

SMTP-32 Library

**Simple Mail Transfer Protocol
Dynamic Link Library
for Microsoft® Windows™**

Version 5.2

**Copyright © 1994 - 2003 by Distinct Corporation
All rights reserved**

Table of Contents

| | |
|----------------------------|----|
| 1 Overview | 5 |
| 1.1 Introduction | 5 |
| 1.2 Registry Entries | 5 |
| 1.3 Function Summary | 7 |
| 2 Reference | 9 |
| 2.1 smtp_data () | 9 |
| 2.2 smtp_data_buf () | 11 |
| 2.3 smtp_date () | 13 |
| 2.4 smtp_direct () | 14 |
| 2.5 smtp_expn () | 15 |
| 2.6 smtp_fw_helo () | 16 |
| 2.7 smtp_helo () | 19 |
| 2.8 smtp_help () | 21 |
| 2.9 smtp_mail () | 22 |
| 2.10 smtp_noop () | 23 |
| 2.11 smtp_opts () | 24 |
| 2.12 smtp_quit () | 26 |
| 2.13 smtp_rcpt () | 27 |
| 2.14 smtp_rply () | 28 |
| 2.15 smtp_rset () | 29 |
| 2.16 smtp_saml () | 30 |
| 2.17 smtp_send () | 31 |
| 2.18 smtp_soml () | 32 |
| 2.19 smtp_turn () | 33 |
| 2.20 smtp_vrfy () | 34 |
| 2.21 smtp_xhelo () | 35 |
| 2.22 smtp_xquit () | 37 |

1 Overview

1.1 Introduction

The Distinct SMTP-32 (Simple Mail Transfer Protocol) library provides developers with an API to a mail delivery service which usually runs on mail servers (such as UNIX workstations) to allow mail clients (usually PCs) to send mail to other mail servers or clients on the network.

Once a connection has been established with a server using the `smtp_(x)helo` or `smtp_fw_helo` call, a new message transfer is initiated by calling `smtp_mail`, `smtp_send`, `smtp_saml` or `smtp_soml`. The only difference between these four calls lies in how messages are delivered to their recipients. The second step to sending a message is to specify one or more recipients by calling the `smtp_rcpt` function one or more times. The third and final step to deliver a message is to call the `smtp_data` function to actually transfer the required message headers and the body of the message. These three steps can be repeated as often as needed to send all queued messages.

When a connection is no longer needed, it must be closed with a call to `smtp_quit`. Connections should not be kept open when not in use since they consume valuable system resources. An application must close all open connections before it terminates.

Most functions in the SMTP-32 library wait for a reply from the SMTP server before returning to the caller. Because communication links or servers sometimes go down, these functions use a default timeout of 20 seconds before returning the error value `SMTP_TIMED_OUT` if no response was received. This timeout value can be changed by editing or creating a registry entry as described in the section '[1.2 - Registry Entries](#)'.

While waiting for a reply to a command or while waiting for a data transfer to complete, most functions in the SMTP-32 library yield control to Windows. This allows applications to process messages while an SMTP operation is in progress. Without yielding, the system would appear to be frozen for the entire duration of each SMTP operation. By default, yielding is therefore enabled but can be disabled if necessary with the `smtp_opts` call. Yielding control to Windows is done by the SMTP-32 library by frequently calling an internal yielding function during lengthy operations. The calling application can also define its own yielding function to be called instead of the default yielding function.

1.2 Registry Entries

Various parameters used to control the behavior of the Distinct SMTP-32 library are stored under the following registry path.

HKEY_LOCAL_MACHINE\SOFTWARE\Distinct\DLLS\SMTP32

The following entries specify various Distinct SMTP-32 library settings.

| | | |
|----------------|------------------|--------------|
| Timeout | REG_DWORD | <i>1-999</i> |
|----------------|------------------|--------------|

Specifies the default timeout in seconds used while connecting to the remote SMTP server and for other SMTP operations. The default value is 20 seconds.

DebugLevel **REG_DWORD** 0-2

Specifies what debug information should be logged into the D32-SMTP.DBG file in the directory from which D32-SMTP.DLL was loaded. The following values are supported.

| Value | Meaning |
|--------------|---|
| 0 | Disable logging. |
| 1 | Log all errors. |
| 2 | Log all function calls and return values. |

A higher debug level automatically includes all lower debug levels. The default value is 0.

1.3 Function Summary

smtp_data

Send message data

smtp_data_buf

Send message data

smtp_date

Get current time and date in RFC 822 format

smtp_direct

Send a command to the SMTP server

smtp_expn

Send expand command

smtp_fw_helo

Open connection with SMTP server via the firewall

smtp_helo

Open connection with SMTP server

smtp_help

Send help command

smtp_mail

Send mail command

smtp_noop

Do nothing

smtp_opts

Set connection option

smtp_quit

Close connection with SMTP server

smtp_rcpt

Send recipient command

smtp_rply

Get last server reply

smtp_rset

Send reset command

smtp_saml

Send send and mail command

smtp_send

Send send command

smtp_soml

Send send or mail command

smtp_turn

Send turn command

smtp_vrfy

Send verify command

smtp_xhelo

Asynchronous version of smtp_helo

smtp_xquit

Cancel asynchronous connect or operation in progress

2 Reference

2.1 smtp_data ()

Description

Send message data.

```
#include <windows.h>
#include <d32-smtp.h>
```

```
int WINAPI smtp_data (hConn, proc)
```

```
    GLOBALHANDLE hConn;  Handle of SMTP structure from smtp_(x)helo
    FARPROC proc;        Application callback function
```

Remarks

The **smtp_data** function is used to transfer a message including required headers to the SMTP server. Before calling **smtp_data**, an application must have made one successful call to **smtp_mail**, **smtp_send**, **smtp_saml** or **smtp_soml** and one or more successful calls to **smtp_rept**. These function calls identify the sender and one or more recipients of the message.

