



HsFtpWM FTP Client Library for Windows Mobile v1.2.0

User Manual

Revision: 1.5
Date: 20 February 2009

HSFTPWM FTP CLIENT LIBRARY FOR WINDOWS MOBILE V1.2.0 USER MANUAL	1
REVISION: 1.5	1
DATE: 20 FEBRUARY 2009	1
1 INTRODUCTION	3
2 HS FTP API	4
2.1 HSFTPINIT	4
2.2 HSFTPCLEANUP	4
2.3 HSFTPTICK	4
2.4 HSFTPCLICONNECT	5
2.4.1 DESCRIPTION OF HS_FTP_CONN_T STRUCTURE	5
2.5 HSFTPCLIDISCONNECT	6
2.6 HSFTPCLICHDIR	6
2.7 HSFTPCLICREATEDIR	6
2.8 HSFTPCLIREMOVEDIR	7
2.9 HSFTPCLILIST	7
2.10 HSFTPCLIGETFILE	7
2.11 HSFTPCLISENDFILE	8
2.12 HSFTPCLIDELETEFILE	9
2.13 HSFTPCLIABORT	9
2.14 HSFTPCLIRENAME	9
2.15 HSFTPCLIGETCURRENTDIRECTORY	10
2.16 HSFTPCLINOOP	10
2.17 HSFTPSETCONFIG	10
2.18 HSFTPGETSTATS	11
3 HS FTP CLIENT MODULE TO USER EVENT CALLBACK AND EVENTS	12
3.1 EVENT CALLBACK PROTOTYPE	12
3.2 EVENTS	12
3.3 INFORMATION CODES	15

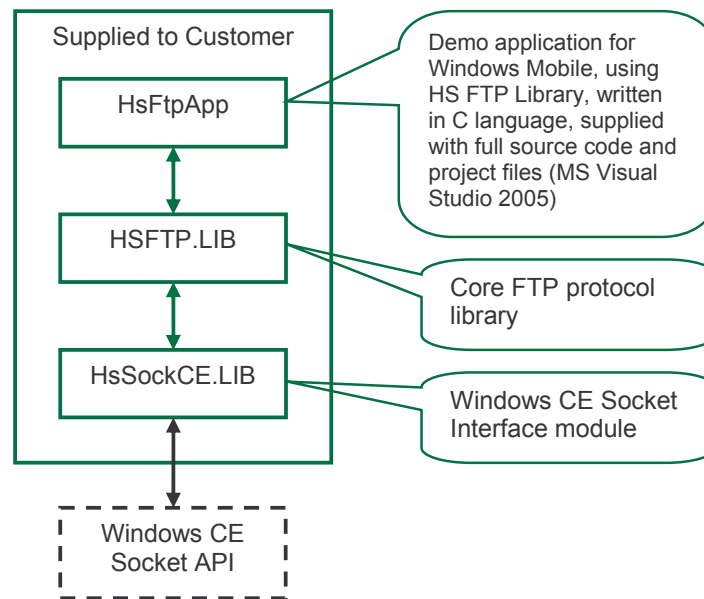
4	RECURSIVE FOLDER OPERATIONS	17
4.1	API FUNCTIONS	17
4.2	HSFTPRECURSINIT	17
4.3	HSFTPRECURSECLEANUP	17
4.4	HSFTPRECURSTICK	18
4.5	HSFTPRECURSDOWNLOADFOLDER	18
4.6	HSFTPRECURSUPLOADFOLDER	18
4.7	HSFTPRECURSDELETEFOLDER	19
4.8	RECURSIVE OPERATIONS CALLBACK AND EVENTS	19
4.8.1	EVENT CALLBACK PROTOTYPE	19
4.8.2	EVENTS	19
4.9	RECURSIVE OPERATIONS MODULE RETURN CODES	21

1 Introduction

HsFtpWM is a software library for Windows Mobile OS written in C language which implements the client side of the File Transfer Protocol over TCP socket layer according to RFC 959.

The library allows a user application running on Smartphone or Pocket PC to connect to remote FTP servers, traverse server directory structure, send and receive files.

HsFtpWM software architecture:



HsFtpWM incorporates the necessary server response processing and state machine required to comply with a simple implementation of FTP client.

The following FTP command sequences are supported:

- "USER" - authentication
- "PASS" - authentication
- "PASV" – establish passive mode data connection
- "ABOR" - abort
- "LIST" – request listing
- "CWD" – change directory
- "MKD" – create directory
- "RMD" – remove directory
- "TYPE" – set transfer type
- "RETR" – receive file
- "STOR" – transmit file
- "DELE" – delete file
- "NOOP" – no operation
- "PWD" – request current directory name
- "RNFR" – rename from
- "RNT0" – rename to

Additionally, HsFtpWM source code package (Blueprint Edition) contains "recursive folder operations" module (HsFtpRecurs) which implements:

- recursive folder download (download folder with all files and sub-folders)
- recursive folder upload (upload folder with all files and sub-folders)
- recursive folder delete (delete folder with all files and sub-folders)

2 HS FTP API

2.1 HsFtpInit

int HsFtpInit(void)		
DESCRIPTION		
<i>This function initialises HS FTP Library and MUST be called once, prior to calling any other library functions</i>		
PARAMETERS		
Type	Name	Description
None	None	None
RETURNS		
<ul style="list-style-type: none"> • <i>HS_FTPCLI_RC_ALRINIT – library is already initialised</i> • <i>HS_FTPCLI_RC_SOCKINIT_FAIL – socket layer initialisation failed</i> • <i>HS_FTP_RC_OK – success</i> 		

