

# HS POP3 Library v2.0.2 User Manual

Revision: 1.4  
Date: 04 July 2011

<b>HS POP3 LIBRARY V2.0.2 USER MANUAL.....</b>	<b>1</b>
<b>REVISION: 1.4.....</b>	<b>1</b>
<b>DATE: 04 JULY 2011.....</b>	<b>1</b>
<b>1 INTRODUCTION.....</b>	<b>2</b>
<b>2 HS POP3 API.....</b>	<b>3</b>
<b>2.1 HsPOP3INIT.....</b>	<b>3</b>
<b>2.2 HsPOP3DESTROY.....</b>	<b>4</b>
<b>2.3 HsPOP3GETMAIL.....</b>	<b>4</b>
<b>2.4 HsPOP3ABORT.....</b>	<b>5</b>
<b>2.5 HsPOP3CONTINUE.....</b>	<b>6</b>
<b>2.6 HsPOP3GETERRSTR.....</b>	<b>6</b>
<b>2.7 HsPOP3MIMEMSGHASATTACHMENTS.....</b>	<b>7</b>
<b>2.8 HsPOP3MIMEGETMULTIPARTINFO.....</b>	<b>7</b>
<b>2.9 HsPOP3MIMEGETTEXTPART.....</b>	<b>9</b>
<b>2.10 HsPOP3MIMEGETBASE64FILE.....</b>	<b>10</b>
<b>3 HS POP3 TO USER EVENT CALLBACK AND EVENTS.....</b>	<b>12</b>
<b>3.1 EVENT CALLBACK PROTOTYPE.....</b>	<b>12</b>
<b>3.2 EVENTS.....</b>	<b>12</b>

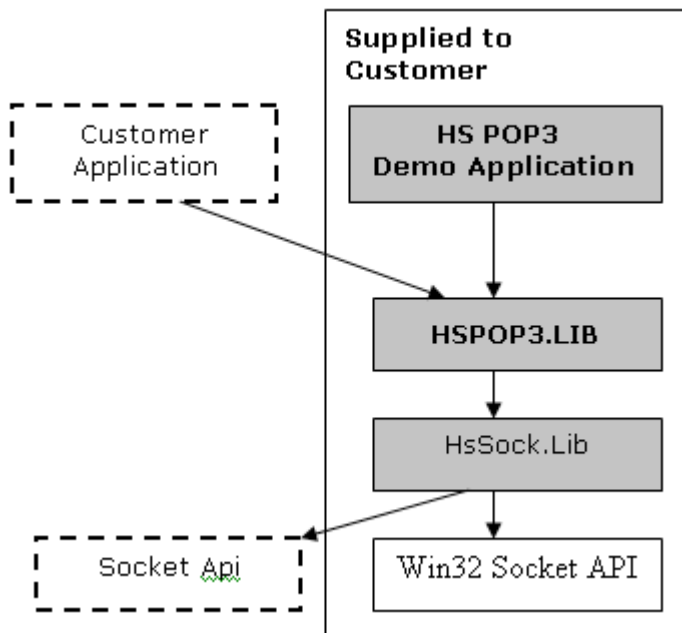
## 1 Introduction

HS POP3 is a software library in C (supplied with full source code) which implements the client side of Post Office Protocol Version 3 (POP3) over TCP socket layer according to RFC 1939. Among other features, the library supports user authentication, reception of basic internet headers and text, message deletion, statistics, MIME v1.0 base64 binary file attachments.

HS POP3 supports secure POP3 in 2 modes:

- Legacy mode: POP3S, used by Gmail
- RFC 2595

The library is a stand-alone module which links directly to customer application:



HS POP3 Library incorporates the necessary state machine, transparency procedures, and server response processing required to comply to a simple and robust POP3 client implementation.

