Baggage Access Functions

To access the data from the Help file's internal file system, Windows Help provides specialized source code. This source code can be built into an application or custom DLL so that it can retrieve the appropriate data files listed

Function Calls

in the Help file's [BAGGAGE] section.

Use the following function calls to access the Help file system:

Open
Close

Use the following function calls to access baggage files in the Help file system:

Open
Close
Read
Tell
Seek
Create
ErrorW
ErrorPstr
LGetInfo
FAPI
LLFile
RCLLInfoFromHF
RCLLInfoFromHFS

The Baggage Include File

The include file for these function calls is provided in the following section.

Use the following include file, IMPDLL.H, to enable your application or DLL to

retrieve baggage files from the Help file system.

```
/*****************************************************************************\
*
*  IMPDLL.H
*
*  Copyright (C) Microsoft Corporation 1990.
*  All rights reserved.
*
******************************************************************************
*
*  Program Description:   Import routine export header for DLLs
*
******************************************************************************
*
*  Revision History:   Created 10/22/90 by Robert Bunney
*
*
******************************************************************************
*
*  Known Bugs:   None
*
*
\*****************************************************************************/

/*****************************************************************************\
*
*  Defines
*
\*****************************************************************************/

/* magic number and version number */

/* file mode flags */

#define fFSReadOnly             (BYTE)0x01    /* file (FS) is readonly */
#define fFSOpenReadOnly         (BYTE)0x02    /* file (FS) is opened in readonly mode */

#define fFSReadWrite            (BYTE)0x00    /* file (FS) is readwrite */
#define fFSOpenReadWrite        (BYTE)0x00    /* file (FS) is opened in read/write mode */


/* flags for FlushHfs */

#define fFSCloseFile            (BYTE)0x01
#define fFSFreeBtreeCache       (BYTE)0x02

/* seek origins */

#define wFSSeekSet              0
#define wFSSeekCur              1
#define wFSSeekEnd              2

/* low level info options */

#define wLLSameFid              0
#define wLLDupFid               1
#define wLLNewFid               2
```

```
/* Exported functions       */

#define HE_Count                26      /* count of exported functions */
#define HE_Documented           17      /* number documented for Help 3.1 */

#define HE_NotUsed              0
#define HE_HfsOpenSz            1
#define HE_RcCloseHfs           2
#define HE_HfOpenHfs            3
#define HE_RcCloseHf            4
#define HE_LcbReadHf            5
#define HE_LTellHf              6
#define HE_LSeekHf              7
#define HE_FEofHf               8
#define HE_LcbSizeHf            9
#define HE_FAccessHfs           10
#define HE_RcLLInfoFromHf       11
#define HE_RcLLInfoFromHfs      12
#define HE_ErrorW               13
#define HE_ErrorSz              14
#define HE_GetInfo              15
#define HE_API                  16

/* Return codes          */

#define rcSuccess               0
#define rcFailure               1
#define rcExists                2
#define rcNoExists              3
#define rcInvalid               4
#define rcBadHandle             5
#define rcBadArg                6
#define rcUnimplemented         7
#define rcOutOfMemory           8
#define rcNoPermission          9
#define rcBadVersion            10
#define rcDiskFull              11
#define rcInternal              12
#define rcNoFileHandles         13
#define rcFileChange            14
#define rcTooBig                15


/**
* Errors for Error()
**/

/* Errors to generate       */

#define wERRS_OOM               2       /* Out of memory                */
#define wERRS_NOHELPPS          3       /* No Help during printer setup */
#define wERRS_NOHELPPR          4       /* No Help while printing       */
#define wERRS_FNF               1001    /* Cannot find file             */
#define wERRS_NOTOPIC           1002    /* Topic does not exist         */
#define wERRS_NOPRINTER         1003    /* Cannot print                 */
#define wERRS_PRINT             1004
#define wERRS_EXPORT            1005    /* Cannot copy to clipboard     */
#define wERRS_BADFILE           1006
#define wERRS_OLDFILE           1007
```

```
#define wERRS_Virus                 1011   /* Bad .EXE                          */
#define wERRS_BADDRIVE              1012   /* Invalid drive                     */
#define wERRS_WINCLASS              1014   /* Bad window class                  */
#define wERRS_BADKEYWORD            3012   /* Invalid keyword                   */
#define wERRS_BADPATHSPEC           3015   /* Invalid path specification        */