2.2 HsFtpCleanUp

int HsFtpCleanUp (void)		
DESCRIPTION		
<i>This function de-initialises HS FTP Library and releases resources used by it. Any active control and data connections shall be disconnected and all contexts and any used memory freed</i>		
PARAMETERS		
Type	Name	Description
None	None	None
RETURNS		
<ul style="list-style-type: none"> • <i>HS_FTPCLI_RC_NOTINIT – library not initialised</i> • <i>HS_FTP_RC_OK – success</i> 		

2.3 HsFtpTick

int HsFtpTick(void)		
DESCRIPTION		
<i>This function must be called as often as possible from the main program loop. This function drives internal timers and socket layer operations used by HS FTP module</i>		
PARAMETERS		
Type	Name	Description
None	None	None
RETURNS		
<ul style="list-style-type: none"> • <i>HS_FTPCLI_RC_NOTINIT – library not initialised</i> • <i>HS_FTP_RC_OK – success</i> 		

2.4 HsFtpCliConnect

```
extern int HsFtpCliConnect(hs_ftp_conn_t *conn, long *session_handle)
```

DESCRIPTION

This function is used to connect to remote FTP server

PARAMETERS

Type	Name	Description
hs_ftp_conn_t	*conn	Pointer to structure in user code which contains connection parameters. Please see the next section below for detailed description of data members.
long	*session_handle	Pointer to long variable in user code to receive HS FTP module handle associated with this FTP session. This handle must then be used in all further calls to HS FTP module related to this FTP session.

RETURNS

- *HS_FTPCLI_RC_NOTINIT – library not initialised*
- *HS_FTPCLI_RC_INV_PAR – invalid parameters supplied*
- *HS_FTPCLI_RC_NO_CTX – there are no more free HS FTP Client contexts, maximum number of concurrent connections reached.*
- *HS_FTPCLI_RC_TCPCONNFAIL – outgoing TCP connection to remote FTP server failed*
- *HS_FTP_RC_OK – success, HS FTP module started session establishment to remote FTP server. The actual result shall be asynchronously indicated via user callback event, part of hs_ftp_conn_t structure, see description in the following section.*

2.4.1 Description of hs_ftp_conn_t structure

Data Member	Description
unsigned char *srv_name;	Remote FTP server DNS name to connect to, for example (ftp.hillstone-software.com). This can also be an IP address string in dotted format, for example "192.168.1.2". This parameter is a pointer to null terminated string.
unsigned short srv_port;	Remote FTP server port to connect to
unsigned char *username;	FTP account user name for authentication, pointer to null terminated string
unsigned char *password;	FTP account password for authentication, pointer to null terminated string
ftp_callback_t *callback;	Pointer to callback function used by HS FTP to communicate to user application. Please see the detailed description in section 3 (HS FTP Client Module to USER Event Callback and Events)
void *user_ref;	User data. The user application can store any value (or pointer) here. HS FTP module shall always pass this value back unmodified to user event callback function

2.5 HsFtpCliDisconnect

extern int HsFtpCliDisconnect(long session_handle)

DESCRIPTION

This function is used to close FTP session to remote FTP server

PARAMETERS

Type	Name	Description
long	session_handle	HS FTP session handle, returned in HsFtpCliConnect call.

RETURNS

- *HS_FTPCLI_RC_NOTINIT – library not initialised*
- *HS_FTPCLI_RC_INV_PAR – invalid parameters supplied*
- *HS_FTPCLI_RC_NOTALLOC – invalid session – not allocated*
- *HS_FTP_RC_OK – success, session disconnected*

2.6 HsFtpCliChDir

extern int HsFtpCliChDir(long session_handle, unsigned char *dir)

DESCRIPTION

This function is used to change to a different directory on remote FTP server

PARAMETERS

Type	Name	Description
long	session_handle	HS FTP session handle, returned in HsFtpCliConnect call.
unsigned char	*dir	Directory name to change to, null terminated string

RETURNS

- *HS_FTPCLI_RC_NOTINIT – library not initialised*
- *HS_FTPCLI_RC_INV_PAR – invalid parameters supplied*
- *HS_FTPCLI_RC_NOTALLOC – invalid session – not allocated*
- *HS_FTPCLI_RC_INVALID_CMD – command is invalid for current session state*
- *HS_FTP_RC_OK – success, HS FTP initiated procedure to change to a new directory on remote FTP server. The actual result shall be asynchronously indicated via user callback event, please see the detailed description in section 3 (HS FTP Client Module to USER Event Callback and Events)*

2.7 HsFtpCliCreateDir

extern int HsFtpCliCreateDir(long session_handle, unsigned char *dir)

DESCRIPTION

This function is used to create a new directory on remote FTP server

PARAMETERS

Type	Name	Description
long	session_handle	HS FTP session handle, returned in HsFtpCliConnect call.
unsigned char	*dir	Directory name, null terminated string

RETURNS

- *HS_FTPCLI_RC_NOTINIT – library not initialised*
- *HS_FTPCLI_RC_INV_PAR – invalid parameters supplied*
- *HS_FTPCLI_RC_NOTALLOC – invalid session – not allocated*

- *HS_FTPCLI_RC_INVALID_CMD – command is invalid for current session state*
- *HS_FTP_RC_OK – success, HS FTP initiated procedure to create a new directory on remote FTP server. The actual result shall be asynchronously indicated via user callback event, please see the detailed description in section 3 (HS FTP Client Module to USER Event Callback and Events)*