## 2 HS POP3 API

### 2.1 HsPop3Init

int <b>HsPop3Init</b> (hs_pop3_api_t *api);		
<b>DESCRIPTION</b>		
<i>This function initialises HS POP3 Library. It must be called at initialisation, before any other functions are called. Calling HsPop3Init twice will return an error. You can call HsPop3Destroy first to deinitialise HS Pop3 Library and then call HsPop3Init again</i>		
<b>PARAMETERS</b>		
Type	Name	Description
hs_pop3_api_t	*api	Pointer to parameter structure, please see next section for details
<b>RETURNS</b>		
<ul style="list-style-type: none"> <li>• <i>HS_POP3_RC_ALRINIT – HS POP3 Library already initialised</i></li> <li>• <i>HS_POP3_RC_INV_PAR – invalid parameters specified</i></li> <li>• <i>HS_POP3_RC SOCKINIT_FAIL – socket layer initialisation failed</i></li> <li>• <i>HS_POP3_RC_OK - success</i></li> </ul>		

#### 2.1.1 Initialisation Structure Definition (hs\_pop3\_api\_t)

Field name	Description
start_timer_t    *start_timer;	Pointer function in user code used by HS POP3 to start a timer  <u>Prototype:</u> long start_timer_t(long handle, unsigned long secs, timer_callback_t *callback);  <u>Parameters:</u> Handle – POP3 session handle secs – timeout in seconds *callback – pointer to function in HS POP3 code that the user code should call on timer expiry.  <u>Callback function prototype:</u> void timer_callback_t(long handle); handle – HS POP3 session handle  <u>Returns:</u> timer handle or NULL if timer start error
stop_timer_t    *stop_timer	Pointer function in user code used by HS POP3 to stop a timer  <u>Prototype</u> void stop_timer_t(long timer_handle);  <u>Parameters:</u> Timer_handle – timer handle (returned from call to start_timer)
event_callback_t *event_cb	Pointer function in user code used by HS POP3 to report operation results, status and errors. (Please see session 3)
log_fn_t        *log_fn	Event log callback function. If logging debug events is enabled, HS POP3 calls this function when it needs to print an event string into application event log.

## 2.2 HsPop3Destroy

int <b>HsPop3Destroy</b> (void);		
<b>DESCRIPTION</b>		
<i>This function de-initialises HS POP3 Library, releases all used resources and cleans up Socket Interface Layer.</i>		
<b>PARAMETERS</b>		
Type	Name	Description
None	None	None
<b>RETURNS</b>		
<ul style="list-style-type: none"> <li>• <i>HS_POP3_RC_OK – success, HS POP3 Library successfully de-initialised</i></li> <li>• <i>HS_POP3_RC_NOTINIT – cannot destroy, HS POP3 Library not yet initialised</i></li> </ul>		

## 2.3 HsPop3GetMail

int HsPop3GetMail( <i>hs_pop3_session_t</i> *s, long *session handle);		
<b>DESCRIPTION</b>		
<p><i>This function initiates mail reception from POP3 server. It works as follows:</i></p> <ol style="list-style-type: none"> <li>1) <i>HS POP3 contacts POP3 server with login information</i></li> <li>2) <i>If login is authorised, HS POP3 checks to see if mailbox has any messages</i></li> <li>3) <i>If mailbox is not empty, HS POP3 follows this procedure for each message until all messages processed:</i> <ol style="list-style-type: none"> <li>A) <i>Get message id from server and report it to user application via HS_POP3_USR_EV_MSGID event</i></li> <li>B) <i>If user application indicates via return of callback that it wants to receive this message, HS POP3 receives message header, parses header fields and passes it to application via HS_POP3_USR_EV_DONE_MSGHDR event.</i></li> <li>C) <i>HS POP3 then receives the message body from the server block by block and each block's data is returned to application via HS_POP3_USR_EV_DONE_MSGBLK event.</i></li> </ol> <p style="margin-left: 20px;"><i>When message is fully received and all blocks have been returned to application, HS POP3 generates HS_POP3_USR_EV_DONE_MSG event</i></p> <li>D) <i>If there are more messages to be received from the mail server, HS POP3 does to step A</i></li> <li>E) <i>When all messages are processed, HS POP3 calls event callback with HS_POP3_USR_EV_DONE event passing total number of messages in mailbox and number of received messages</i></li> </li></ol> <p><i>Depending on parameters, HS POP3 can delete messages from the server after successful reception.</i></p> <p><i>If proto member of hs_pop3_session_t structure is set to HS_POP3_PROTO_POP3, standard POP3 protocol without SSL / TLS is used.</i></p> <p><i>If proto member of hs_pop3_session_t structure is set to HS_POP3_PROTO_POP3S, POP3S secure SMTP method is used. Gmail for example uses this method. According to it, SSL negotiation starts immediately after TCP connection from client to server is established on special "secure" port number which is different from standard SMTP. In this case srv_port of hs_pop3_session_t structure must be set to known secure port where the server is expecting start of SSL negotiation. For Gmail it is currently port 995</i></p> <p><i>If proto member of hs_pop3_session_t structure is set to HS_POP3_PROTO_RFC2595, POP3 transaction starts on standard POP3 port 110. SSL security is requested by HS POP3 by sending TLS command to server.</i></p>		

