Secure Shell2

Introduction

Bindings to the » libssh2 library which provide access to resources (shell, remote exec, tunneling, file transfer) on a remote machine using a secure cryptographic transport.

Installing/Configuring

Requirements

No external libraries are needed to build this extension.

Installation

Windows binaries may be found at » http://snaps.php.net/. To install, download php_ssh2.dll to the folder specified by your php.ini file's *extension_dir* directive. Enable it by adding *extension=php_ssh2.dll* to your php.ini and restarting your web server.

```
extension_dir=c:/php5/exts/
extension=php_ssh2.dll
```

*Linux, BSD, and other *nix variants* can be compiled using the following steps:

- Download and install » OpenSSL. If you install OpenSSL via your distribution's packaging system be sure to install the development libraries as well. This will typically be a package named *openssl-dev*, *openssl_devel*, or some variation thereof.
- Download and install » libssh2. Typically this means executing the following command from the libssh2 source tree.*./configure && make all install*.
- Run the pear installer for PECL/ssh2: *pear install ssh2*
- Copy *ssh2.so* from the directory indicated by the build process to the location specified in your php.ini file under *extension_dir*.
- Add *extension=ssh2.so* to your php.ini
- Restart your web server to reload your php.ini settings.

Note

Development Versions

There are currently no *stable* versions of PECL/ssh2, to force installation of the *beta* version of PECL/ssh2 execute: *pear install ssh2- beta*

Tip

Compiling PECL/ssh2 without using the PEAR command

Rather than using *pear install ssh2* to automatically download and install PECL/ssh2, you may download the tarball from » PECL. From the root of the unpacked tarball, run: *phpize && ./configure --with-ssh2 && make* to generate *ssh2.so*. Once built, continue the installation from step 4 above.

Information for installing this PECL extension may be found in the manual chapter titled Installation of PECL extensions. Additional information such as new releases, downloads, source files, maintainer information, and a CHANGELOG, can be located here: » http://pecl.php.net/package/ssh2.

Note

You will need version 0.4 or greater of the libssh2 library (possibly higher, see release notes).

Runtime Configuration

This extension has no configuration directives defined in *php.ini*.

Resource Types

This extension has no resource types defined.

Predefined Constants

The constants below are defined by this extension, and will only be available when the extension has either been compiled into PHP or dynamically loaded at runtime.

SSH2_FINGERPRINT_MD5 (integer)
Flag to ssh2_fingerprint() requesting hostkey fingerprint as an MD5 hash.

SSH2_FINGERPRINT_SHA1 (integer)
Flag to ssh2_fingerprint() requesting hostkey fingerprint as an SHA1 hash.

SSH2_FINGERPRINT_HEX (integer)
Flag to ssh2_fingerprint() requesting hostkey fingerprint as a string of hexits.

SSH2_FINGERPRINT_RAW (integer)
Flag to ssh2_fingerprint() requesting hostkey fingerprint as a raw string of 8-bit characters.

SSH2_TERM_UNIT_CHARS (integer)
Flag to ssh2_shell() specifying that *`width`* and *`height`* are provided as character sizes.

SSH2_TERM_UNIT_PIXELS (integer)
Flag to ssh2_shell() specifying that *`width`* and *`height`* are provided in pixel units.

SSH2_DEFAULT_TERM_WIDTH (integer)
Default terminal width requested by ssh2_shell().

SSH2_DEFAULT_TERM_HEIGHT (integer)
Default terminal height requested by ssh2_shell().

SSH2_DEFAULT_TERM_UNIT (integer)
Default terminal units requested by ssh2_shell().

SSH2_STREAM_STDIO (integer)
Flag to ssh2_fetch_stream() requesting STDIO subchannel.

SSH2_STREAM_STDERR (integer)
Flag to ssh2_fetch_stream() requesting STDERR subchannel.

SSH2_DEFAULT_TERMINAL (string)
Default terminal type (e.g. vt102, ansi, xterm, vanilla) requested by ssh2_shell().

SSH2 Functions

ssh2_auth_hostbased_file

ssh2_auth_hostbased_file -- Authenticate using a public hostkey

Description

bool **ssh2_auth_hostbased_file** (resource `$session`, string `$username`, string `$hostname` , string `$pubkeyfile`, string `$privkeyfile` [, string `$passphrase` [, string `$local_username`]])

Authenticate using a public hostkey read from a file.

Parameters

`session`
An SSH connection link identifier, obtained from a call to ssh2_connect().

`username`

`hostname`

`pubkeyfile`

`privkeyfile`

`passphrase`
If *`privkeyfile`* is encrypted (which it should be), the passphrase must be provided.