2.8 HsFtpCliRemoveDir

extern int HsFtpCliRemoveDir(long session_handle, unsigned char *dir)

DESCRIPTION

This function is used to remove a directory on remote FTP server

PARAMETERS

Type	Name	Description
long	session_handle	HS FTP session handle, returned in HsFtpCliConnect call.
unsigned char	*dir	Directory name, null terminated string

RETURNS

- *HS_FTPCLI_RC_NOTINIT – library not initialised*
- *HS_FTPCLI_RC_INV_PAR – invalid parameters supplied*
- *HS_FTPCLI_RC_NOTALLOC – invalid session – not allocated*
- *HS_FTPCLI_RC_INVALID_CMD – command is invalid for current session state*
- *HS_FTP_RC_OK – success, HS FTP initiated procedure to remove a directory on remote FTP server. The actual result shall be asynchronously indicated via user callback event, please see the detailed description in section 3 (HS FTP Client Module to USER Event Callback and Events)*

2.9 HsFtpCliList

extern int HsFtpCliList(long session_handle);

DESCRIPTION

This function is used to receive directory file listing from the remote FTP server. The listing is returned asynchronously via user event callback – event HS_FTPCLI_USR_EV_LIST, , please see the detailed description in section 3 (HS FTP Client Module to USER Event Callback and Events). The listing is returned in the same format as it came in from the server, the actual format depends on server OS and FTP software. Please see the example of parsing the listing (breaking down into filenames) supplied in HS FTP Demo application, function load_remote_directory.

PARAMETERS

Type	Name	Description
long	session_handle	HS FTP session handle, returned in HsFtpCliConnect call.

RETURNS

- *HS_FTPCLI_RC_NOTINIT – library not initialised*
- *HS_FTPCLI_RC_INV_PAR – invalid parameters supplied*
- *HS_FTPCLI_RC_NOTALLOC – invalid session – not allocated*
- *HS_FTP_RC_OK – success, HS FTP library initiated a procedure to receive directory listing from remote FTP server. The actual result shall be asynchronously indicated via user callback event, please see the detailed description in section 3 (HS FTP Client Module to USER Event Callback and Events)*

2.10 HsFtpCliGetFile

extern int HsFtpCliGetFile(long session_handle, unsigned char *filename, __int64 fszize)

DESCRIPTION

This function is used to transfer the specified file from remote FTP server to local system.

Successful return of this function indicates that HS FTP has started file reception protocol sequence.

HS FTP handles all aspects of receiving the file (opening disk file, receiving and writing blocks of data and closing file). As the file data gets transferred block by block, events `HS_FTPCLI_USR_EV_RX_STATUS` are generated, informing the application that the next block of data (specifying its size) has been written from to file.

Successful completion of file transfer (when file has been completely received and disk file closed) is reported to application via `HS_FTPCLI_USR_EV_RXDONE` event

PARAMETERS

Type	Name	Description
long	session_handle	HS FTP session handle, returned in HsFtpCliConnect call.
unsigned char	*filename	Filename to get, pointer to null terminated string
__int64	fsize	Size of file to get, 64 bit integer. File size shall be obtained first after successful call to HsFtpCliList

RETURNS

- `HS_FTPCLI_RC_NOTINIT` – library not initialised
- `HS_FTPCLI_RC_INV_PAR` – invalid parameters supplied
- `HS_FTPCLI_RC_NOTALLOC` – invalid session – not allocated
- `HS_FTPCLI_RC_INVALID_CMD` – command is invalid for current session state
- `HS_FTP_RC_OK` – success, HS FTP initiated procedure to start downloading the specified file from remote FTP server. The actual result shall be asynchronously indicated via user callback event, please see the detailed description in section 3 (HS FTP Client Module to USER Event Callback and Events)

2.11 HsFtpCliSendFile

extern int HsFtpCliSendFile (long session_handle, unsigned char *filename)

DESCRIPTION

This function is used to transfer the specified file from local system to remote FTP server.

Successful return of this function indicates that HS FTP has started file sending protocol sequence.

HS FTP handles all aspects of sending the file (opening disk file, reading blocks of data and sending them, closing file). As the file data gets transferred block by block, events `HS_FTPCLI_USR_EV_TX_STATUS` are generated, informing the application that the next block of data (specifying its size) has been read from disk file and sent to FTP server

Successful completion of file transfer (when file has been completely sent and disk file closed) is reported to application via `HS_FTPCLI_USR_EV_TXDONE` event

PARAMETERS

Type	Name	Description
long	session_handle	HS FTP session handle, returned in HsFtpCliConnect call.
unsigned char	*filename	Filename to get, pointer to null terminated string

RETURNS

- `HS_FTPCLI_RC_NOTINIT` – library not initialised
- `HS_FTPCLI_RC_INV_PAR` – invalid parameters supplied
- `HS_FTPCLI_RC_NOTALLOC` – invalid session – not allocated
- `HS_FTPCLI_RC_INVALID_CMD` – command is invalid for current session state
- `HS_FTP_RC_OK` – success, HS FTP initiated procedure to start uploading the specified file to remote FTP server. The completion of file transfer shall be asynchronously indicated via user callback event, please see the detailed description in section 3 (HS FTP Client Module to USER Event Callback and Events)

