LPARAM lparam) PURE;
    STDMETHOD(SaveViewState)(THIS) PURE;
    STDMETHOD(SelectItem)(THIS_ LPCVOID lpvID, UINT uFlags) PURE;
};

typedef IShellView FAR* LPSHELLVIEW;

//
// uState values for IShellView::UIActivate
//
typedef enum {
    SVUIA_DEACTIVATE       = 0,
    SVUIA_ACTIVATE_NOFOCUS = 1,
    SVUIA_ACTIVATE_FOCUS   = 2
} SVUIA_STATUS;

//=============================================================================
// The IShellFolder interface.
//=============================================================================

// structure for returning strings from member functions
#define STRRET_OLESTR   0x0000
#define STRRET_OFFSET   0x0001
#define STRRET_CSTR     0x0002

typedef struct _STRRET
{
        UINT uType;     // One of the STRRET_* values
        union
        {
                LPOLESTR        pOleStr;        // OLESTR that will be freed
                UINT            uOffset;        // Offset into SHITEMID
                char            cStr[MAX_PATH]; // Buffer to fill in
        };
} STRRET, FAR *LPSTRRET;
```

```
//
// IDMoniker related functions
//
HRESULT WINAPI SHCreateIDMoniker(LPCITEMIDLIST pidl, LPMONIKER FAR* ppmk);
HRESULT WINAPI SHGetIDListFromIDMoniker(LPMONIKER pmk, LPITEMIDLIST FAR* ppidlOut);

// IDList related functions
BOOL WINAPI SHGetPathFromIDList(LPCITEMIDLIST pidl, LPSTR pszPath);

// SHGetFileIcon API
#define SHGFI_LARGEICON         0x000000000
#define SHGFI_SMALLICON         0x000000001
#define SHGFI_OPENICON          0x000000002
HICON WINAPI SHGetFileIcon(HINSTANCE hinst, LPCSTR pszPath, DWORD dwFileAttribute,
UINT uFlags);


//--------------------------------------------------------------------------
// IShellFolder interface
//
// Member functions:
//
// IShellFolder::BindToObject(pidl, pbc, riid, ppvOut)
//   This function returns an instance of a sub-folder which is specified
//  by the IDList (pidl).
//
// IShellFolder::BindToStorage(pidl, pbc, riid, ppvObj)
//   This function returns a storage instance of a sub-folder which is
//  specified by the IDList (pidl). The shell never calls this member
//  function in the first release of Chicago.
//
// IShellFolder::CompareIDs(lParam, pidl1, pidl2)
//   This function compares two IDLists and returns the result. The shell
//  explorer always passes 0 as lParam, which indicates "sort by name".
//  It should return 0 (as CODE of the scode), if two id indicates the
//  same object; negative value if pidl1 should be placed before pidl2;
//  positive value if pidl2 should be placed before pidl1.
//
// IShellFolder::CreateViewObject(hwndOwner, riid, ppvOut)
//   This function creates a view object. The shell browser always passes
//  IID_IShellView as riid. "hwndOwner" can be used  as the owner
//  window of its dialog box or menu during the lifetime
//  of the view object. This member function should always create a new
//  instance which has only one reference count. The explorer may create
//  more than one instances of view object from one shell folder object
//  and treat them as separate instances.
//
// IShellFolder::GetAttributesOf(cidl, apidl, prgfInOut)
//   This function returns the attributes of specified objects in that
//  folder. "cidl" and "apidl" specifies objects. "apidl" contains only
//  simple IDLists. The explorer initializes *prgfInOut with a set of
//  flags to be evaluated. The shell folder may optimize the operation
//  by not returning unspecified flags.
//
// IShellFolder::GetUIObjectOf(hwndOwner, cidl, apidl, riid, prgfInOut, ppvOut)
//   This function creates a UI object to be used for specified objects.
//  The shell explorer passes either IID_IDataObject (for transfer operation)
//  or IID_IContextMenu (for context menu operation) as riid.
//
// IShellFolder::GetDisplayNameOf
//   This function returns the display name of the specified object.
//  If the ID contains the display name (in the locale character set),
//  it returns the offset to the name. Otherwise, it returns a pointer
```

```
//  to the display name string (UNICODE), which is allocated by the
//  task allocator, or fills in a buffer.
//
// IShellFolder::SetNameOf
//   This function sets the display name of the specified object.
//  If it changes the ID as well, it returns the new ID which is
//  alocated by the task allocator.
//
// IShellFolder::MonikerToIDList
//   This function creates a corresponding IDList from the specified relative
//  moniker. Typically, it simply calls SHGetIDFromIDMoniker.
//
//------------------------------------------------------------------------