`local_username`
If *`local_username`* is omitted, then the value for *`username`* will be used for it.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Examples

Example #1 - Authentication using a public hostkey

```
<?php
$connection = ssh2_connect('shell.example.com', 22,
```

```
array('hostkey'=>'ssh-rsa'));

if (ssh2_auth_hostbased_file($connection, 'remoteusername',
'myhost.example.com',
                            '/usr/local/etc/hostkey_rsa.pub',
                            '/usr/local/etc/hostkey_rsa', 'secret',
                            'localusername')) {
 echo "Public Key Hostbased Authentication Successful\n";
} else {
 die('Public Key Hostbased Authentication Failed');
}
?>
```

Notes

Note
ssh2_auth_hostbased_file() requires libssh2 >= 0.7 and PHP/SSH2 >= 0.7

ssh2_auth_none

ssh2_auth_none -- Authenticate as "none"

Description

mixed **ssh2_auth_none** (resource `$session`, string `$username`)

Attempt "none" authentication which usually will (and should) fail. As part of the failure, this function will return an array of accepted authentication methods.

Parameters

`session`
An SSH connection link identifier, obtained from a call to ssh2_connect().

`username`
Remote user name.

Return Values

Returns **TRUE** if the server doesc accept "none" as an authentication method, or an array of accepted authentication methods on failure.

Examples

Example #2 - Retreiving a list of authentication methods

```
<?php
$connection = ssh2_connect('shell.example.com', 22);

$auth_methods = ssh2_auth_none($connection, 'user');

if (in_array('password', $auth_methods)) {
 echo "Server supports password based authentication\n";
}
?>
```

ssh2_auth_password

ssh2_auth_password -- Authenticate over SSH using a plain password

Description

bool **ssh2_auth_password** (resource `$session`, string `$username`, string `$password`)

Authenticate over SSH using a plain password

Parameters

`session`
An SSH connection link identifier, obtained from a call to ssh2_connect().

`username`
Remote user name.

`password`
Password for *`username`*

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Examples

Example #3 - Authenticating with a password

```
<?php
$connection = ssh2_connect('shell.example.com', 22);

if (ssh2_auth_password($connection, 'username', 'secret')) {
 echo "Authentication Successful!\n";
} else {
 die('Authentication Failed...');
}
?>
```

ssh2_auth_pubkey_file

ssh2_auth_pubkey_file -- Authenticate using a public key

Description

bool **ssh2_auth_pubkey_file** (resource `$session`, string `$username`, string `$pubkeyfile`, string `$privkeyfile` [, string `$passphrase`])

Authenticate using a public key read from a file.

Parameters

`session`
An SSH connection link identifier, obtained from a call to ssh2_connect().

`username`

`pubkeyfile`

`privkeyfile`

`passphrase`
If *`privkeyfile`* is encrypted (which it should be), the *`passphrase`* must be provided.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Examples

Example #4 - Authentication using a public key

```
<?php
$connection = ssh2_connect('shell.example.com', 22,
array('hostkey'=>'ssh-rsa'));

if (ssh2_auth_pubkey_file($connection, 'username',
                          '/home/username/.ssh/id_rsa.pub',
                          '/home/username/.ssh/id_rsa', 'secret')) {
 echo "Public Key Authentication Successful\n";
} else {
 die('Public Key Authentication Failed');
```

```
}
?>
```

ssh2_connect

ssh2_connect -- Connect to an SSH server

Description

resource **ssh2_connect** (string `$host` [, int `$port` [, array `$methods` [, array `$callbacks`]]])

Establish a connection to a remote SSH server.

Once connected, the client should verify the server's hostkey using ssh2_fingerprint(), then authenticate using either password or public key.

Parameters

`host`

`port`

`methods`

`methods` may be an associative array with up to four parameters as described below.

`methods` **may be an associative array with any or all of the following parameters.**

Index	Meaning	Supported Values*
kex	List of key exchange methods to advertise, comma separated in order of preference.	*diffie-hellman-group1-sha1*, *diffie-hellman-group14-sha1*, and *diffie-hellman-group-exchange-sha1*
hostkey	List of hostkey methods to advertise, come separated in order of preference.	*ssh-rsa* and *ssh-dss*
client_to_server	Associative array containing crypt, compression, and message authentication code (MAC) method preferences for messages sent from client to server.	
server_to_client	Associative array containing crypt, compression, and	

	message authentication code (MAC) method preferences for messages sent from client to server.	

* - Supported Values are dependent on methods supported by underlying library. See » libssh2 documentation for additional information.