2.12 HsFtpCliDeleteFile

extern int HsFtpCliDeleteFile(long session_handle, unsigned char *filename)		
DESCRIPTION		
<i>This function is used to delete a file on remote FTP server</i>		
PARAMETERS		
Type	Name	Description
long	session_handle	HS FTP session handle, returned in HsFtpCliConnect call.
unsigned char	*filename	File name, null terminated string
RETURNS		
<ul style="list-style-type: none"> • <i>HS_FTPCLI_RC_NOTINIT – library not initialised</i> • <i>HS_FTPCLI_RC_INV_PAR – invalid parameters supplied</i> • <i>HS_FTPCLI_RC_NOTALLOC – invalid session – not allocated</i> • <i>HS_FTPCLI_RC_INVALID_CMD – command is invalid for current session state</i> • <i>HS_FTP_RC_OK – success, HS FTP initiated procedure to delete a file on remote FTP server. The actual result shall be asynchronously indicated via user callback event, please see the detailed description in section 3 (HS FTP Client Module to USER Event Callback and Events)</i> 		

2.13 HsFtpCliAbort

extern int HsFtpCliAbort (long session_handle)		
DESCRIPTION		
<i>This function is used to abort current FTP operation in progress</i>		
PARAMETERS		
Type	Name	Description
long	session_handle	HS FTP session handle, returned in HsFtpCliConnect call.
RETURNS		
<ul style="list-style-type: none"> • <i>HS_FTPCLI_RC_NOTINIT – library not initialised</i> • <i>HS_FTPCLI_RC_INV_PAR – invalid parameters supplied</i> • <i>HS_FTPCLI_RC_NOTALLOC – invalid session – not allocated</i> • <i>HS_FTP_RC_OK – success, success, HS FTP initiated procedure to abort current command. The completion shall be asynchronously indicated via user callback event, please see the detailed description in section 3 (HS FTP Client Module to USER Event Callback and Events)</i> 		

2.14 HsFtpCliRename

extern int HsFtpCliRename(long session_handle, unsigned char *oldname, unsigned char *newname)		
DESCRIPTION		
<i>This function is used to rename a file or folder on remote FTP server</i>		
PARAMETERS		
Type	Name	Description
long	session_handle	HS FTP session handle, returned in HsFtpCliConnect call.
unsigned char	*oldname	File or folder name to rename, null terminated string
unsigned char	*newname	New name for file or folder name, null terminated string
RETURNS		

- *HS_FTPCLI_RC_NOTINIT – library not initialised*
- *HS_FTPCLI_RC_INV_PAR – invalid parameters supplied*
- *HS_FTPCLI_RC_NOTALLOC – invalid session – not allocated*
- *HS_FTPCLI_RC_INVALID_CMD – command is invalid for current session state*
- *HS_FTP_RC_OK – success, HS FTP initiated procedure to rename a file or folder on remote FTP server. The actual result shall be asynchronously indicated via user callback event, please see the detailed description in section 3 (HS FTP Client Module to USER Event Callback and Events)*

2.15 HsFtpCliGetCurrentDirectory

extern int HsFtpCliGetCurrentDirectory(long session_handle);		
DESCRIPTION		
<i>This function is used to request the current working directory name from remote FTP server</i>		
PARAMETERS		
Type	Name	Description
long	session_handle	HS FTP session handle, returned in HsFtpCliConnect call.
RETURNS		
<ul style="list-style-type: none"> • <i>HS_FTPCLI_RC_NOTINIT – library not initialised</i> • <i>HS_FTPCLI_RC_INV_PAR – invalid parameters supplied</i> • <i>HS_FTPCLI_RC_NOTALLOC – invalid session – not allocated</i> • <i>HS_FTPCLI_RC_INVALID_CMD – command is invalid for current session state</i> • <i>HS_FTP_RC_OK – success, HS FTP initiated procedure to request the current working directory from remote FTP server. The actual result shall be asynchronously indicated via user callback event, please see the detailed description in section 3 (HS FTP Client Module to USER Event Callback and Events)</i> 		

2.16 HsFtpCliNoop

extern int HsFtpCliNoop(long session_handle);		
DESCRIPTION		
<i>This function is used to send NOOP command to remote FTP server. This command does not require the server to execute any action except send a valid response</i>		
PARAMETERS		
Type	Name	Description
long	session_handle	HS FTP session handle, returned in HsFtpCliConnect call.
RETURNS		
<ul style="list-style-type: none"> • <i>HS_FTPCLI_RC_NOTINIT – library not initialised</i> • <i>HS_FTPCLI_RC_INV_PAR – invalid parameters supplied</i> • <i>HS_FTPCLI_RC_NOTALLOC – invalid session – not allocated</i> • <i>HS_FTPCLI_RC_INVALID_CMD – command is invalid for current session state</i> • <i>HS_FTP_RC_OK – success, HS FTP initiated procedure to send NOOP command to remote FTP server. The actual result shall be asynchronously indicated via user callback event, please see the detailed description in section 3 (HS FTP Client Module to USER Event Callback and Events)</i> 		

2.17 HsFtpSetConfig

extern void HsFtpSetConfig (hs_ftp_config_t *cfg);		
DESCRIPTION		
<i>This function is used to set HS FTP global configuration parameters</i>		

PARAMETERS		
Type	Name	Description
hs_ftp_config_t	*cfg	Pointer to configuration structure hs_ftp_config_t, with the following data members: long t1_timeout – timeout in milliseconds for all HS FTP operations. Default timeout is 15000 milliseconds
RETURNS		
NONE		

2.18 HsFtpGetStats

extern void HsFtpGetStats (hsftp_stats_t *pStats);		
DESCRIPTION		
<i>This function is used to set HS FTP global configuration parameters</i>		
PARAMETERS		
Type	Name	Description
hsftp_stats_t	*pStats	Pointer to stats structure with the following data members: <pre> __int64 total_bytes_Tx; // total number of bytes transmitted __int64 total_bytes_Rx; // total number of bytes received </pre> These counters include both control and data FTP connections
RETURNS		
NONE		