#define wERRS_PATHNOTFOUND          3017   /* Path not found                    */
#define wERRS_DIALOGBOXOOM          3018   /* Insufficient memory for dialog    */
#define wERRS_DiskFull              5001   /* Disk is full                      */
#define wERRS_FSReadWrite           5002   /* File read/write failure           */


/**
* Actions for LGetInfo()
**/

#define GI_NOTHING              0       /* Not used                              */
#define GI_INSTANCE             1       /* Application instance handle           */
#define GI_MAINHWND             2       /* Main window handle                    */
#define GI_CURRHWND             3       /* Current window handle                 */
#define GI_HFS                  4       /* Handle to file system in use          */
#define GI_FGCOLOR              5       /* Foreground color used by application  */
#define GI_BKCOLOR              6       /* Background color used by application  */
#define GI_TOPICNO              7       /* Topic number                          */
#define GI_HPATH                8       /* Handle containing path -- caller      */

                                        /*  must free          */


/*****************************************************************************\
*
*           Types
*
\*****************************************************************************/

typedef WORD RC;                /* Error return (return code)    */
typedef HANDLE HFS;             /* Handle to a file system       */
typedef HANDLE HF;              /* Handle to a file system file  */

/*****************************************************************************\
*
*           Prototypes
*
\*****************************************************************************/

VOID FAR PASCAL SetCallbacks(FARPROC FAR *);


/*****************************************************************************\
*
*           Public Functions pointers
*
\*****************************************************************************/

/*****************************************************************************\
*
* Function:   RcGetFSError()
*
* Purpose:    return the most recent FS error code
*
```

```
* Method:   give value of last error that the file system encountered
*
* ASSUMES
*
*  globals IN:   rcFSError - current error code, set by most recent FS call
*
* PROMISES
*
*  returns:   returns current error in file system
*
\*****************************************************************************/

extern RC  (FAR PASCAL *RcGetFSError)(void);

/*****************************************************************************\
*
* Function:   HfsOpenSz( sz, bFlags )
*
* Purpose:   open a file system
*
* ASSUMES
*
*  args IN:   sz - path to file system to open
*             bFlags - fFSOpenReadOnly or fFSOpenReadWrite
*
* PROMISES
*
*  returns:   handle to file system if opened OK, else hNil
*
\*****************************************************************************/

extern HFS (FAR PASCAL *HfsOpenSz)( LPSTR, BYTE );


/*****************************************************************************\
*
* Function:   RcCloseHfs( hfs )
*
* Purpose:   Close an open file system
*       All files must be closed or changes made will be lost
*
* ASSUMES
*
*  args IN:   hfs - handle to an open file system
*
* PROMISES
*
*  returns:   standard return code
*
*  globals OUT:   rcFSError
*
\*****************************************************************************/

extern HFS (FAR PASCAL *RcCloseHfs)( HFS );


/*****************************************************************************\
*
* Function:   HfOpenHfs( hfs, sz, bFlags )
```

```
*
* Purpose:   open a file in a file system
*
* ASSUMES
*
*  args IN:   hfs - handle to file system
*             sz - name (key) of file to open
*             bFlags -
*
* PROMISES
*
*  returns:   handle to open file or hNil on failure
*
* Notes:   strlen( qNil ) and strcpy( s, qNil ) don't work as they should
*
\*****************************************************************************/

extern HF  (FAR PASCAL *HfOpenHfs    ) ( HFS,  LPSTR,  BYTE );

/*****************************************************************************\
*
* Function:   RcCloseHf( hf )
*
* Purpose:   close an open file in a file system
*
* Method:   If the file is dirty, copy the scratch file back to the FS file.
*             If this is the first time the file has been closed, enter the
*             name into the FS directory. If this file is the FS directory,
*             store the location in a special place instead. Write the FS
*             directory and header to disk. Do other various hairy stuff.
*
* ASSUMES
*
*  args IN:   hf - file handle
*
* PROMISES
*
*  returns:   rcSuccess on successful closing
*
\*****************************************************************************/

extern RC  (FAR PASCAL *RcCloseHf    ) ( HF        );