#define LPSHELLFOLDER      IShellFolder FAR*

#undef  INTERFACE
#define INTERFACE       IShellFolder

DECLARE_INTERFACE_(IShellFolder, IOleContainer)
{
    // *** IUnknown methods ***
    STDMETHOD(QueryInterface) (THIS_ REFIID riid, LPVOID FAR* ppvObj) PURE;
    STDMETHOD_(ULONG,AddRef) (THIS)  PURE;
    STDMETHOD_(ULONG,Release) (THIS) PURE;

    // *** IParseDisplayName method ***
    STDMETHOD(ParseDisplayName) (THIS_ LPBC pbc, LPOLESTR lpszDisplayName,
        ULONG FAR* pchEaten, LPMONIKER FAR* ppmkOut) PURE;

    // *** IOleContainer methods ***
    STDMETHOD(EnumObjects) ( THIS_ DWORD grfFlags, LPENUMUNKNOWN FAR* ppenumUnknown)
PURE;
    STDMETHOD(LockContainer) (THIS_ BOOL fLock) PURE;

    // *** IShellFolder methods ***
    STDMETHOD(BindToObject)     (THIS_ LPCITEMIDLIST pidl, LPBC pbc,
                                 REFIID riid, LPVOID FAR* ppvOut) PURE;
    STDMETHOD(BindToStorage)    (THIS_ LPCITEMIDLIST pidl, LPBC pbc,
                                 REFIID riid, LPVOID FAR* ppvObj) PURE;
    STDMETHOD(CompareIDs)       (THIS_ LPARAM lParam, LPCITEMIDLIST pidl1,
LPCITEMIDLIST pidl2) PURE;
    STDMETHOD(CreateViewObject) (THIS_ HWND hwndOwner, REFIID riid, LPVOID FAR*
ppvOut) PURE;
    STDMETHOD(GetAttributesOf)  (THIS_ UINT cidl, LPCITEMIDLIST FAR* apidl,
                                     ULONG FAR* rgfInOut) PURE;
    STDMETHOD(GetUIObjectOf)    (THIS_ HWND hwndOwner, UINT cidl, LPCITEMIDLIST FAR*
apidl,
                                 REFIID riid, UINT FAR* prgfInOut, LPVOID FAR*
ppvOut) PURE;
    STDMETHOD(GetDisplayNameOf) (THIS_ LPCITEMIDLIST pidl, DWORD uFlags, LPSTRRET
lpName) PURE;
    STDMETHOD(SetNameOf)        (THIS_ HWND hwndOwner, LPCITEMIDLIST pidl,
                                 LPCOLESTR lpszName, DWORD uFlags,
                                 LPITEMIDLIST FAR* ppidlOut) PURE;
    STDMETHOD(MonikerToIDList)  (THIS_ LPMONIKER pmk, LPITEMIDLIST FAR* pidl) PURE;
};

// IShellFolder::GetDisplayNameOf/SetNameOf uFlags
typedef enum tagSHGDN
{
    SHGDN_NORMAL            = 0,        // plain
    SHGDN_NOEXTENSION       = 1,        // no extension
```

```
    SHGDN_INFOLDER          = 2,        // displayed under a folder
} SHGNO;

// IShellFolder::EnumObjects
typedef enum tagSHCONTF
{
    SHCONTF_FOLDERS         = 32,       // for shell browser
    SHCONTF_NONFOLDERS      = 64,       // for default view
    SHCONTF_INCLUDEHIDDEN   = 128,      // for hidden/system objects
} SHCONTF;

// IShellFolder::GetAttributesOf flags
#define SFGAO_CANCOPY           DROPEFFECT_COPY // Objects can be copied
#define SFGAO_CANMOVE           DROPEFFECT_MOVE // Objects can be moved
#define SFGAO_CANLINK           DROPEFFECT_LINK // Objects can be linked
#define SFGAO_CANRENAME         0x00000010L     // Objects can be renamed
#define SFGAO_CANDELETE         0x00000020L     // Objects can be deleted
#define SFGAO_HASPROPSHEET      0x00000040L     // Objects has property sheets
#define SFGAO_CAPABILITYMASK    0x00000077L
#define SFGAO_LINK              0x00010000L     // linked
#define SFGAO_SHARE             0x00020000L     // shared
#define SFGAO_READONLY          0x00040000L     // read-only
#define SFGAO_GHOSTED           0x00080000L     // ghosted icon
#define SFGAO_DISPLAYATTRMASK   0x000F0000L
#define SFGAO_FILESYSANCESTOR   0x10000000L     // It contains file system folder
#define SFGAO_FOLDER            0x20000000L     // It's a folder.
#define SFGAO_FILESYSTEM        0x40000000L     // is a file system thing
(file/folder/root)
#define SFGAO_HASSUBFOLDER      0x80000000L     // Expandable in the map pane
#define SFGAO_CONTENTSMASK      0x80000000L
#define SFGAO_VALIDATE          0x01000000L     // invalidate cached information