3 HS FTP Client Module to USER Event Callback and Events

3.1 Event Callback Prototype

```
typedef int ftp_callback_t(void *user_ref, int ev, long arg1, long arg2, long arg3);
```

Parameter	Description
void *user_ref	User data. The user application passes any value (or pointer) in the call to HsFtpCliConnect. HS FTP module always pass this value back unmodified to this parameter, which can be used by user code to identify FTP session contexts.
int ev	Event id, the full list and description is provided in section 3.2 Events.
long arg1	Parameter 1, specific to event id
long arg2	Parameter 2, specific to event id
long arg3	Parameter 3, specific to event id

Returns:

True or False. For most events the return is insignificant and is not checked by HS FTP Client. Where HS FTP needs a specific return, it is clearly specified in this document.

3.2 Events

Event	Description
HS_FTPCLI_USR_EV_LOGGEDIN	FTP session to remote FTP server established, login successful. The user application can now call functions to get directory listing, change directory, get and send files. Arg1 – pointer to buffer containing ftp server reply string (for example “226 logged in”) Arg2 – length of buffer containing ftp server reply string Arg3 - 0
HS_FTPCLI_USR_EV_CLOSED	FTP control connection and FTP session closed (error condition). Arg1 – 0 Arg2 – 0 Arg3 – HS FTP library integer debug code – indicates from which state the callback function was called. Refer to section 3.3 for full list of codes.
HS_FTPCLI_USR_EV_CONNFAIL	Outgoing TCP connection to remote FTP server failed to establish. Arg1 – pointer to null terminated additional information string Arg2 – 0 Arg3 – 0
HS_FTPCLI_USR_EV_SRVERR	FTP session closed due to server error reply. Arg1 – pointer to buffer containing ftp server reply string Arg2 – length of buffer containing ftp server reply string Arg3 – HS FTP library integer debug code – indicates from which state the callback function was called. Refer to section 3.3 for full list of codes.
HS_FTPCLI_USR_EV_UNEXP	Unexpected server response, session closed. Arg1 – pointer to buffer containing ftp server reply string Arg2 – length of buffer containing ftp server reply string Arg3 – HS FTP library integer debug code – indicates from which state the callback function was called. Refer to section 3.3 for full list of codes.
HS_FTPCLI_USR_EV_UNKNOWN	Unrecognized server response, session closed.

	<p>Arg1 – pointer to buffer containing ftp server reply string Arg2 – length of buffer containing ftp server reply string Arg3 – HS FTP library integer debug code – indicates from which state the callback function was called. Refer to section 3.3 for full list of codes</p>
HS_FTPCLI_USR_EV_NOCTX	<p>FTP session closed, HS FTP library run out of free concurrent connection contexts</p> <p>Arg1 – 0 Arg2 – 0 Arg3 – 0</p>
HS_FTPCLI_USR_EV_TIMEDOUT	<p>FTP session closed due to timeout</p> <p>Arg1 – pointer to null terminated sting with additional information Arg2 – 0 Arg3 – 0</p>
HS_FTPCLI_USR_EV_NOMEM	<p>FTP session closed, no free memory</p> <p>Arg1 – 0 Arg2 – 0 Arg3 – HS FTP library integer debug code – indicates from which state the callback function was called. Refer to section 3.3 for full list of codes</p>
HS_FTPCLI_USR_EV_LIST	<p>This event is used by HS FTP module to return a buffer containing current directory listing from remote FTP server after HsFtpCliList function call</p> <p>Arg1 – pointer to start of buffer containing remote FTP server current directory listing. Arg2 – length of buffer Arg3 – 0</p> <p>User application must copy buffer content to local memory. On return from the callback the buffer memory is deallocated by HS FTP module.</p>
HS_FTPCLI_USR_EV_CWD_FAILED	<p>HsFtpCliChDir function failed (CWD FTP operation failed).</p> <p>Arg1 – pointer to buffer containing ftp server reply string Arg2 – length of buffer containing ftp server reply string Arg3 – 0</p>
HS_FTPCLI_USR_EV_CWD_DONE	<p>HsFtpCliChDir function successfully completed – Current directory on remote FTP server successfully changed to new specified directory.</p> <p>Arg1 – pointer to buffer containing ftp server reply string Arg2 – length of buffer containing ftp server reply string Arg3 – 0</p>
HS_FTPCLI_USR_EV_RXDONE	<p>File successfully received – HsFtpCliGetFile function completed.</p> <p>Arg1 – pointer to buffer containing ftp server reply string Arg2 – length of buffer containing ftp server reply string Arg3 – 0</p>
HS_FTPCLI_USR_EV_TXDONE	<p>File successfully transmitted – HsFtpCliSendFile completed.</p> <p>Arg1 – pointer to buffer containing ftp server reply string Arg2 – length of buffer containing ftp server reply string Arg3 – 0</p>

