NAME

libssh2_sftp_write - write SFTP data

SYNOPSIS

```
#include <libssh2.h>
#include <libssh2_sftp.h>

ssize_t
libssh2_sftp_write(LIBSSH2_SFTP_HANDLE *handle,
                   const char *buffer,
                   size_t count);
```

DESCRIPTION

libssh2_sftp_write(3) writes a block of data to the SFTP server. This method is modeled after the POSIX write() function and uses the same calling semantics.

handle - SFTP file handle as returned by *libssh2_sftp_open_ex(3)*.

buffer - points to the data to send off.

count - Number of bytes from ’buffer’ to write. Note that it may not be possible to write all bytes as requested.

libssh2_sftp_handle(3) will use as much as possible of the buffer and put it into a single SFTP protocol packet. This means that to get maximum performance when sending larger files, you should try to always pass in at least 32K of data to this function.

WRITE AHEAD

Starting in libssh2 version 1.2.8, the default behavior of libssh2 is to create several smaller outgoing packets for all data you pass to this function and it will return a positive number as soon as the first packet is acknowledged from the server.

This has the effect that sometimes more data has been sent off but is not acked yet when this function returns, and when this function is subsequently called again to write more data, libssh2 will immediately figure out that the data is already received remotely.

In most normal situation this should not cause any problems, but it should be noted that if you have once called libssh2_sftp_write() with data and it returns short, you MUST still assume that the rest of the data might have been cached so you need to make sure you do not alter that data and think that the version you have in your next function invoke will be detected or used.

The reason for this funny behavior is that SFTP can only send 32K data in each packet and it gets all packets acked individually. This means we cannot use a simple serial approach if we want to reach high performance even on high latency connections. And we want that.

RETURN VALUE

Actual number of bytes written or negative on failure.

If used in non-blocking mode, it returns LIBSSH2_ERROR_EAGAIN when it would otherwise block. While LIBSSH2_ERROR_EAGAIN is a negative number, it is not really a failure per se.

If this function returns 0 (zero) it should not be considered an error, but that there was no error but yet no payload data got sent to the other end.

ERRORS

LIBSSH2_ERROR_ALLOC - An internal memory allocation call failed.

LIBSSH2_ERROR_SOCKET_SEND - Unable to send data on socket.

LIBSSH2_ERROR_SOCKET_TIMEOUT -

LIBSSH2_ERROR_SFTP_PROTOCOL - An invalid SFTP protocol response was received on the socket, or an SFTP operation caused an errorcode to be returned by the server.

SEE ALSO

libssh2_sftp_open_ex(3)