

TFTP-32 Library

**Trivial File Transfer Protocol
Dynamic Link Library
for Microsoft® Windows™**

Version 5.2

**Copyright © 1995 - 2003 by Distinct Corporation
All rights reserved**

Table of Contents

Table of Contents.....	iii
1 Overview	5
1.1 Introduction	5
1.2 Function Summary	6
2 Reference	7
2.1 tftp_cancel ()	7
2.2 tftp_close ()	8
2.3 tftp_get ().....	9
2.4 tftp_open ().....	11
2.5 tftp_option ()	12
2.6 tftp_put ().....	14

1 Overview

1.1 Introduction

The Distinct Trivial File Transfer Protocol (TFTP) library provides developers with an API to send and receive files from a remote host. The **tftp_open** function opens a TFTP connection on the default port. After a successful connection has been made the application can either get a file or put a file on the remote host by calling the API **tftp_get** or **tftp_put** respectively. Both **tftp_get** and **tftp_put** are blocking calls. Any get or put can be aborted by calling **tftp_cancel**. An application can also set any application-specific parameter using the **tftp_option** function. The application can specify a blocksize for data transfer by calling the **tftp_option** function before calling **tftp_get** or **tftp_put**. Lastly a TFTP connection is closed by making a call to **tftp_close**.

Various parameters used to control the behaviour of the Distinct TFTP library are stored under the following registry path.

HKEY_LOCAL_MACHINE\SOFTWARE\Distinct\Dills\TFTP32

The following entries specify the various Distinct TFTP library settings.

BlockSize REG_DWORD 8-65464

The default size of blocks for data transfer is 512, the application can edit the registry entry in order to specify a blocksize for data transfer.

1.2 Function Summary

tftp_cancel

Aborts the current operation

tftp_close

Closes the TFTP connection

tftp_get

Gets a file from the remote host

tftp_open

Opens a TFTP connection

tftp_options

Sets TFTP options

tftp_put

Puts a file on the remote host

2 Reference

2.1 tftp_cancel ()

Description

Cancels the current operation in progress.

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

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

```
int WINAPI tftp_cancel (void)
```

Remarks

The **tftp_cancel** function aborts a get or put operation that is in progress.

Return Value

The **tftp_cancel** function returns TRUE.

2.2 tftp_close ()

Description

Closes a TFTP connection.

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

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

```
int WINAPI tftp_close (htftp)
```

```
    GLOBALHANDLE *htftp;           Handle of the TFTP structure from tftp_open
```

Remarks

The **tftp_close** function terminates a TFTP connection and frees all the associated resources. The connection handle *htftp* must have been obtained with a call to **tftp_open** and will be invalid once the **tftp_close** function returns.

Return Value

The function returns TRUE if successful or it returns BAD_GLOBALHANDLE if a invalid connection handle was specified.

2.3 tftp_get ()

Description

Gets a file from the remote host.

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

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

```
long int WINAPI tftp_get (htftp, remote, transfer_mode, hostname, local, BytesCopied,
client_timeout, client_attempts, proc)
```

GLOBALHANDLE *htftp;	Handle of the tftp structure from tftp_open
char far *remote;	The name of the remote file.
int transfer_mode;	The mode in which the file has to be transferred.
char far *hostname	The name of the host from where the file has to be fetched.
char far *local	The local filename.
unsigned long far * BytesCopied	BytesCopied will have the number of bytes copied in the process of getting the file.
unsigned int client_timeout	The amount of time after which the operation will be aborted
unsigned int client_attempts	Number of resend attempts to be made if there is no response from the other side
FARPROC proc	Application callback function.

Remarks

The function **tftp_get** gets a file from the remote host in the ascii or binary mode. The *htftp* parameter must be a valid tftp structure handle obtained from **tftp_open**, the *remote* parameter must specify the name of the remote file that has to be fetched. The mode in which the file has to be transferred is determined by the *transfer_mode* parameter, the *transfer_mode* can be either TYPE_ASCII or TYPE_BINARY. The *hostname* parameter can either point to a machine name (possibly a domain name) or to an internet address in dotted decimal notation (192.1.2.3). The *local* parameter must contain the full path of the local file where the remote file will be copied. The *client_timeout* is the amount of time the client or server will wait for an acknowledgement before it times out. The *client_attempts* must specify the number of times data should be resend in case of no response from the other side. The *proc* is the application callback function which the TFTP library upcalls to return the number of bytes copied. Note that *proc* should be obtained by calling **MakeProcInstance** and specifying an exported function. The function must be defined as follows.

```
int WINAPI proc ( htftp, bytes, lParam)
```

```
GLOBALHANDLE htftp;
unsigned long bytes;
DWORD lParam;
```

Note that the application can specify a blocksize for the data transfer by calling the function **tftp_option**. The call to the function **tftp_option** must be made before calling **tftp_get**. The default packet size is 512 which is not very efficient if the file is large or on a LAN whose MTU may be 1500 octets or greater. The TFTP library when loading will first try to find the BlockSize key in the registry under **HKEY_LOCAL_MACHINE\SOFTWARE\DistinctDlls\TFTP32**, if it is not found the Library will use the default size 512. Once the library has been loaded the only way to change the blocksize is by calling the function **tftp_option** with OPT_SET_BLKSIZE as the option. (check tftp_option for more detail).