HS_FTPCLI_USR_EV_RXINCOMPL	File receive operation failed. Partial file received. Arg1 – 0 Arg2 – 0 Arg3 – 0
HS_FTPCLI_USR_EV_TX_STATUS	Notifies the application that the next block of file data has been transmitted Arg3 – size of data block that has been transmitted
HS_FTPCLI_USR_EV_RX_STATUS	Notifies the application that the next data block of file from remote FTP server has been received Arg3 – size of data block that has been received
HS_FTPCLI_USR_EV_TYP_FAILED	Setting transfer type failed (TYPE command failed) Arg1 – pointer to buffer containing ftp server reply string Arg2 – length of buffer containing ftp server reply string Arg3 – HS FTP library integer debug code – indicates from which state the callback function was called. Refer to section 3.3 for full list of codes
HS_FTPCLI_USR_EV_STOR_FAILED	File transmit operation failed – HsFtpCliSendFile function completed with error. Arg1 – 0 Arg2 – 0 Arg3 – HS FTP library integer debug code – indicates from which state the callback function was called. Refer to section 3.3 for full list of codes
HS_FTPCLI_USR_EV_ABORTED	Current FTP operation aborted – HsFtpCliAbort function completed Arg1 – pointer to buffer containing ftp server reply string Arg2 – length of buffer containing ftp server reply string Arg3 – HS FTP library integer debug code – indicates from which state the callback function was called. Refer to section 3.3 for full list of codes
HS_FTPCLI_USR_EV_ABORT_FAILED	Abort operation failed - HsFtpCliAbort function completed with error Arg1 – pointer to buffer containing ftp server reply string Arg2 – length of buffer containing ftp server reply string Arg3 – HS FTP library integer debug code – indicates from which state the callback function was called. Refer to section 3.3 for full list of codes
HS_FTPCLI_USR_EV_MKD_FAILED	HsFtpCliCreateDir function failed (MKD FTP operation failed). Arg1 – pointer to buffer containing ftp server reply string Arg2 – length of buffer containing ftp server reply string Arg3 – 0
HS_FTPCLI_USR_EV_MKD_DONE	HsFtpCliCreateDir function successfully completed – Directory on remote FTP server successfully created. Arg1 – pointer to buffer containing ftp server reply string Arg2 – length of buffer containing ftp server reply string Arg3 – 0
HS_FTPCLI_USR_EV_RMD_FAILED	HsFtpCliRemoveDir function failed (RMD FTP operation failed). Arg1 – pointer to buffer containing ftp server reply string Arg2 – length of buffer containing ftp server reply string Arg3 – 0

HS_FTPCLI_USR_EV_RMD_DONE	HsFtpCliRemoveDir function successfully completed – Directory on remote FTP server successfully deleted Arg1 – pointer to buffer containing ftp server reply string Arg2 – length of buffer containing ftp server reply string Arg3 – 0
HS_FTPCLI_USR_EV_DELE_DONE	HsFtpCliDeleteFile function successfully completed – File on remote FTP server successfully deleted Arg1 – pointer to buffer containing ftp server reply string Arg2 – length of buffer containing ftp server reply string Arg3 – 0
HS_FTPCLI_USR_EV_DELE_FAILED	HsFtpCliDeleteFile function failed (DELE FTP operation failed). Arg1 – pointer to buffer containing ftp server reply string Arg2 – length of buffer containing ftp server reply string Arg3 – 0
HS_FTPCLI_USR_EV_RENAME_DONE	HsFtpCliRename function successfully completed file or folder on remote FTP server renamed Arg1 – pointer to buffer containing ftp server reply string Arg2 – length of buffer containing ftp server reply string Arg3 – 0
HS_FTPCLI_USR_EV_RENAME_FAILED	HsFtpCliRename function failed (RNFR FTP command failed) Arg1 – pointer to buffer containing ftp server reply string Arg2 – length of buffer containing ftp server reply string Arg3 – 0
HS_FTPCLI_USR_EV_PWD_DONE	HsFtpCliGetCurrentDirectory function completed successfully. Arg1 – pointer to buffer containing ftp server reply string Arg2 – length of buffer containing ftp server reply string Arg3 – 0
HS_FTPCLI_USR_EV_PWD_FAILED	HsFtpCliGetCurrentDirectory function failed (FTP PWD command failed). Arg1 – pointer to buffer containing ftp server reply string Arg2 – length of buffer containing ftp server reply string Arg3 – 0
HS_FTPCLI_USR_EV_NOOP_DONE	Response to NOOP command received Arg1 – pointer to buffer containing ftp server reply string Arg2 – length of buffer containing ftp server reply string Arg3 – 0

3.3 Information Codes

HS_FTPEC_ST_WAIT_WELCOME
HS_FTPEC_ST_CHDIR_RSP
HS_FTPEC_ST_LIST_WAIT_RSP
HS_FTPEC_ST_WAIT_PASV_RSP
HS_FTPEC_ST_WAIT_USER_RSP
HS_FTPEC_ST_WAIT_PASS_RSP

- waiting for initial server reply on control session setup
- waiting for reply to CWD command
- waiting for reply to LIST command
- waiting for reply to PASV command
- waiting for reply to USER command
- waiting for reply to PASS command