## PARAMETERS

Type	Name	Description
hs_pop3_session_t	*s	<p>Pointer to session parameters structure:</p> <p>srv_name – POP3 server name (Note 1)            username – POP3 username, 0 terminated            password – POP3 password, 0 terminated            srv_port - POP3 server port (usually 110)            delete_msgs – is TRUE, delete messages from server            user_data – user handle, not modified by HS POP3 and always passed back in event callback function.</p> <p>proto – protocol to use, as follows:            HS_POP3_PROTO_POP3 – standard POP3, no SSL/TLS            HS_POP3_PROTO_POP3S – POP3S mode enabled            HS_POP3_PROTO_RFC2595 – POP3 over TLS, RFC2595</p> <p>check_server_certificate – if protocol is HS_POP3_PROTO_POP3 or HS_POP3_PROTO_RFC2595, this variable controls if HS POP3 library will validate server certificate received during SSL negotiation.</p> <p>log_enabled – enabled debug event logging via log_fn callback supplied to HsPop3Init function.</p> <p>Note 1: Either server name or server IP address string can be supplied in the srv_name to the function</p>
long	*session_handle	HS POP3 fills in the variable pointed to by this parameter with POP3 session reference. Use this parameter to close a session with HsPop3Abort if required.

## RETURNS

- *HS\_POP3\_RC\_NOTINIT – HS POP3 Library not initialised*
- *HS\_POP3\_RC\_INV\_PAR – invalid parameters specified*
- *HS\_POP3\_RC\_NOPASS – no password specified in parameters*
- *HS\_POP3\_RC\_NOUSER – no username specified in parameters*
- *HS\_POP3\_RC\_NO\_CTX – maximum number of open HS POP3 sessions reached*
- *HS\_POP3\_RC\_NONAME – POP3 server name could not be resolved*
- *HS\_POP3\_RC\_TCPCONNFAIL - TCP connection to server could not be established*
- *HS\_POP3\_RC\_TIMERFAIL – timer start failure*
- *HS\_POP3\_RC\_OK - success*

### 2.4 HsPop3Abort

int **HsPop3Abort**(long session\_handle)

## DESCRIPTION

*This function is used to abort current mail session in progress. HS POP3 library will discard current message (if not fully received) free message memory and exit mail session cleanly via POP3 QUIT command. This means that the session is not closed immediately, but only after reception of a valid response to QUIT command or timeout.*

## PARAMETERS

Type	Name	Description
long	session_handle	POP3 session handle of the session to abort
<b>RETURNS</b>		
<ul style="list-style-type: none"> <li>• <i>HS_POP3_RC_OK – success</i></li> <li>• <i>HS_POP3_RC_NOTINIT – HS POP3 Library not initialised</i></li> <li>• <i>HS_POP3_RC_INV_PAR – invalid parameters specified</i></li> </ul>		