If a blocksize greater than 512 has been specified the TFTP library will first try to negotiate the blocksize with the TFTP server. If the server does not accept the blocksize specified by the

application then an error will be returned and if the server does not support blocksize negotiation then the TFTP library will ignore the blocksize request and start sending default size packets.

The size of the file can be determined before the actual file is received if the application turns on the file size check option by calling the **tftp_option** with the option parameter set to `OPT_CHECK_FILESIZE`. If the server supports the file size negotiation option then it would return the size of the file prior to sending the actual file. If there is not enough disk space to get the file the operation will be terminated with an error.

Return Value

If the function succeeds it will return the number of bytes copied. In case of any error it will return one of the following values.

`BAD_GLOBALHANDLE`
Invalid connection handle was specified.

`TFTP_GLOBALALLOC_FAILED`
Out of memory

`TFTP_ILLEGAL_REQUEST`
An illegal request was made

`TFTP_SOCKET_FAILURE`
Failed to get a socket

`TFTP_BIND_FAILED`
Bind failed

`TFTP_ILLEGAL_FILE_MODE`
Transfer mode specified was not `TYPE_ASCII` or `TYPE_BINARY`

`TFTP_FILE_ERROR`
Error in copying data to the local file

`TFTP_PARTIAL_FILE`
Only part of the file was fetched

`TFTP_SENDTO_ERR`
Windows socket library was not able to accept outgoing data

`TFTP_CANNOT_RESOLVE_HOSTNAME`
Hostname specified could not be resolved

`TFTP_CANNOT_NEGOTIATE_OPTION`
Could not negotiate the blocksize with TFTP server

2.5 tftp_option ()

Description

Sets or retrieves TFTP options.

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

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

Int FAR PASCAL tftp_option (*htftp, option, value*)

GLOBALHANDLE <i>htftp</i>	Handle of the tftp structure from tftp_open .
WORD <i>option</i> ;	The option to set.
DWORD <i>value</i> ;	The return value or the value to set.

Remarks

The function **tftp_option** allows the application to send values to (or retrieve values from) the TFTP library. Values are sent and retrieved through the *value* parameter. The use of this parameter is dependent upon the setting of the *option* parameter. Any one of the following options may be used for the *option* parameter.

Option	Meaning
OPT_SET_PARAM	Pass any value to the TFTP library.
OPT_GET_PARAM	Retrieve a previously-set value.
OPT_SET_BLKSIZE	Set the blocksize for data transfer
OPT_GET_BLKSIZE	Get the blocksize set for this particular connection
OPT_CHECK_FILESIZE	Check the file size
OPT_NOCHECK_FILESIZE	Do not check the file size
E	

The parameter *htftp* is the GLOBALHANDLE returned by **tftp_open**.

The application can pass any application specific data in the *value* parameter and set the option parameter to OPT_SET_PARAM. The value set by the application will be passed back in each of the callback functions. The application can retrieve this value by setting the option parameter to OPT_GET_PARAM and by passing a address in the *value* parameter.

The application can change the blocksize of data transfer by setting the *option* parameter to OPT_SET_BLKSIZE , it should be set before calling **tftp_get** or **tftp_put**. The *value* parameter must contain the desired blocksize. The valid range is between 8 and 65464 octets. Note that setting the blocksize will also overwrite the current registry entry (HKEY_LOCAL_MACHINE\SOFTWARE\Distinct\Dlls\TFTP32\BlockSize) with the blocksize passed in the *value* parameter. The current blocksize can be retrieved by setting the option parameter to OPT_GET_BLKSIZE and value parameter must be of type int far *.

There is an extended option in the TFTP protocol which allows the receiving side to find out the file size before the file is actually transferred. The application can turn on this option by calling the **tftp_option** function with the *option* parameter set to OPT_CHECK_FILESIZE. The application must set the *value* parameter to NULL. If the option is turned on then when requesting for a file (that is when doing a Get) the TFTP library will first request the size of the file from the server. If server supports this option it would send back the file size, the TFTP library will then check if there is enough disk space to get the file, if not an error will be returned. Similarly when putting a file on the server the TFTP library will send the file size along with the put request to the server. The server can then decide whether it can handle a file of the specified size. If the application does not want any file size negotiation to be made it should turn off this option by calling **tftp_option** with

the option parameter set to `TFTP_NOCHECK_FILESIZE` and the value parameter must be set to `NULL`.