HS_FTPC_ST_WAIT_LSTDATA_CONN	- waiting for data connection (LIST command)
HS_FTPC_ST_WAIT_CWDDATA_CONN	- waiting for data connection (CWD command)
HS_FTPC_ST_WAIT_TYPE_RSP	- waiting for reply to TYPE command
HS_FTPC_ST_WAIT_RETRDATA_CONN	- waiting for data connection (RETR command)
HS_FTPC_ST_WAIT_STORDATA_CONN	- waiting for data connection (STOR command)
HS_FTPC_ST_STOR_WAIT_RSP	- STOR wait response state
HS_FTPC_ST_RETR_WAIT_RSP	- RETR wait response state
HS_FTPC_ST_STOR_SENDING	- STOR sending file data state
HS_FTPC_ST_WAIT_ABOR_RSP	- waiting for reply to ABOR command
HS_FTPC_ST_LIST_WAIT_DTCLOSE	- waiting for data connection to close
HS_FTPC_ST_MKDIR_RSP	- waiting for response to MKD
HS_FTPC_ST_RMDIR_RSP	- waiting for response to RMD
HS_FTPC_ST_DELE_RSP	- waiting for response to DELE
HS_FTPC_ST_WAIT_MKDDATA_CONN	- waiting for data connection (MKD)
HS_FTPC_ST_WAIT_RMDDATA_CONN	- waiting for data connection (RMD)
HS_FTPC_ST_WAIT_DELEDATA_CONN	- waiting for data connection (DELE)
HS_FTPC_ST_WAIT_NOOP_RSP	- waiting for response to NOOP
HS_FTPC_ST_WAIT_PWD_RSP	- waiting for response to PWD
HS_FTPC_ST_WAIT_RNFR_RSP	- waiting for response to RNFR
HS_FTPC_ST_WAIT_RNTO_RSP	- waiting for response to RNTO
HS_FTPC_ST_LIST_WAIT_DATA_CLOSED	- waiting for data connection closed in LIST sequence
HS_FTPC_ST_LOGGED_IN	- logged_in state

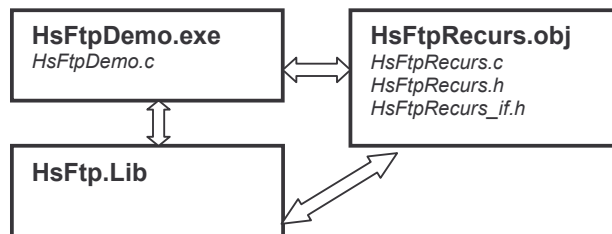
4 Recursive Folder Operations

HS FTP Source Code Library package includes recursive folder operations module HsFtpRecurs which includes the following functions:

- HsFtpRecursDownloadFolder – recursively download entire folder with all sub-folders and files from remote FTP server.
- HsFtpRecursUploadFolder – recursively upload entire folder with all sub-folders and files to remote FTP server.
- HsFtpRecursDeleteFolder – recursively delete entire folder with all sub-folders and files from remote FTP server.

Recursive operations module is currently only supplied to customers with HS FTP Library Blueprint Edition and HsFtpWM Blueprint Edition.

Recursive operations module is not part of the code HSFTP library, but instead is implemented as an object module that links to the main application which in turn links in the code HSFTP.LIB



4.1 API Functions

4.2 HsFtpRecursInit

Extern void HsFtpRecursInit (hsftp_recursive_callback_t *cb)		
DESCRIPTION		
<i>This function is used to initialise recursive folder operations module, it must be used before any other functions of this module are used</i>		
PARAMETERS		
Type	Name	Description
hsftp_recursive_callback_t	*cb	Function pointer to recursive module callback functions which receives event notifications of folder operations completion, failures, status and progress. See section 4.7 for definition of the callback function
RETURNS		
NONE		

4.3 HsFtpRecurseCleanUp

Extern void HsFtpRecurseCleanUp (void)		
DESCRIPTION		
<i>This function is used to release all resources used by recursive folder operations module</i>		
PARAMETERS		
NONE		
RETURNS		
NONE		

4.4 HsFtpRecursTick

void HsFtpTick(void)		
DESCRIPTION		
<i>This function must be called as often as possible from the main program loop.</i>		
PARAMETERS		
Type	Name	Description
None	None	None
RETURNS		
NONE		

4.5 HsFtpRecursDownloadFolder

extern int HsFtpRecursDownloadFolder (long session_handle, unsigned char *foldername, int overwrite)		
DESCRIPTION		
<i>This function is used to download an entire folder with all sub-folders and files from remote FTP server</i>		
PARAMETERS		
Type	Name	Description
long	session_handle	HS FTP session handle, returned in HsFtpCliConnect call.
unsigned char	*foldername	Folder name to download, null terminated string
int	overwrite	1 = Any files already existing at local file system shall be overwritten 0 = Files already existing at local files system shall be overwritten only if the files size does not match the files size of the corresponding file at the remote FTP server, otherwise if the file sizes match, the file is not downloaded
RETURNS		
<ul style="list-style-type: none"> • <i>HSFTP_RECURS_RC_NOTINIT – Recursive folder operations module not initialised</i> • <i>HSFTP_RECURS_RC_INVPAR – invalid parameters supplied</i> • <i>HSFTP_RECURS_RC_BUSY – command is invalid for current session state</i> • <i>HSFTP_RECURS_RC_OK – success, Recursive folder download started. The actual result shall be asynchronously indicated via user callback event, please see the detailed description in section 4.8</i> 		

4.6 HsFtpRecursUploadFolder

extern int HsFtpRecursUploadFolder (long session_handle, unsigned char *foldername)		
DESCRIPTION		
<i>This function is used to upload an entire folder with all sub-folders and files to remote FTP server</i>		
PARAMETERS		
Type	Name	Description
Long	session_handle	HS FTP session handle, returned in HsFtpCliConnect call.
unsigned char	*foldername	Folder name to upload, null terminated string
RETURNS		
<ul style="list-style-type: none"> • <i>HSFTP_RECURS_RC_NOTINIT – Recursive folder operations module not initialised</i> • <i>HSFTP_RECURS_RC_INVPAR – invalid parameters supplied</i> • <i>HSFTP_RECURS_RC_BUSY – command is invalid for current session state</i> 		