The **smtp_data** function transfers a message to the server by calling the application supplied callback function *proc* for message data until *proc* returns zero to indicate the end of the message. The callback function *proc* must return the number of bytes it copied into the buffer and must be defined as follows:

```
int WINAPI (*proc) (hConn, buf, len, lParam)
```

```
GLOBALHANDLE hConn;
char far *buf;
int len;
DWORD lParam;
```

An application can abort the transfer of a message by returning -1 in the callback function or by calling the **smtp_xquit** function. Note that aborting a message transfer will also abort the underlying connection by resetting it. If the transfer was aborted successfully, then the **smtp_data** call will return SMTP_ABORTED and the connection handle *hConn* will no longer be valid. No more calls (including **smtp_quit**) can be made until a new connection has been established with **smtp_(x)helo**.

The connection handle *hConn* must have been obtained with a call to **smtp_(x)helo**.

Return Value The `smtp_data` function either returns `SMTP_SUCCESS` or one of the following errors.

`SMTP_INVALID_HANDLE`

Invalid connection handle was specified.

`SMTP_FAILURE`

Host was not able to execute command.

`SMTP_CANNOT_SEND_COMMAND`

Windows Sockets library was not able to accept outgoing data.

`SMTP_TIMED_OUT`

Time out occurred while waiting for a response from host.

`SMTP_IN_PROGRESS`

Another operation is already in progress.

`SMTP_ABORTED`

Operation was aborted and connection has been closed.

If this function returns `SMTP_FAILURE`, then the `smtp_rply` function can be used to retrieve the error message sent by the host which describes the reason why the command failed.

2.2 smtp_data_buf ()

Description

Send message data.

```
#include <windows.h>
#include <d32-smtp.h>
```

```
int WINAPI smtp_data_buf (hConn, buf, len)
```

| | |
|------------------------------------|---|
| GLOBALHANDLE <i>hConn</i> ; | Handle of SMTP structure from smtp_(x)helo |
| char far * <i>buf</i> ; | Containing data to be sent out |
| int <i>len</i> ; | Length of buffer |

Remarks

The **smtp_data_buf** function is used to transfer a message including required headers to the SMTP server. Before calling **smtp_data_buf**, an application must have made one successful call to **smtp_mail**, **smtp_send**, **smtp_saml** or **smtp_soml** and one or more successful calls to **smtp_rcpt**. These function calls identify the sender and one or more recipients of the message.

The **smtp_data_buf** can be called one or more times to send out the entire data. The *buf* parameter should contain the data to be sent out the and *len* should specify the length of data. To signify the end of data the *len* parameter should be set to 0.

The connection handle *hConn* must have been obtained with a call to **smtp_(x)helo**.

Return Value The `smtp_data_buf` function either returns `SMTP_SUCCESS` or one of the following errors.

`SMTP_INVALID_HANDLE`

Invalid connection handle was specified.

`SMTP_FAILURE`

Host was not able to execute command.

`SMTP_CANNOT_SEND_COMMAND`

Windows Sockets library was not able to accept outgoing data.

`SMTP_TIMED_OUT`

Time out occurred while waiting for a response from host.

`SMTP_IN_PROGRESS`

Another operation is already in progress.

`SMTP_ABORTED`

Operation was aborted and connection has been closed.

If this function returns `SMTP_FAILURE`, then the `smtp_rply` function can be used to retrieve the error message sent by the host which describes the reason why the command failed.

2.3 smtp_date ()

Description Get current time and date in RFC 822 format.

```
#include <windows.h>
```

```
#include <d32-smtp.h>
```

```
int WINAPI smtp_date (date)
```

```
    LPSTR date;           Buffer for formatted date string
```

Remarks The `smtp_date` function builds a string specifying the current time and date in RFC 822 format. This string can then be used to build the required date field in an SMTP message header.

Return Value The `smtp_date` function returns the length of the string placed into the buffer pointed to by *date*.

2.4 smtp_direct ()

Description Send a command to the SMTP server.

```
#include <windows.h>
#include <d32-smtp.h>
```

```
int WINAPI smtp_direct (hConn, msg, rep_buf, len)
```

| | |
|------------------------------------|-----------------------------|
| GLOBALHANDLE <i>hConn</i> ; | Handle of SMTP structure |
| LPSTR <i>msg</i> ; | Command to send |
| char * <i>rep_buf</i> ; | Buffer for command response |
| int * <i>len</i> ; | Length of response buffer |

Remarks This function sends the command specified by *msg* to the SMTP server. The response returned by the server is placed into the buffer pointed to by *rep_buf*. The size of the response buffer is specified by *len*. The return buffer is NULL terminated. If *rep_buf* is not large enough for the entire response then the message from the SMTP server will be truncated and the *len* parameter will contain the required size of the *rep_buf*.

Note: This function is supplied to allow the user to send any commands that are not supported by the SMTP Library. This method is not a substitute for the existing functions and should not be used as such. In particular this function should not be used to disconnect as the SMTP Library will not be aware of the action. This function cannot be used until a successful connection has been made with `smtp_helo` or `smtp_fw_helo`.

Return Value The `smtp_direct` function either returns SMTP_SUCCESS or one of the following errors.

SMTP_INVALID_HANDLE

Invalid connection handle was specified.

SMTP_FAILURE

Host was not able to execute command.

SMTP_CANNOT_SEND_COMMAND

Windows Sockets library was not able to accept outgoing data.

SMTP_TIMED_OUT

Time out occurred while waiting for a response from host.

SMTP_IN_PROGRESS

Another operation is already in progress.

SMTP_ABORTED

Operation was aborted and connection has been closed.

SMTP_INSUFFICIENT_LENGTH

The reply buffer (*rep_buf*) was not large enough to hold the entire server response. The *len* parameter will contain the required length of the buffer.

If this function returns SMTP_FAILURE, then the `smtp_rply` function can be used to retrieve the error message sent by the host which describes the reason why the command failed.