// Bitfield validation
#ifdef DROPEFFECT_COPY
#if ((DROPEFFECT_COPY|DROPEFFECT_MOVE|DROPEFFECT_LINK)!=0x00000007L)
#error Definitions of DROPEFFECT flags has changed.
#endif
#endif // DROPEFFECT_COPY

/* This is the interface for a browser to "subclass" the main File Cabinet
** window.  Note that only the hwnd, message, wParam, and lParam fields of
** the msg structure are used.  The browser window will get a WM_NOTIFY
** message with NULL ID, FCN_MESSAGE as the code, and a far pointer to
** FCMSG_NOTIFY as the lParam.
*/
typedef struct tagFCMSG_NOTIFY
{
        NMHDR   hdr;
        MSG     msg;
        LRESULT lResult;
} FCMSG_NOTIFY;

#define FCN_MESSAGE     (100)


//---------------------------------------------------------------------------
// messages that can be send to the cabinet by other apps
//
//---------------------------------------------------------------------------

#define NF_INHERITVIEW  0x0000
#define NF_LOCALVIEW    0x0001
```

```
// Change the path of an existing folder.
// wParam:
//      0:              LPARAM is a string, handle the message immediately.
//      CSP_HANDLE:     LPARAM is a handle. handle the message immediatelt
//                      and then free the handle.
//      CSP_REPOST:     LPARAM is a string, copy the string and handle the
//                      message later.
//      CSP_REPOST|CSP_HANDLE:
//                      LPARAM is a handle, just handle the message later
//                      and free the handle then.
// lParam: LPSTR or HANDLE of path.
//
#define CSP_REPOST      0x0001
#define CSP_HANDLE      0x0002
#define CWM_SETPATH             (WM_USER + 2)

// lpsv points to the Shell View extension that requested idle processing
// uID is an app define identifier for the processor
// returns: TRUE if there is more idle processing necessary, FALSE if all done
// Note that the idle processor should do one "atomic" operation and return
// as soon as possible.
typedef BOOL (CALLBACK *FCIDLEPROC)(void FAR *lpsv, UINT uID);

// Inform the File Cabinet that you want idle messages.
// This should ONLY be used by File Cabinet extensions.
// wParam: app define UINT (passed to FCIDLEPROC).
// lParam: pointer to an FCIDLEPROC.
// return: TRUE if successful; FALSE otherwise
//
#define CWM_WANTIDLE            (WM_USER + 3)

// get or set the FOLDERSETTINGS for a view
// wParam: BOOL TRUE -> set to view info buffer, FALSE -> get view info buffer
// lParam: LPFOLDERSETTINGS buffer to get or set view info
//
#define CWM_GETSETCURRENTINFO   (WM_USER + 4)
#define FileCabinet_GetSetCurrentInfo(_hwnd, _bSet, _lpfs) \
        SendMessage(_hwnd, CWM_GETSETCURRENTINFO, (WPARAM)(_bSet), \
        (LPARAM)(LPFOLDERSETTINGS)_lpfs)

// selects the specified item in the current view
// wParam: BOOL TRUE -> select, FALSE -> deselect
// lParam: LPCSTR of the item ID (not display name), NULL -> all items
//
#define CWM_SELECTITEM          (WM_USER + 5)

// tells the window to punt its wait cursor.  used for handoff
// while thread inits
#define CWM_STOPWAITING         (WM_USER + 6)

// Get the IShellBrowser object associated with an hwndMain
#define CWM_GETISHELLBROWSER    (WM_USER + 7)
#define FileCabinet_GetIShellBrowser(_hwnd) \
        (IShellBrowser FAR *)SendMessage(_hwnd, CWM_GETISHELLBROWSER, 0, 0L)


//====================================================================
// Clipboard format which may be supported by IDataObject from system
// defined shell folders (such as directories, network, ...).
//====================================================================

#define CFSTR_SHELLIDLIST       "Shell IDList Array"
#define CFSTR_NETRESOURCES      "Net Resource"
```

```
#define DVASPECT_SHORTNAME      2 // use for CF_HDROP to get short name version
//
// Data format of CFSTR_NETRESOURCES.
//
typedef struct _NRESARRAY       // anr
{
    UINT cItems;
    NETRESOURCE nr[1];
} NRESARRAY, FAR* LPNRESARRAY;

//
// Data format of CFSTR_SHELLIDLIST
//
typedef struct _IDA {
    UINT cidl;          // number of relative IDList
    UINT aoffset[1];    // [0]: folder IDList, [1]-[cidl]: item IDList
} CIDA, FAR* LPIDA;

#ifdef __cplusplus
}
#endif  /* __cplusplus */

#ifndef RC_INVOKED
#pragma pack()
#endif  /* !RC_INVOKED */

#endif // _SHLOBJ_H_
```