- *HSFTP_RECURS_RC_LCHDIR – failed to change into local directory (local chdir failed)*
- *HSFTP_RECURS_RC_OK – success, Recursive folder upload started. The actual result shall be asynchronously indicated via user callback event, please see the detailed description in section 4.8*

4.7 HsFtpRecursDeleteFolder

extern int HsFtpRecursUploadFolder (long session_handle, unsigned char *foldername)		
DESCRIPTION		
<i>This function is used to delete an entire folder with all sub-folders and files from remote FTP server</i>		
PARAMETERS		
Type	Name	Description
long	session_handle	HS FTP session handle, returned in HsFtpCliConnect call.
unsigned char	*foldername	Folder name to delete, null terminated string
RETURNS		
<ul style="list-style-type: none"> • <i>HSFTP_RECURS_RC_NOTINIT – Recursive folder operations module not initialised</i> • <i>HSFTP_RECURS_RC_INVPAR – invalid parameters supplied</i> • <i>HSFTP_RECURS_RC_BUSY – command is invalid for current session state</i> • <i>HSFTP_RECURS_RC_OK – success, Recursive folder deletion started. The actual result shall be asynchronously indicated via user callback event, please see the detailed description in section 4.8</i> 		

4.8 Recursive Operations Callback and Events

4.8.1 Event Callback Prototype

typedef void hsftp_recursive_callback_t (long ftp_session, int ev, long arg1, long arg2);

Parameter	Description
long ftp_session	FTP session handle.
int ev	Event id, the full list and description is provided in next section
long arg1	Parameter 1, specific to event id
long arg2	Parameter 2, specific to event id

4.8.2 Events

Event	Description
HSFTP_RECURS_USR_EV_FOLDER_DOWNLOAD_FAILED	<p>Recursive folder download operation failed.</p> <p>Arg1 – Recursive operations module return code, see next section for a list of return codes</p> <p>Arg2 – Core HS FTP library event that caused this event, see section 3.2 for a list of core HS FTP events</p>
HSFTP_RECURS_USR_EV_FOLDER_DOWNLOAD_DONE	<p>Recursive folder download operation complete with success.</p> <p>Arg1 – 0</p> <p>Arg2 – 0</p>
HSFTP_RECURS_USR_EV_FOLDER_DOWNLOAD_STATUS	<p>reports start of stop of individual file download operation</p> <p>Arg1 – If Arg2 == 1, long pointer to a pathname of the item now starting to download</p>

	<p>Arg2 – status code: 1 = Start of individual item download 2 = Completion of individual item download</p>
HSFTP_RECURS_USR_EV_FOLDER_UPLOAD_FAILED	<p>Recursive folder upload operation failed.</p> <p>Arg1 – Recursive operations module return code, see next section for a list of return codes</p> <p>Arg2 – Core HS FTP library event that caused this event, see section 3.2 for a list of core HS FTP events</p>
HSFTP_RECURS_USR_EV_FOLDER_UPLOAD_DONE	<p>Recursive folder upload operation complete with success.</p> <p>Arg1 – 0 Arg2 – 0</p>
HSFTP_RECURS_USR_EV_FOLDER_UPLOAD_STATUS	<p>reports start of stop of individual file upload operation</p> <p>Arg1 – If Arg2 == 1, long pointer to a pathname of the item now starting to upload</p> <p>Arg2 – status code: 1 = Start of individual item upload 2 = Completion of individual item upload</p>
HSFTP_RECURS_USR_EV_FOLDER_DELETE_FAILED	<p>Recursive folder delete operation failed.</p> <p>Arg1 – Recursive operations module return code, see next section for a list of return codes</p> <p>Arg2 – Core HS FTP library event that caused this event, see section 3.2 for a list of core HS FTP events</p>
HSFTP_RECURS_USR_EV_FOLDER_DELETE_DONE	<p>Recursive folder delete operation complete with success.</p> <p>Arg1 – 0 Arg2 – 0</p>
HSFTP_RECURS_USR_EV_FOLDER_DELETE_STATUS	<p>reports start of stop of individual item delete operation</p> <p>Arg1 – If Arg2 == 1, long pointer to a pathname of the item now being deleted</p> <p>Arg2 – status code: 1 = Start of individual item delete 2 = Completion of individual item delete</p>
HSFTP_RECURS_USR_EV_FOLDER_PROGRESS	<p>Indicates individual item operation progress (item upload, download or delete)</p> <p>Arg1 – percent complete 0 – 100 Arg2 – 0</p>

4.9 Recursive Operations Module Return Codes

Return code	Description
HSFTP_RECURS_RC_OK	Success
HSFTP_RECURS_RC_BUSY	Invalid state, recursive folder operation in progress
HSFTP_RECURS_RC_NOTINIT	HsFtpRecurs module not initialised
HSFTP_RECURS_RC_INVPAR	Invalid parameters
HSFTP_RECURS_RC_NOMEM	Out of memory
HSFTP_RECURS_RC_RCHDIR	Remote change directory failed
HSFTP_RECURS_RC_LIST	List command failed
HSFTP_RECURS_RC_FILE	Individual file operation failed
HSFTP_RECURS_RC_RMDIR	Remove directory at remote FTP server failed
HSFTP_RECURS_RC_LCHDIR	Failed to change into local directory
HSFTP_RECURS_RC_FOPEN	Failed to open local file
HSFTP_RECURS_RC_RMkdir	Failed to create folder at remote FTP server
HSFTP_RECURS_RC_FSEND	Failure during sending a file
HSFTP_RECURS_RC_PATH	Pathname too long