/*****************************************************************************\
*
* Function:   LcbReadHf()
*
* Purpose:   read bytes from a file in a file system
*
* ASSUMES
*
*  args IN:   hf - file
*             lcb - number of bytes to read
*
* PROMISES
*
*  returns:   number of bytes actually read; -1 on error
*
*  args OUT:   qb - data read from file goes here (must be big enough)
```

```
*
*  Notes:   These are signed longs we're dealing with. This means
*           behavior is different from read() when < 0.
*
\*****************************************************************************/

extern LONG (FAR PASCAL *LcbReadHf    ) ( HF,  LPSTR, LONG );

/*****************************************************************************\
*
* Function:   LcbWriteHf( hf, qb, lcb )
*
* Purpose:   write the contents of buffer into file
*
* Method:   If file isn't already dirty, copy data into temporary file.
*           Do the write.
*
* ASSUMES
*
*  args IN:   hf - file
*             qb - user's buffer full of stuff to write
*             lcb - number of bytes of qb to write
*
* PROMISES
*
*  returns:   number of bytes written if successful, -1L if not
*
*  args OUT:   hf - lifCurrent, lcbFile updated; dirty flag set
*
*  globals OUT:   rcFSError
*
\*****************************************************************************/

extern LONG (FAR PASCAL *LcbWriteHf    ) ( HF,  LPSTR, LONG );

/*****************************************************************************\
*
* Function:   LTellHf( hf )
*
* Purpose:   return current file position
*
* ASSUMES
*
*  args IN:   hf - handle to open file
*
* PROMISES
*
*  returns:   file position
*
\*****************************************************************************/

extern LONG (FAR PASCAL *LTellHf     ) ( HF        );

/*****************************************************************************\
*
* Function:   LSeekHf( hf, lOffset, wOrigin )
*
* Purpose:   set current file pointer
*
```

```
* ASSUMES
*
*  args IN:   hf   - file
*             lOffset - offset from origin
*             wOrigin - origin (wSeekSet, wSeekCur, or wSeekEnd)
*
* PROMISES
*
*  returns:   new position offset in bytes from beginning of file if
*             successful, or -1L if not
*
*  state OUT:   file pointer is set to new position unless error occurs,
*               in which case it stays where it was
*
\*************************************************************************/

extern LONG (FAR PASCAL *LSeekHf     ) ( HF,  LONG, WORD );

/*************************************************************************\
*
* Function:   FEofHf()
*
* Purpose:   tell whether file pointer is at end of file
*
* ASSUMES
*
*  args IN:   hf
*
* PROMISES
*
*  returns:   fTrue if file pointer is at EOF, fFalse otherwise
*
\*************************************************************************/

extern BOOL (FAR PASCAL *FEofHf      ) ( HF        );

/*************************************************************************\
*
* Function:   LcbSizeHf( hf )
*
* Purpose:   return the size in bytes of specified file
*
* ASSUMES
*
*  args IN:   hf - file handle
*
* PROMISES
*
*  returns:   size of the file in bytes
*
\*************************************************************************/

extern LONG (FAR PASCAL *LcbSizeHf    ) ( HF        );

/*************************************************************************\
*
* Function:   FAccessHfs( hfs, sz, bFlags )
*
* Purpose:   determine existence or legal access to an FS file
```

```
*
* ASSUMES
*
*  args IN:  hfs
*            sz  - file name
*            bFlags - ignored
*
* PROMISES
*
*  returns:  fTrue if file exists (is accessible in stated mode),
*            fFalse otherwise
*
* Bugs:  access mode part is unimplemented
*
\*************************************************************************/

extern BOOL (FAR PASCAL *FAccessHfs   ) ( HFS,  LPSTR, BYTE );

/*******************
 -
 - Name:  ErrorW
 *
 * Purpose:  displays an error message
 *
 * Arguments:  nError - string identifier. See wERRS_* messages.
 *
 * Returns:  nothing
 *
 *******************/

extern VOID (FAR PASCAL *ErrorW     ) ( int );

/*************************************************************************
 *
 - Name:  ErrorSz
 -
 * Purpose:  displays standard Windows Help error message dialog based on
 *           the string passed
 *
 * Arguments:  lpstr - string to display
 *
 * Returns:  nothing
 *
 *************************************************************************/

extern VOID (FAR PASCAL *ErrorSz    ) ( LPSTR );

/*************************************************************************
 *
 - Name:  LGetInfo
 -
 * Purpose:  gets global information from the application
 *
 * Arguments:  hwnd - window handle of topic to query.
 *             wItem - item to get
 *             GI_INSTANCE - application instance handle