Return Value

The **`tftp_option`** function returns `TRUE` if the function is successful. If a bad connection handle is passed it will return `BAD_GLOBALHANDLE` or `FALSE` in case of any other error.

2.6 tftp_put ()

Description

Puts a file on the remote host.

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

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

```
long int WINAPI tftp_put (htftp, remote, transfer_mode, hostname, local, BytesCopied,
client_timeout, client_attempts, proc)
```

GLOBALHANDLE *htftp;	Handle of the tftp structure from tftp_open
char far *remote;	The name of the remote file.
int transfer_mode;	The mode in which the file has to be transferred.
char far * hostname	The name of the host where the file has to be put.
char far * local	The local filename.
unsigned long far * BytesCopied	BytesCopied will have the number of bytes transferred in the process of putting the file.
unsigned int client_timeout	The amount of time after which the operation will be aborted
unsigned int client_attempts	Number of resend attempts to be made if there is no response from the other side
FARPROC proc	Application callback function.

Remarks

The function **tftp_put** puts a file on the remote host in ascii or binary mode. The *htftp* parameter must be a valid TFTP structure handle obtained from **tftp_open**, the *remote* parameter must specify the name of the remote file where the local file has to be put. The mode in which the file has to be transferred is determined by the *transfer_mode* parameter, the *transfer_mode* can be either TYPE_ASCII or TYPE_BINARY. The *hostname* parameter can either point to a machine name (possibly a domain name) or to an internet address in dotted decimal notation (192.1.2.3). The *local* parameter must contain the full path of the local file which has to be put on the remote host. The *client_timeout* is the amount of time the client or server will wait for an acknowledgement before it times out. The *client_attempts* must specify the number of times data should be resent in case of no response from the other side. The *proc* is the application callback function which the TFTP library upcalls to return the number of bytes copied. Note that *proc* should be obtained by calling **MakeProcInstance** and specifying an exported function. The function must be defined as follows.

```
int WINAPI proc ( htftp, bytes, lParam)
```

```
GLOBALHANDLE htftp;
unsigned long bytes;
DWORD lParam;
```

Note that the application can specify a blocksize for the data transfer by calling the function **tftp_option**. The call to the function **tftp_option** must be made before calling **tftp_put**. The default packet size is 512 which is not very efficient if the file is large or on a LAN whose MTU may be 1500 octets or greater. The TFTP library when loading will first try to find the BlockSize key in the registry under **HKEY_LOCAL_MACHINE\SOFTWARE\Distinct\Dills\TFTP32**, if it is not found the Library will use the default size 512. Once the library has been loaded the only way to change the blocksize is by calling the function **tftp_option** with OPT_SET_BLKSIZE as the option. (check **tftp_option** for more detail).

If a blocksize greater than 512 octets has been specified the TFTP library will first try to negotiate the blocksize with the TFTP server. If the server does not accept the blocksize specified by the

application then an error will be returned and if the server does not support blocksize negotiation then the TFTP library will ignore the blocksize request and start sending default size packets.

The TFTP library will send the size of the file along with the put request packet if the application turns on the file size check option by calling the **tftp_option** with the option parameter set to `OPT_CHECK_FILESIZE`. If the server supports the file size negotiation option then it would check if it can handle the file of the specified size, this option is very useful when the file is too big and there is not enough disk space on the server. An error will be returned if the server is unable to handle the file of the specified size.

Return Value

If the function succeeds it will return the number of bytes copied. In case of an error it will return one of the following values.

`BAD_GLOBALHANDLE`

Invalid connection handle was specified.

`TFTP_GLOBALALLOC_FAILED`

Out of memory

`TFTP_ILLEGAL_REQUEST`

An illegal request was made

`TFTP_SOCKET_FAILURE`

Failed to get a socket

`TFTP_BIND_FAILED`

Bind failed

`TFTP_ILLEGAL_FILE_MODE`

Transfer mode specified was not `TYPE_ASCII` or `TYPE_BINARY`

`TFTP_READ_ERROR`

Error in reading data from the local file

`TFTP_PARTIAL_FILE`

Only part of the file was fetched

`TFTP_SENDTO_ERR`

Windows socket library was not able to accept outgoing data

`TFTP_CANNOT_RESOLVE_HOSTNAME`

Hostname specified could not be resolved

`TFTP_CANNOT_NEGOTIATE_OPTION`

Could not negotiate the blocksize with TFTP server