2.5 smtp_expn ()

Description Send expand command.

```
#include <windows.h>
#include <d32-smtp.h>
```

```
int WINAPI smtp_expn (hConn, name, proc)
```

```
    GLOBALHANDLE hConn;  Handle of SMTP structure from smtp_(x)helo
    LPSTR name;          Expression to expand
    FARPROC proc;       Application callback function
```

Remarks The **smtp_expn** function provides the caller with a list of all individual mail addresses contained in the mailing group specified by *name*. For each address in the group, the callback function *proc* is called with a string containing the full mail address of one recipient. The callback function *proc* must be defined as follows:

```
int WINAPI (*proc) (hConn, line, len, lParam)
```

```
GLOBALHANDLE hConn;
char far *line;
int len;
DWORD lParam;
```

An application can abort the listing by returning -1 in the callback function (all other return values are currently ignored) or by calling the **smtp_xquit** function. Note that aborting a listing will also abort the underlying connection by resetting it. If the listing was aborted successfully, then the **smtp_expn** call will return SMTP_ABORTED and the connection handle *hConn* will no longer be valid. No more calls (including **smtp_quit**) can be made until a new connection has been established with **smtp_(x)helo**.

The connection handle *hConn* must have been obtained with a call to **smtp_(x)helo**.

Return Value The **smtp_expn** function either returns SMTP_SUCCESS or one of the following errors.

```
SMTP_INVALID_HANDLE
    Invalid connection handle was specified.

SMTP_FAILURE
    Host was not able to execute command.

SMTP_CANNOT_SEND_COMMAND
    Windows Sockets library was not able to accept outgoing data.

SMTP_TIMED_OUT
    Time out occurred while waiting for a response from host.

SMTP_IN_PROGRESS
    Another operation is already in progress.

SMTP_ABORTED
    Operation was aborted and connection has been closed.
```

If this function returns SMTP_FAILURE, then the **smtp_rply** function can be used to retrieve the error message sent by the host which describes the reason why the command failed.

2.6 smtp_fw_helo ()

Description Open connection with SMTP server via the firewall.

```
#include <windows.h>
#include <d32-smtp.h>
#include <d32-fw.h>
```

```
int WINAPI smtp_fw_helo (local, conn, proc, lParam, fw)
```

| | |
|--------------------------------|--------------------------------------|
| char far *local; | Local name |
| GLOBALHANDLE far *conn; | Buffer for SMTP structure handle |
| FARPROC proc; | Application callback function |
| DWORD lParam; | Application supplied parameter |
| fw_param far *fw; | A pointer to the fw_param structure. |

Remarks The `smtp_fw_helo` function establishes a TCP/IP connection with the specified host via the firewall. The name pointed to by the `local` parameter is used to identify the local machine to the server and should specify the network name assigned to the local PC (possibly also including a domain name).

After a successful connection, the handle of the connection control structure is placed into the buffer pointed to by `conn`. This handle must be used in all subsequent calls to the other SMTP functions for this connection. The Distinct SMTP Library supports multiple concurrent SMTP connections limited only by system resources.

During development it may be necessary to observe exactly what commands are sent to the SMTP server and what replies are generated. For this purpose, the application can specify the address of a debug callback function with `proc`. This function will receive a copy of all network traffic and must be defined as follows.

```
int FAR PASCAL (*proc) (hConn, buf, len, lParam)
```

```
GLOBALHANDLE hConn;
char far *buf;
int len;
DWORD lParam;
```

The `proc` parameter must be set to NULL if the application does not want to receive debug messages.

The `lParam` parameter can be set to any value that has meaning to the application such as an index into an array or a pointer to a structure. This parameter is passed back to the application as an argument in each callback function and allows the application to pass connection specific data to the callback function.

The `fw` parameter is a pointer to the `fw_param` firewall structure which must contain all the connection specific fields required to establish connection to the SMTP server via the firewall. The `fw_param` structure has the following fields.

```
typedef struct fw_parameters
{
    char far * methods;                different types of methods.
    unsigned char n_methods;          the number of methods
    unsigned char add_type;           address type of the destination address
    char * domain_name;              the domain name
    char far * fw_host;               IP address of the firewall server
    char far * dest_host;             IP address of the SMTP server
    unsigned short fw_port;           firewall port
}
```

```

    unsigned short dest_port;           destination port
    char far * username;                valid user id
    char far * passwd;                  user password
    int socks_ver;                       the socks version;
    char reserved [256 - 6 * sizeof(char far *) - 2 * sizeof(unsigned char) - 2 * sizeof(unsigned short) - sizeof(int)];
} fw_param;

```

Distinct firewall library supports both SOCKS version 4 and version 5 firewall server. The *socks_ver* field of the **fw_param** structure should specify the SOCKS version. The *socks_ver* field should be any or a combination of the following values:

| | |
|-------------|------------------------|
| FW_VERSION5 | The Socks version is 5 |
| FW_VERSION4 | The Socks version is 4 |

If the *socks_ver* field contains both FW_VERSION5 and FW_VERSION4 then the Distinct firewall library will try to automatically determine the firewall server version and send appropriate information to the firewall server to establish a connection.

The *methods* field can point to string "0", "1", or "2" or any combination of "0", "1", "2", for example "01" or "12". Method 0 means no authentication is required, 1 means GSSAPI and 2 means a *username* and *passwd* is required. *n_methods* is the number of methods that appear in the *method* field. Note that only methods 0 and 2 are supported currently and if method 2 is specified then the application must specify a *username* and *passwd* for authentication. Note that the *methods*, *n_methods*, and *passwd* fields are required only if the application is trying to connect to a version 5 firewall server.

The field *addr_type* specifies the type of address of the remote host. The *addr_type* can be any one of the following types.

| Type | Meaning |
|-------------|------------------------------------|
| FW_ADDR_IP4 | address is a version 4 IP address |
| FW_ADDR_DNS | address is a DNS style domain name |
| FW_ADDR_IP6 | address is a version 6 IP address |

If the *addr_type* field is of type FW_ADDR_DNS then the *domain_name* field must point to a DNS-style domain name. If the value of the *addr_type* field is FW_ADDR_IP4 or FW_ADDR_IP6 then the *dest_host* field must contain the IP address of the destination host. Note that if the SOCKS version is 4 then the *addr_type* field should always be FW_ADDR_IP4 or FW_ADDR_DNS.