 *             GI_MAINHWND - main window handle
 *             GI_CURRHWND - current window handle
 *             GI_HFS - handle to file system in use
```

```
 *                  GI_FGCOLOR - foreground color used by application
 *                  GI_BKCOLOR - background color used by application
 *                  GI_TOPICNO - topic number
 *                  GI_HPATH - handle containing path -- caller must free
 *
 *
 * Notes:   if the HWND is NULL, then the data will come from the window
 *          that currently has the focus
 *
 ****************************************************************************/


extern LONG (FAR PASCAL *LGetInfo    ) ( WORD, HWND    );

/*******************
 -
 - Name:   FAPI
 *
 * Purpose:   post a message for Help requests
 *
 * Arguments:   qchHelp - path (if not current directory) and file to use for Help topic
 *                  usCommand - command to send to Help
 *                  ulData - data associated with command
 *
 *
 * Returns:   TRUE iff success
 *
 * Notes:   see the Chapter 19 entry for the WinHelp API
 *
 ******************/

extern LONG (FAR PASCAL *FAPI      ) ( LPSTR, WORD, DWORD );

/*****************************************************************************\
*
- Function:   RcLLInfoFromHf( hf, wOption, qfid, qlBase, qlcb )
-
* Purpose:   map an HF into low-level file info
*
* ASSUMES
*  args IN:   hf - an open HF
*             qfid, qlBase, qlcb - pointers to user's variables
*             wOption - wLLSameFid, wLLDupFid, or wLLNewFid
*
* PROMISES
*  returns:   RcFSError(); rcSuccess on success
*
*  args OUT:   qfid - depending on value of wOption, either the same fid
*                     used by hf, a dup() of this fid, or a new fid obtained
*                     by reopening the file
*
*                qlBase - byte offset of first byte in the file
*                qlcb - size in bytes of the data in the file
*
*  globals OUT:   rcFSError
*
* Notes:   It is possible to read data outside the range specified by *qlBase
*          and *qlcb. Nothing is guaranteed about what will be found there.
*          If wOption is wLLSameFid or wLLDupFid, and the FS is opened in
```

```
*           opened in write mode, this fid will be writable. However, writing
*           is not allowed and may destroy the file system.
*
*           Fids obtained with the options wLLSameFid and wLLDupFid share
*           a file pointer with the hfs. This file pointer may change after any
*           operation on this FS. The fid obtained with the option wLLSameFid
*           may be closed by FS operations. If it is, the fid is invalid.
*
*           NULL can be passed for qfid, qlbase, qlcb and this routine will not
*           pass back the information.
*
* Bugs:   wLLDupFid is unimplemented
*
* +++
*
* Method:
*
* Notes:
*
\***************************************************************************/

extern RC  (FAR PASCAL *RcLLInfoFromHf  ) ( HF,     WORD, int FAR *, LONG FAR *, LONG
FAR * );

/***************************************************************************\
*
- Function:   RcLLInfoFromHfsSz( hfs, sz, wOption, qfid, qlBase, qlcb )
-
* Purpose:   map an HF into low-level file info
*
* ASSUMES
*  args IN:   hfs - an open HFS
*             szName - name of file in FS
*             qfid, qlBase, qlcb - pointers to user's variables
*             wOption - wLLSameFid, wLLDupFid, or wLLNewFid
*
* PROMISES
*  returns:   RcFSError(); rcSuccess on success
*
*  args OUT:   qfid  - depending on value of wOption, either the same fid
*                      used by hf, a dup() of this fid, or a new fid obtained by
*                      reopening the file
*
*              qlBase - byte offset of first byte in the file
*              qlcb - size in bytes of the data in the file
*
*  globals OUT:   rcFSError
*
* Notes:   It is possible to read data outside the range specified by *qlBase
*           and *qlcb. Nothing is guaranteed about what will be found. If wOption
*           is wLLSameFid or wLLDupFid, and the FS is opened in write mode,
*           this fid will be writable. However, writing is not allowed and may
*           destroy the file system.
*
*           Fids obtained with the options wLLSameFid and wLLDupFid share
*           a file pointer with the hfs. This file pointer may change after any
*           operation on this FS. The fid obtained with the option wLLSameFid
*           may be closed by FS operations. If it is, your fid is invalid.
*
```

```
*           NULL can be passed for qfid, qlbase, qlcb and this routine will not
*           pass back the information.
*
* Bugs:   wLLDupFid is unimplemented
*
* Method:   calls RcLLInfoFromHf()
*
* Notes:
*
\***************************************************************************/

extern RC  (FAR PASCAL *RcLLInfoFromHfs ) (HFS, LPSTR, WORD, int FAR *, LONG FAR *,
LONG FAR * );
```