## 2.5 HsPop3Continue

int HsPop3Continue(long session_handle)		
<b>DESCRIPTION</b>		
<i>This function is only applicable if POP3 over SSL /TLS has been enabled.</i>		
<i>When calling HsPop3GetMail, user specifies several parameters related to secure POP3:</i>		
<ol style="list-style-type: none"> <li>1. <i>proto</i></li> <li>2. <i>check_server_certificate</i></li> </ol>		
<i>If proto is set to HS_POP3_PROTO_POP3 or HS_POP3_PROTO_RFC2595 and check_server_certificate is set to 1, HS POP3 shall check server certificate as part of SSL negotiation. If server certificate validation fails, HS POP3 calls event callback function with event HS_POP3_USR_EV_BADCERT. From that point HS POP3 is in wait state, expecting user application to either abort by calling HsPop3Abort or to ignore this error, continue and call this function HsPop3Continue, if server certificate is not important.</i>		
<b>PARAMETERS</b>		
Type	Name	Description
long	session_handle	POP3 session handle of the session to abort
<b>RETURNS</b>		
<ul style="list-style-type: none"> <li>• <i>HS_POP3_RC_OK – success</i></li> <li>• <i>HS_POP3_RC_NOTINIT – HS POP3 Library not initialised</i></li> <li>• <i>HS_POP3_RC_INV_PAR – invalid parameters specified</i></li> </ul>		

## 2.6 HsPop3GetErrStr

unsigned char *HsPop3GetErrStr (int rc)		
<b>DESCRIPTION</b>		
<i>This function is used to convert integer HS POP3 return code into a readable string – error description</i>		
<b>PARAMETERS</b>		
Type	Name	Description
int	rc	HS POP3 integer return code
<b>RETURNS</b>		
<ul style="list-style-type: none"> <li>• <i>Pointer to zero terminated error string</i></li> <li>• <i>NULL – error not found (not a valid return code)</i></li> </ul>		

## 2.7 HsPop3MimeMsgHasAttachments

```
int HsPop3MimeMsgHasAttachments(unsigned char *pMsgBuf, long len);
```

### DESCRIPTION

*This function is used to determine if the received message has MIME v1.0 attachments, such as file binary or html/text attachments.*

### PARAMETERS

Type	Name	Description
unsigned char	*pMsgBuf	Buffer containing received message headers. The message headers are separated from start of message body with a sequence of 4 bytes 0x0d 0x0a 0x0d 0x0a (CR LF CR LF)
long	len	Length of message buffer in bytes

### RETURNS

- *TRUE – message has attachments*
- *FALSE – message has no attachments*

## 2.8 HsPop3MimeGetMultiPartInfo

```
void HsPop3MimeGetMultiPartInfo(
    void *MsgHan, read_msg_block_t *read_fn, mime_mpart_info_cb_t *info_cb
);
```

### DESCRIPTION

*This function is used to extract information for each part of MIME v1.0 multipart message.*

*The supplied callback function info\_cb is called for each part found within MIME v1.0 multipart message.*

*The info\_cb function is passed pointer to hs\_pop3\_msgpart\_info\_t structure which describes the part of MIME multipart message. This structure contains boundary string which separates this part from the rest of the message, filename of the file encoded within this MIME multipart, and an integer type – identifying the type of part content.*

*Another parameter to HsPop3MimeGetMultiPartInfo function is read\_fn callback. This callback function is called by HS POP3 to read the next block of the message for parsing. The user may define this callback to read a disk file containing the message or to read memory content. The void \*MsgHan parameter to HsPop3MimeGetMultiPartInfo and to read\_fn callback identifies a handle for reading the message. If the read\_fn function reads a disk file, user may pass the file handle to HsPop3MimeGetMultiPartInfo. Otherwise the value of MsgHan is dependant on the implementation of the read\_fn callback.*