The *fw_host* field should contain the address of the firewall server. The port of the firewall server must be specified in the *fw_port* field, if this field is 0 then the firewall library will use the default port 1080. The field *dest_port* must contain port number of the destination host. Both *fw_port* and the *dest_port* must be in host byte order.

The *username* should contain a valid user id if the application is trying to connect to a SOCKS version 4 firewall server or if the application has specified method 2 for authentication in case of SOCKS version 5.

Note that any string pointer of the **fw_param** structure that is not used should be set to NULL.

Return Value The `smtp_fw_helo` function either returns `SMTP_SUCCESS` or one of the following errors.

`SMTP_FAILURE`

Remote error.

`SMTP_CANNOT_INITIALIZE`

Support libraries or network drivers could not be initialized.

`SMTP_CANNOT_INIT_WINSOCK`

Windows Sockets transport library could not be initialized.

`SMTP_OUT_OF_MEMORY`

Not enough memory to allocate connection control structure.

`SMTP_UNKNOWN_HOST`

Unable to resolve specified host name.

`SMTP_CANNOT_ALLOC_SOCKET`

Windows Sockets library could not allocate a socket.

`SMTP_CANNOT_BIND_SOCKET`

Windows Sockets library was not able to bind socket to local port.

`SMTP_HOST_NOT_RESPONDING`

Unable to connect to SMTP port on specified host.

`SMTP_CANNOT_SEND_COMMAND`

Windows Sockets library was not able to accept outgoing data.

`SMTP_TIMED_OUT`

Time out occurred while waiting for a response from host.

If this function returns `SMTP_FAILURE`, then the `smtp_rply` function can be used to retrieve the error message sent by the host which describes the reason why the command failed.

2.7 smtp_helo ()

Description Open connection with SMTP server.

```
#include <windows.h>
#include <d32-smtp.h>
```

```
int WINAPI smtp_helo (host, local, conn, proc, lParam)
```

| | |
|---|----------------------------------|
| LPSTR <i>host</i> ; | Name of SMTP server |
| LPSTR <i>local</i> ; | Local name |
| GLOBALHANDLE far * <i>conn</i> ; | Buffer for SMTP structure handle |
| FARPROC <i>proc</i> ; | Application callback function |
| DWORD <i>lParam</i> ; | Application supplied parameter |

Remarks The `smtp_helo` function establishes a TCP/IP connection with the specified host. The *host* parameter can point either to a machine name (possibly including a domain name) or to an internet address in dotted decimal notation (e.g. "192.1.2.3"). The name pointed to by the *local* parameter is used to identify the local machine to the server and should specify the network name assigned to the local PC (possibly also including a domain name).

After a successful connection, the handle of the connection control structure is placed into the buffer pointed to by *conn*. This handle must be used in all subsequent calls to the other SMTP functions for this connection. The Distinct SMTP-32 library supports multiple concurrent SMTP connections limited only by system resources.

During development it may be necessary to observe exactly what commands are sent to the SMTP server and what replies are generated. For this purpose, the application can specify the address of a debug callback function with *proc*. This function will receive a copy of all network traffic and must be defined as follows.

```
int WINAPI (*proc) (hConn, buf, len, lParam)
```

```
GLOBALHANDLE hConn;
char far *buf;
int len;
DWORD lParam;
```

The *proc* parameter must be set to NULL if the application does not want to receive debug messages.

The *lParam* parameter can be set to any value that has meaning to the application such as an index into an array or a pointer to a structure. This parameter is passed back to the application as an argument in each callback function and allows the application to pass connection specific data to the callback function.

Return Value The `smtp_helo` function either returns SMTP_SUCCESS or one of the following errors.

SMTP_FAILURE

Remote error.

SMTP_CANNOT_INITIALIZE

Support libraries or network drivers could not be initialized.

SMTP_CANNOT_INIT_WINSOCK

Windows Sockets transport library could not be initialized.

SMTP_OUT_OF_MEMORY

Not enough memory to allocate connection control structure.

SMTP_UNKNOWN_HOST

Unable to resolve specified host name.

SMTP_CANNOT_ALLOC_SOCKET

Windows Sockets library could not allocate a socket.

SMTP_CANNOT_BIND_SOCKET

Windows Sockets library was not able to bind socket to local port.

SMTP_HOST_NOT_RESPONDING

Unable to connect to SMTP port on specified host.

SMTP_CANNOT_SEND_COMMAND

Windows Sockets library was not able to accept outgoing data.

SMTP_TIMED_OUT

Time out occurred while waiting for a response from host.

If this function returns SMTP_FAILURE, then the `smtp_rply` function can be used to retrieve the error message sent by the host which describes the reason why the command failed.

2.8 smtp_help ()

Description Send help command.

```
#include <windows.h>
#include <d32-smtp.h>
```

```
int WINAPI smtp_help (hConn, topic, proc)
```

```
    GLOBALHANDLE hConn;  Handle of SMTP structure from smtp_(x)helo
    LPSTR topic;         Desired help topic
    FARPROC proc;        Application callback function
```

Remarks The **smtp_help** function provides the caller either with a list of all commands available on the SMTP server or with a detailed explanation of the command specified by *topic*. To obtain a list of all commands, the topic parameter should be set to NULL or point to an empty string. For each line received from the server, the callback function *proc* will be called with the line and its length. The callback function *proc* must be defined as follows:

```
int WINAPI (*proc) (hConn, line, len, lParam)

GLOBALHANDLE hConn;
char far *line;
int len;
DWORD lParam;
```

An application can abort the listing of help instructions by returning -1 in the callback function (all other return values are currently ignored) or by calling the **smtp_xquit** function. Note that aborting help instructions will also abort the underlying connection by resetting it. If the help instructions were aborted successfully, then the **smtp_help** call will return SMTP_ABORTED and the connection handle *hConn* will no longer be valid. No more calls (including **smtp_quit**) can be made until a new connection has been established with **smtp_(x)helo**.