### PARAMETERS

Type	Name	Description
void	*MsgHan	Handle for reading next message block. This handle will be passed unchanged to read_fn callback by HS POP3 when it calls read_fn callback to read next part of the message. If read_fn callback is defined as a function reading disk file, pass file handle to MsgHan.

read_msg_block_t	*read_fn	<p>Callback function to be called by HS POP3 to read the next block of the message for parsing. The user may define this callback to read a disk file containing the message or to read memory content. The void *MsgHan parameter to HsPop3MimeGetMultiPartInfo and to read_fn callback identifies a handle for reading the message. If the read_fn function reads a disk file, user may pass the file handle to HsPop3MimeGetMultiPartInfo. Otherwise the value of MsgHan is dependant on the implementation of the read_fn callback</p> <p>Prototype:  <pre>typedef unsigned char *read_msg_block_t(     void *MsgHan, int want_bytes, int *read_bytes );</pre> </p> <p>void *MsgHan – handle passed as first parameter to HsPop3MimeGetMultiPartInfo.</p> <p>int want_bytes – number of bytes requested to read  int *read_bytes – the callback come must return number of bytes actually read.</p> <p>Returns pointer to block of memory containing the portion of message read or NULL if error occurred or end of message reached.</p>
mime_mpart_info_cb_t	*info_cb	<p>Pointer to callback function called by HS POP3 MIME code when it processed a given message body. For each found part within a message this callback is called once giving the information about message part</p> <p>Prototype:  <pre>typedef void mime_mpart_info_cb_t(hs_pop3_msgpart_info_t *pInfo);</pre> </p> <p>structure hs_pop3_msgpart_info_t is defined as:</p> <pre>// message part part info typedef struct {     unsigned char boundary[256];     unsigned char filename[256];     int type; } hs_pop3_msgpart_info_t;</pre> <p>boundary: part boundary string, zero terminated</p> <p>filename: part filename, zero terminated string. May not be used for non-file content type</p> <p>type: content type, one of the following:</p> <p>HSPOP3_MIME_UNKNOWN - unknown  HSPOP3_MIME_MULTIPART – mime multi-part  HSPOP3_MIME_TEXT_ASCII - ASCII text  HSPOP3_MIME_TEXT_HTML – HTML text  HSPOP3_MIME_BINARY – binary file</p>

## R E T U R N S

NONE

## 2.9 HsPop3MimeGetTextPart

```
int HsPop3MimeGetTextPart(
    void *MsgHan,
    read_msg_block_t *read_fn,
    unsigned char *boundary_val,
    mime_textpart_line_cb_t *cb);
```

### DESCRIPTION

*This function is used to extract text from text part of the MIME multi-part message. The text is extracted line by line by HS POP3 calling the supplied callback function mime\_textpart\_line\_cb\_t \*cb.*

### PARAMETERS

Type	Name	Description
void	*MsgHan	Handle for reading next message block. This handle will be passed unchanged to read_fn callback by HS POP3 when it calls read_fn callback to read next part of the message. If read_fn callback is defined as a function reading disk file, pass file handle to MsgHan.
read_msg_block_t	*read_fn	<p>Callback function to be called by HS POP3 to read the next block of the message for parsing. The user may define this callback to read a disk file containing the message or to read memory content. The void *MsgHan parameter to HsPop3MimeGetTextPart and to read_fn callback identifies a handle for reading the message. If the read_fn function reads a disk file, user may pass the file handle to HsPop3MimeGetTextPart. Otherwise the value of MsgHan is dependant on the implementation of the read_fn callback</p> <p>Prototype:            typedef unsigned char *read_msg_block_t(                void *MsgHan, int want_bytes, int *read_bytes            );</p> <p>void *MsgHan – handle passed as first parameter to HsPop3MimeGetMultiPartInfo.</p> <p>int want_bytes – number of bytes requested to read            int *read_bytes – the callback come must return number of bytes actually read.</p> <p>Returns pointer to block of memory containing the portion of message read or NULL if error occurred or end of message reached.</p>
unsigned char	*boundary_val	Pointer to null terminated string, boundary value of the text part within MIME multi-part message, Boundary string is obtained with a call to HsPop3MimeGetMultiPartInfo

mime_textpart_line_cb_t	*cb	Pointer to callback function called for each new text line parsed from requested part of the MIME multi-part message  Prototype: typedef void mime_textpart_line_cb_t(unsigned char *pLine);  pLine: pointer to next text file parsed from the message, null terminated
<b>RETURNS</b>		
<ul style="list-style-type: none"> <li>• <i>TRUE</i> – completed with success</li> <li>• <i>FALSE</i> – error occurred while parsing the message</li> </ul>		

## 2.10 HsPop3MimeGetBase64File

<pre>extern int HsPop3MimeGetBase64File(     void *MsgHan,     read_msg_block_t *read_fn,     unsigned char *boundary_val,     unsigned char *filename,     unsigned char *pUserData,     mime_file_buffer_cb_t *cb);</pre>		
<b>DESCRIPTION</b>		
<p><i>This function is used to extract binary file content from binary part of the MIME multi-part message.</i></p> <p><i>The file is extracted buffer by buffer in a number of calls to callback function mime_file_buffer_cb_t *cb.</i></p> <p>The message is read in for parsing as necessary block by block by HS POP3 using supplied callback function read_msg_block_t *read_fn. The read_fn callback may be implemented in user code as a function reading disk file or memory. The *MsgHan is a handle to be passed unchanged by HS POP3 when calling read_fn callback. If reading a disk file, pass file handle in MsgHan parameter.</p>		
<b>PARAMETERS</b>		
<b>Type</b>	<b>Name</b>	<b>Description</b>
void	*MsgHan	Handle for reading next message block. This handle will be passed unchanged to read_fn callback by HS POP3 when if calls read_fn callback to read next part of the message. If read_fn callback is defined as a function reading disk file, pass file handle to MsgHan.

read_msg_block_t	*read_fn	<p>Callback function to be called by HS POP3 to read the next block of the message for parsing. The user may define this callback to read a disk file containing the message or to read memory content. The void *MsgHan parameter to HsPop3MimeGetBase64File and to read_fn callback identifies a handle for reading the message. If the read_fn function reads a disk file, user may pass the file handle to HsPop3MimeGetBase64File. Otherwise the value of MsgHan is dependant on the implementation of the read_fn callback</p> <p>Prototype:  <pre>typedef unsigned char *read_msg_block_t(     void *MsgHan, int want_bytes, int *read_bytes );</pre> </p> <p>void *MsgHan – handle passed as first parameter to HsPop3MimeGetBase64File.</p> <p>int want_bytes – number of bytes requested to read  int *read_bytes – the callback come must return number of bytes actually read.</p> <p>Returns pointer to block of memory containing the portion of message read or NULL if error occurred or end of message reached.</p>
unsigned char	*boundary_val	<p>Pointer to null terminated string, boundary value of the binary part within MIME multi-part message, Boundary string is obtained with a call to sPop3MimeGetMultiPartInfo</p>
unsigned char	*filename	<p>Filename of the binary file to extract. Filename is obtained with a call to sPop3MimeGetMultiPartInfo.</p>
unsigned char	*pUserData	<p>User handle to be passed back unchanged to mime_file_buffer_cb_t *cb callback for each new block of file extracted. User may pass file handle of the file created for saving the extracted binary file content.</p>
mime_file_buffer_cb_t	*cb	<p>Pointer to callback function called for each next block of binary file extracted from MIME message.</p> <p>Prototype:  <pre>typedef void mime_file_buffer_cb_t(void *pUserData,     unsigned char *pBuffer, int length);</pre> </p> <p>pUserData: User handle, same as in pUserData parameter to HsPop3MimeGetBase64File.</p> <p>pBuffer: pointer to buffer containing next block of file extracted.</p> <p>Length: length of buffer extracted in bytes.</p>

## R E T U R N S

- *TRUE* – completed with success
- *FALSE* – error occurred while parsing the message