The connection handle *hConn* must have been obtained with a call to **smtp_(x)helo**.

Return Value The **smtp_help** function either returns SMTP_SUCCESS or one of the following errors.

```
SMTP_INVALID_HANDLE
    Invalid connection handle was specified.

SMTP_FAILURE
    Host was not able to execute command.

SMTP_CANNOT_SEND_COMMAND
    Windows Sockets library was not able to accept outgoing data.

SMTP_TIMED_OUT
    Time out occurred while waiting for a response from host.

SMTP_IN_PROGRESS
    Another operation is already in progress.

SMTP_ABORTED
    Operation was aborted and connection has been closed.
```

If this function returns SMTP_FAILURE, then the **smtp_rply** function can be used to retrieve the error message sent by the host which describes the reason why the command failed.

2.9 smtp_mail ()

Description Send mail command.

```
#include <windows.h>
```

```
#include <d32-smtp.h>
```

```
int WINAPI smtp_mail (hConn, from)
```

```
    GLOBALHANDLE hConn; Handle of SMTP structure from smtp_(x)helo
```

```
    LPSTR from; Sender of mail
```

Remarks The **smtp_mail** function informs the SMTP server that a message from the individual specified by *from* is about to be uploaded. If this function succeeds, then one or more recipients of the message must be identified with one or more calls to the **smtp_rcpt** function. The actual message including required headers must then be sent with the **smtp_data** function. Once all three steps have been completed successfully, the message is placed into the mail boxes of all specified recipients.

The connection handle *hConn* must have been obtained with a call to **smtp_(x)helo**.

Return Value The **smtp_mail** function either returns SMTP_SUCCESS or one of the following errors.

```
SMTP_INVALID_HANDLE
```

Invalid connection handle was specified.

```
SMTP_FAILURE
```

Host was not able to execute command.

```
SMTP_CANNOT_SEND_COMMAND
```

Windows Sockets library was not able to accept outgoing data.

```
SMTP_TIMED_OUT
```

Time out occurred while waiting for a response from host.

```
SMTP_IN_PROGRESS
```

Another operation is already in progress.

```
SMTP_ABORTED
```

Operation was aborted and connection has been closed.

If this function returns SMTP_FAILURE, then the **smtp_rply** function can be used to retrieve the error message sent by the host which describes the reason why the command failed.

2.10 smtp_noop ()

Description Do nothing.

```
#include <windows.h>
```

```
#include <d32-smtp.h>
```

```
int WINAPI smtp_noop (hConn)
```

GLOBALHANDLE *hConn*; Handle of SMTP structure from **smtp_(x)helo**

Remarks The **smtp_noop** function sends an empty command to the SMTP server and waits for a proper reply. This command can be used to verify that the server is active. The connection handle *hConn* must have been obtained with a call to **smtp_(x)helo**.

Return Value The **smtp_noop** function either returns SMTP_SUCCESS or one of the following errors.

SMTP_INVALID_HANDLE

Invalid connection handle was specified.

SMTP_FAILURE

Host was not able to execute command.

SMTP_CANNOT_SEND_COMMAND

Windows Sockets library was not able to accept outgoing data.

SMTP_TIMED_OUT

Time out occurred while waiting for a response from host.

SMTP_IN_PROGRESS

Another operation is already in progress.

If this function returns SMTP_FAILURE, then the **smtp_rply** function can be used to retrieve the error message sent by the host which describes the reason why the command failed.

2.11 smtp_opts ()

Description Set connection option.

```
#include <windows.h>
```

```
#include <d32-smtp.h>
```

```
int WINAPI smtp_opts (hConn, option, value)
```

GLOBALHANDLE *hConn*; Handle of SMTP structure from **smtp_(x)helo**

WORD *option*; Option to set

DWORD *value*; Value of option

Remarks The **smtp_opts** function sets options on a per connection basis. The *option* parameter specifies which option to set and must be equal to one of the following constants.

| | |
|---------------|---------------------------------|
| OPT_YIELD_ON | Enable yielding. |
| OPT_YIELD_OFF | Disable yielding. |
| OPT_SET_PARAM | Change callback parameter. |
| OPT_GET_PARAM | Get current callback parameter. |
| OPT_DEBUG | Change debug callback. |
| OPT_TIMEOUT | Change timeout value. |

When enabling yielding with the OPT_YIELD_ON option, the *value* parameter can be used to specify the address of an application supplied yielding callback function. This function will be called during lengthy operations and should yield control to Windows so that other applications can process their messages. The function must be defined as follows.

```
int WINAPI (*proc) (void)
```

If an application wants to use the default SMTP-32 library yielding function, then the *value* parameter must be set to NULL. The default yielding function should be sufficient in most cases.

The *value* parameter is not used for the OPT_YIELD_OFF option and must be set to NULL.

The callback parameter supplied by the application in the **smtp_(x)helo** call can be changed with the OPT_SET_PARAM option. To do so, set *value* to the new callback parameter. The current value of the callback parameter can be obtained with the OPT_GET_PARAM option. In this case, *value* must point to the DWORD which is to receive the current callback parameter (i.e. *value* must be a DWORD far *).

The OPT_DEBUG option allows the application to change or disable the debug callback function. To change the debug callback function, set *value* to the address of the new callback function. Note that this function address must be obtained by calling **MakeProcInstance** specifying an exported function. To disable debug callbacks, set *value* to NULL. Refer to the **smtp_(x)helo** description for more details on debug callback functions.

The OPT_TIMEOUT option allows the application to set a timeout value other than the default for an open connection. The *value* parameter must specify the new timeout value in seconds and must be in the range of 1 to 999.

The connection handle *hConn* must have been obtained with a call to **smtp_(x)helo**.

Return Value The `smtp_opts` function either returns `SMTP_SUCCESS` or one of the following errors.

`SMTP_INVALID_HANDLE`

Invalid connection handle was specified.

`SMTP_INVALID_OPTION`

Invalid connection option was specified.

2.12 smtp_quit ()

Description Close connection with SMTP server.

```
#include <windows.h>
```

```
#include <d32-smtp.h>
```

```
int WINAPI smtp_quit (hConn)
```

GLOBALHANDLE *hConn*; Handle of SMTP structure from **smtp_(x)helo**

Remarks The **smtp_quit** function terminates an SMTP connection and frees all associated resources. The connection handle *hConn* must have been obtained with a call to **smtp_(x)helo** and will be invalid once the **smtp_quit** function returns.

Return Value The **smtp_quit** function either returns SMTP_SUCCESS or one of the following errors.

SMTP_INVALID_HANDLE

Invalid connection handle was specified.

SMTP_IN_PROGRESS

Another operation is already in progress.

2.13 smtp_rcpt ()

Description Send recipient command.

```
#include <windows.h>
```

```
#include <d32-smtp.h>
```

```
int WINAPI smtp_rcpt (hConn, to)
```

```
    GLOBALHANDLE hConn;  Handle of SMTP structure from smtp_(x)helo  
    LPSTR to;            Recipient of message
```

Remarks The **smtp_rcpt** function informs the SMTP server that the following message should be sent to the recipient specified by *to*. At least one call to **smtp_rcpt** must be made after calling **smtp_mail**, **smtp_send**, **smtp_saml** or **smtp_soml**, but multiple calls can be made to send the same message to multiple recipients. The actual message including required headers must then be sent with the **smtp_data** function.

The connection handle *hConn* must have been obtained with a call to **smtp_(x)helo**.

Return Value The **smtp_rcpt** function either returns SMTP_SUCCESS or one of the following errors.

SMTP_INVALID_HANDLE

Invalid connection handle was specified.

SMTP_FAILURE

Host was not able to execute command.

SMTP_CANNOT_SEND_COMMAND

Windows Sockets library was not able to accept outgoing data.

SMTP_TIMED_OUT

Time out occurred while waiting for a response from host.

SMTP_IN_PROGRESS

Another operation is already in progress.

SMTP_ABORTED

Operation was aborted and connection has been closed.

If this function returns SMTP_FAILURE, then the **smtp_rply** function can be used to retrieve the error message sent by the host which describes the reason why the command failed.

2.14 smtp_rply ()

Description Get last server reply.

```
#include <windows.h>
```

```
#include <d32-smtp.h>
```

```
int WINAPI smtp_rply (hConn, reply, len, code)
```

GLOBALHANDLE *hConn*; Handle of SMTP structure from **smtp_(x)helo**

char far **reply*; Buffer for last server reply

int *len*; Size of server reply buffer

int far **code*; Buffer for last server reply code

Remarks The **smtp_rply** function retrieves the last response sent by the SMTP server. This can be useful in determining the cause of an SMTP_FAILURE error return. The buffer pointed to by *reply* will receive the entire response string from the server and the buffer pointed to by *code* will receive the integer equivalent of the reply code contained in the response.

The connection handle *hConn* must have been obtained with a call to **smtp_(x)helo**.

Return Value The **smtp_rply** function either returns SMTP_SUCCESS or SMTP_INVALID_HANDLE to indicate that an invalid connection handle was specified.

2.15 smtp_rset ()

Description Send reset command.

```
#include <windows.h>
```

```
#include <d32-smtp.h>
```

```
int WINAPI smtp_rset (hConn)
```

GLOBALHANDLE *hConn*; Handle of SMTP structure from **smtp_(x)helo**

Remarks The **smtp_rset** function resets the SMTP server to its original state. The connection handle *hConn* must have been obtained with a call to **smtp_(x)helo**.

Return Value The **smtp_rset** function either returns SMTP_SUCCESS or one of the following errors.

SMTP_INVALID_HANDLE

 Invalid connection handle was specified.

SMTP_FAILURE

 Host was not able to execute command.

SMTP_CANNOT_SEND_COMMAND

 Windows Sockets library was not able to accept outgoing data.

SMTP_TIMED_OUT

 Time out occurred while waiting for a response from host.

SMTP_IN_PROGRESS

 Another operation is already in progress.

SMTP_ABORTED

 Operation was aborted and connection has been closed.

If this function returns SMTP_FAILURE, then the **smtp_rply** function can be used to retrieve the error message sent by the host which describes the reason why the command failed.

2.16 smtp_saml ()

Description Send send and mail command.

```
#include <windows.h>
```

```
#include <d32-smtp.h>
```

```
int WINAPI smtp_saml (hConn, from)
```

```
    GLOBALHANDLE hConn; Handle of SMTP structure from smtp_(x)helo
```

```
    LPSTR from; Sender of mail
```

Remarks The **smtp_saml** function informs the SMTP server that a message from the individual specified by *from* is about to be uploaded. If this function succeeds, then one or more recipients of the message must be identified with one or more calls to the **smtp_rcpt** function. The actual message including required headers must then be sent with the **smtp_data** function. Once all three steps have been completed successfully, the message is placed into the mail boxes of all specified recipients.

Unlike the **smtp_mail** function, the **smtp_saml** function also sends messages directly to the terminals of all specified recipients currently logged into the server.

The connection handle *hConn* must have been obtained with a call to **smtp_(x)helo**.

Return Value The **smtp_saml** function either returns SMTP_SUCCESS or one of the following errors.

```
SMTP_INVALID_HANDLE
```

Invalid connection handle was specified.

```
SMTP_FAILURE
```

Host was not able to execute command.

```
SMTP_CANNOT_SEND_COMMAND
```

Windows Sockets library was not able to accept outgoing data.

```
SMTP_TIMED_OUT
```

Time out occurred while waiting for a response from host.

```
SMTP_IN_PROGRESS
```

Another operation is already in progress.

```
SMTP_ABORTED
```

Operation was aborted and connection has been closed.