### 3 HS POP3 to USER Event Callback and Events

#### 3.1 Event Callback Prototype

```
typedef int event_callback_t(long handle, int ev, long arg1, long arg2);
```

Parameter	Description
handle	User handle, the same as specified in call to HsPop3getMail in user_data parameter of hs_pop3_session_t structure
ev	Event code (see next section for list of event codes)
Arg1	Parameter 1, value depends on event
Arg2	Parameter 2, value depends on event

Returns:

True or False. For most events the return is insignificant and is not checked by HS POP3. Where HS POP3 needs a specific return, it is clearly specified in this document

#### 3.2 Events

Event	Arg1	Arg2	Description
HS_POP3_USR_EV_MSGID	Pointer to message ID string, zero terminated with maximum length 71 bytes (not including last zero)	0	User application at this point may go through the list of locally stored (previously received) messages to see if it already has a message with the same message id. If the user wishes to receive this message, the return from callback function must be TRUE, otherwise to skip the message return FALSE
HS_POP3_USR_EV_DONE_MSGHDR	Pointer to message header structure received and parsed by HS POP3. The message header structure is defined in hs_pop3_msg_t	0	User application should store the message header of the message which is going now to be received by HS POP3  User must copy the passed structure content into a local structure.  HS POP3 shall release the internally allocated memory for the structure pointer on return from callback function

HS_POP3_USR_EV_DONE_MSGBLK	Pointer to buffer next block of message data received	Length of data block received	Next block of message data received  User must copy the passed buffer content into a local message storage (or mailbox file).  HS POP3 shall release the internally allocated memory for the buffer pointer on return from callback function
HS_POP3_USR_EV_DONE_MSG	0	0	Current message fully received
HS_POP3_USR_EV_DONE	Long number of messages received	0	Mail session complete
HS_POP3_USR_EV_CLOSED	0	0	Socket layer closed TCP connection. HS POP3 will release any allocated memory for a message being currently received
HS_POP3_USR_EV_CONNFALL	0	0	HS POP3 could not connect to POP3 server
HS_POP3_USR_EV_SRVERR	Unsigned char pointer to server reply string	Length of server reply string	Session closed because of POP3 error response received from server  HS POP3 will release any allocated memory for a message being currently received
HS_POP3_USR_EV_TIMEDOUT	Unsigned char pointer to additional information string (about the context in which timeout occurred)	Length of additional information string	Timed out waiting for server response, session closed. HS POP3 will release any allocated memory for a message being currently received
HS_POP3_USR_EV_PROGRESS1			HS POP3 reports number of messages in mailbox and total size of mailbox in octets. This event is indicated once per POP3 session
HS_POP3_USR_EV_PROGRESS2	Current message number (long)	Number of bytes received so far (long)	current receiving message number and number of bytes

			received so far.  This event is indicated for each message block received until full message is received. This event is indicated for each message within the same session. It can be used to indicated individual message reception progress.
HS_POP3_USR_EV_INTERR	Pointer to zero terminated error string	0	Internal error occurred, the TCP POP3 session has been closed and mail session context de-allocated
HS_POP3_USR_EV_BADCERT	Pointer to zero terminated error string	Length of error string	SSL server certificate validation was enabled and has failed. User must call HsPop3Continue to ignore this error and continue session or call HsPop3Abort to disconnect

### 3.2.1 Message structure (hs\_pop3\_msg\_t)

Data Member	Description
unsigned char from[HS_POP3_MAX_PATH];	FROM internet address, parsed out from message header
unsigned char date[HS_POP3_MAX_DATE];	DATE parsed out from message header
unsigned char subj[HS_POP3_MAX_SUBJ];	SUBJECT parsed out from message header
unsigned char msgid[HS_POP3_MAX_MSGID];	Message ID string as received from POP3 server for that message
int hdrlen;	Length of internet headers. The headers start from the first byte of the message
DWORD dwMsgLen;	Full message length including internet headers and data