If this function returns SMTP_FAILURE, then the **smtp_rply** function can be used to retrieve the error message sent by the host which describes the reason why the command failed.

2.17 smtp_send ()

Description Send send command.

```
#include <windows.h>
#include <d32-smtp.h>
```

```
int WINAPI smtp_send (hConn, from)
```

```
    GLOBALHANDLE hConn; Handle of SMTP structure from smtp_(x)helo
    LPSTR from;          Sender of mail
```

Remarks The **smtp_send** function informs the SMTP server that a message from the individual specified by *from* is about to be uploaded. If this function succeeds, then one or more recipients of the message must be identified with one or more calls to the **smtp_rcpt** function. The actual message including required headers must then be sent with the **smtp_data** function.

Unlike the **smtp_mail** function, the **smtp_send** function sends messages directly to the terminals of all specified recipients. If some recipients are not currently logged into the server or are not accepting terminal messages, then the **smtp_rcpt** function calls for those recipients will return error values.

The connection handle *hConn* must have been obtained with a call to **smtp_(x)helo**.

Return Value The **smtp_send** function either returns SMTP_SUCCESS or one of the following errors.

```
SMTP_INVALID_HANDLE
    Invalid connection handle was specified.
```

```
SMTP_FAILURE
    Host was not able to execute command.
```

```
SMTP_CANNOT_SEND_COMMAND
    Windows Sockets library was not able to accept outgoing data.
```

```
SMTP_TIMED_OUT
    Time out occurred while waiting for a response from host.
```

```
SMTP_IN_PROGRESS
    Another operation is already in progress.
```

```
SMTP_ABORTED
    Operation was aborted and connection has been closed.
```

If this function returns SMTP_FAILURE, then the **smtp_rply** function can be used to retrieve the error message sent by the host which describes the reason why the command failed.

2.18 smtp_soml ()

Description Send send or mail command.

```
#include <windows.h>
```

```
#include <d32-smtp.h>
```

```
int WINAPI smtp_soml (hConn, from)
```

```
    GLOBALHANDLE hConn; Handle of SMTP structure from smtp_(x)helo
```

```
    LPSTR from; Sender of mail
```

Remarks The **smtp_soml** function informs the SMTP server that a message from the individual specified by *from* is about to be uploaded. If this function succeeds, then one or more recipients of the message must be identified with one or more calls to the **smtp_rcpt** function. The actual message including required headers must then be sent with the **smtp_data** function.

Unlike the **smtp_mail** function, the **smtp_soml** function sends messages directly to the terminals of all specified recipients currently logged into the server. If some recipients are not currently logged into the server or are not accepting terminal messages, then the message is placed into the mail boxes of those recipients.

The connection handle *hConn* must have been obtained with a call to **smtp_(x)helo**.

Return Value The **smtp_soml** function either returns SMTP_SUCCESS or one of the following errors.

```
SMTP_INVALID_HANDLE
```

```
    Invalid connection handle was specified.
```

```
SMTP_FAILURE
```

```
    Host was not able to execute command.
```

```
SMTP_CANNOT_SEND_COMMAND
```

```
    Windows Sockets library was not able to accept outgoing data.
```

```
SMTP_TIMED_OUT
```

```
    Time out occurred while waiting for a response from host.
```

```
SMTP_IN_PROGRESS
```

```
    Another operation is already in progress.
```

```
SMTP_ABORTED
```

```
    Operation was aborted and connection has been closed.
```

If this function returns SMTP_FAILURE, then the **smtp_rply** function can be used to retrieve the error message sent by the host which describes the reason why the command failed.

2.19 smtp_turn ()

Description Send turn command.

```
#include <windows.h>
#include <d32-smtp.h>
```

```
int WINAPI smtp_turn (hConn)
```

GLOBALHANDLE *hConn*; Handle of SMTP structure from **smtp_(x)helo**

Remarks The **smtp_turn** function reverses the roles of the two machines communicating over the SMTP connection. The sending SMTP becomes the receiving SMTP and vice versa. This command is not normally used over a TCP/IP connection. The connection handle *hConn* must have been obtained with a call to **smtp_(x)helo**.

Return Value The **smtp_turn** function either returns SMTP_SUCCESS or one of the following errors.

SMTP_INVALID_HANDLE

Invalid connection handle was specified.

SMTP_FAILURE

Host was not able to execute command.

SMTP_CANNOT_SEND_COMMAND

Windows Sockets library was not able to accept outgoing data.

SMTP_TIMED_OUT

Time out occurred while waiting for a response from host.

SMTP_IN_PROGRESS

Another operation is already in progress.

SMTP_ABORTED

Operation was aborted and connection has been closed.

If this function returns SMTP_FAILURE, then the **smtp_rply** function can be used to retrieve the error message sent by the host which describes the reason why the command failed.

2.20 smtp_vrfy ()

Description Send verify command.
#include <windows.h>
#include <d32-smtp.h>

int WINAPI **smtp_vrfy** (*hConn*, *name*)

GLOBALHANDLE *hConn*; Handle of SMTP structure from **smtp_(x)helo**
LPSTR *name*; Name to verify

Remarks The **smtp_vrfy** function is used to verify the mail address specified by the *name* parameter. The response, which can be obtained with the **smtp_rply** command, might specify the complete user name associated with the mail address or should at least verify the mail address. The connection handle *hConn* must have been obtained with a call to **smtp_(x)helo**.

Return Value The **smtp_vrfy** function either returns SMTP_SUCCESS or one of the following errors.

SMTP_INVALID_HANDLE

Invalid connection handle was specified.

SMTP_FAILURE

Host was not able to execute command.

SMTP_CANNOT_SEND_COMMAND

Windows Sockets library was not able to accept outgoing data.

SMTP_TIMED_OUT

Time out occurred while waiting for a response from host.

SMTP_IN_PROGRESS

Another operation is already in progress.

SMTP_ABORTED

Operation was aborted and connection has been closed.

If this function returns SMTP_FAILURE, then the **smtp_rply** function can be used to retrieve the error message sent by the host which describes the reason why the command failed.

2.21 smtp_xhelo ()

Description Open asynchronous connection with SMTP server.

```
#include <windows.h>
#include <d32-smtp.h>
```

```
int WINAPI smtp_xhelo (host, local, conn, dbg, stat, lParam)
```

| | |
|---|--------------------------------------|
| LPSTR <i>host</i> ; | Name of SMTP server |
| LPSTR <i>local</i> ; | Local name |
| GLOBALHANDLE far * <i>conn</i> ; | Buffer for SMTP structure handle |
| FARPROC <i>dbg</i> ; | Application debug callback function |
| FARPROC <i>stat</i> ; | Application status callback function |
| DWORD <i>lParam</i> ; | Application supplied parameter |

Remarks The **smtp_xhelo** function establishes a TCP/IP connection with the specified host in an asynchronous manner. The *host* parameter can point either to a machine name (possibly including a domain name) or to an internet address in dotted decimal notation (e.g. "192.1.2.3"). The name pointed to by the *local* parameter is used to identify the local machine to the server and should specify the network name assigned to the local PC (possibly also including a domain name).

Unlike the **smtp_helo** function, the **smtp_xhelo** function does not actually establish the connection before returning to the application. Instead, it allocates required local resources and returns the handle of the connection control structure in the buffer pointed to by *conn*. Once the connection has been established, the status callback function specified with *stat* will be called. This function must be defined as follows.

```
int WINAPI (*stat) (hConn, result, lParam)
```

```
GLOBALHANDLE hConn;
int result;
DWORD lParam;
```

If the connection was established successfully, then the *result* parameter will contain the value SMTP_SUCCESS. If the connection could not be established, then the *result* parameter will contain one of the following errors.

SMTP_FAILURE

Remote error.

SMTP_UNKNOWN_HOST

Unable to resolve specified host name.

SMTP_CANNOT_ALLOC_SOCKET

Windows Sockets library could not allocate a socket.

SMTP_CANNOT_BIND_SOCKET

Windows Sockets library was not able to bind socket to local port.

SMTP_HOST_NOT_RESPONDING

Unable to connect to SMTP port on specified host.

SMTP_CANNOT_SEND_COMMAND

Windows Sockets library was not able to accept outgoing data.

SMTP_TIMED_OUT

Time out occurred while waiting for a response from host.

SMTP_ABORTED

Asynchronous connection attempt aborted with call to **smtp_xquit**.

Once the status callback function has been called with *result* set to SMTP_SUCCESS, then the handle placed into the buffer pointed to by *conn* must be used in all subsequent calls to the other SMTP functions for this connection. Any calls made with this handle before the connection has been established will fail with a return value of SMTP_IN_PROGRESS.

Multiple SMTP connections can be established simultaneously with the **smtp_xhelo** or **smtp_helo** calls. The Distinct SMTP-32 library supports multiple concurrent SMTP connections limited only by system resources.

During development it may be necessary to observe exactly what commands are sent to the SMTP server and what replies are generated. For this purpose, the application can specify the address of a debug callback function with *dbg*. This function will receive a copy of all network traffic and must be defined as follows.

```
int WINAPI (*dbg) (hConn, buf, len, lParam)

GLOBALHANDLE hConn;
char far *buf;
int len;
DWORD lParam;
```

The *dbg* parameter must be set to NULL if the application does not want to receive debug messages.

The *lParam* parameter can be set to any value that has meaning to the application such as an index into an array or a pointer to a structure. This parameter is passed back to the application as an argument in each callback function and allows the application to pass connection specific data to the callback function.

Return Value The **smtp_xhelo** function either returns SMTP_SUCCESS or one of the following errors.

SMTP_CANNOT_INITIALIZE
Support libraries or network drivers could not be initialized.

SMTP_CANNOT_INIT_WINSOCK
Windows Sockets transport library could not be initialized.

SMTP_OUT_OF_MEMORY
Not enough memory to allocate connection control structure.

2.22 smtp_xquit ()

Description Cancel asynchronous connect or operation in progress.

```
#include <windows.h>
```

```
#include <d32-smtp.h>
```

```
int WINAPI smtp_xquit (hConn)
```

 GLOBALHANDLE *hConn*; Handle of SMTP structure from **smtp_(x)helo**

Remarks The **smtp_xquit** function terminates an asynchronous connection attempt initiated with a call to **smtp_xhelo** or an SMTP operation in progress and frees all associated resources. The connection handle *hConn* must have been obtained with a call to **smtp_(x)helo** and will be invalid once the **smtp_xquit** function returns.

If **smtp_xquit** is called to cancel an asynchronous connect, then after the **smtp_xquit** call returns, the application must not terminate until the status callback function *stat* specified in the **smtp_xhelo** call has been called with the *result* parameter set to SMTP_ABORTED. If **smtp_xquit** is called to cancel an operation in progress, then after the **smtp_xquit** call returns, the application must not terminate until the canceled function returns.

The **smtp_xquit** function can be called after calling **smtp_xhelo** and before the status callback function *stat* specified in the **smtp_xhelo** call is called with the result of the connection attempt. The **smtp_xquit** function can also be called to cancel an **smtp_data**, **smtp_expn**, **smtp_help**, **smtp_mail**, **smtp_rcpt**, **smtp_rset**, **smtp_saml**, **smtp_send**, **smtp_soml**, **smtp_turn** or **smtp_vrfy** operation in progress.

Return Value The **smtp_xquit** function either returns SMTP_SUCCESS or one of the following errors.

SMTP_INVALID_HANDLE

 Invalid connection handle was specified.

SMTP_FAILURE

 No asynchronous connect in progress or asynchronous connect already aborted.