`client_to_server` **and** `server_to_client` **may be an associative array with any or all of the following parameters.**

Index	Meaning	Supported Values*
crypt	List of crypto methods to advertise, comma separated in order of preference.	*rijndael-cbc@lysator.liu.se*, *aes256-cbc*, *aes192-cbc*, *aes128-cbc*, *3des-cbc*, *blowfish-cbc*, *cast128-cbc*, *arcfour*, and *none***
comp	List of compression methods to advertise, comma separated in order of preference.	*zlib* and *none*
mac	List of MAC methods to advertise, come separated in order of preference.	*hmac-sha1*, *hmac-sha1-96*, *hmac-ripemd160*, *hmac-ripemd160@openssh.com*, and *none***

Note
Crypt and MAC method " *none* **"** For security reasons, *none* is disabled by the underlying » libssh2 library unless explicitly enabled during build time by using the appropriate ./configure options. See documentation for the underlying library for more information.

`callbacks`

`callbacks` may be an associative array with any or all of the following parameters.

Callbacks parameters

Index	Meaning	Prototype
ignore	Name of function to call when an **SSH2_MSG_IGNORE** packet is received	void ignore_cb($message)

debug	Name of function to call when an **SSH2_MSG_DEBUG** packet is received	void debug_cb($message, $language, $always_display)
macerror	Name of function to call when a packet is received but the message authentication code failed. If the callback returns **TRUE**, the mismatch will be ignored, otherwise the connection will be terminated.	bool macerror_cb($packet)
disconnect	Name of function to call when an **SSH2_MSG_DISCONNECT** packet is received	void disconnect_cb($reason, $message, $language)

Return Values

Returns a resource on success, or **FALSE** on error.

Examples

Example #5 - ssh2_connect() **example**

Open a connection forcing 3des-cbc when sending packets, any strength aes cipher when receiving packets, no compression in either direction, and Group1 key exchange.

```
<?php
/* Notify the user if the server terminates the connection */
function my_ssh_disconnect($reason, $message, $language) {
 printf("Server disconnected with reason code [%d] and message: %s\n",
        $reason, $message);
}

$methods = array(
 'kex' => 'diffie-hellman-group1-sha1',
 'client_to_server' => array(
   'crypt' => '3des-cbc',
   'comp' => 'none'),
 'server_to_client' => array(
   'crypt' => 'aes256-cbc,aes192-cbc,aes128-cbc',
   'comp' => 'none'));

$callbacks = array('disconnect' => 'my_ssh_disconnect');

$connection = ssh2_connect('shell.example.com', 22, $methods, $callbacks);
```

```
if (!$connection) die('Connection failed');
?>
```

See Also

- ssh2_fingerprint()
- ssh2_auth_none()
- ssh2_auth_password()
- ssh2_auth_pubkey_file()

ssh2_exec

ssh2_exec -- Execute a command on a remote server

Description

resource **ssh2_exec** (resource `$session`, string `$command` [, string `$pty` [, array `$env` [, int `$width` [, int `$height` [, int `$width_height_type`]]]]])

Execute a command at the remote end and allocate a channel for it.

Parameters

`session`
An SSH connection link identifier, obtained from a call to ssh2_connect().

`command`

`pty`

`env`
`env` may be passed as an associative array of name/value pairs to set in the target environment.

`width`
Width of the virtual terminal.

`height`
Height of the virtual terminal.

`width_height_type`
`width_height_type` should be one of **SSH2_TERM_UNIT_CHARS** or **SSH2_TERM_UNIT_PIXELS**.

Return Values

Returns a stream on success or **FALSE** on failure.

Examples

Example #6 - Executing a command

```
<?php
```

```
$connection = ssh2_connect('shell.example.com', 22);
ssh2_auth_password($connection, 'username', 'password');

$stream = ssh2_exec($connection, '/usr/local/bin/php -i');
?>
```

See Also

- ssh2_connect()
- ssh2_shell()
- ssh2_tunnel()

ssh2_fetch_stream

ssh2_fetch_stream -- Fetch an extended data stream

Description

resource **ssh2_fetch_stream** (resource `$channel`, int `$streamid`)

Fetches an alternate substream associated with an SSH2 channel stream. The SSH2 protocol currently defines only one substream, STDERR, which has a substream ID of **SSH2_STREAM_STDERR** (defined as 1).

Parameters

`channel`

`streamid`
An SSH2 channel stream.

Return Values

Returns the requested stream resource.

Examples

Example #7 - Opening a shell and retrieving the stderr stream associated with it

```
<?php
$connection = ssh2_connect('shell.example.com', 22);
ssh2_auth_password($connection, 'username', 'password');

$stdio_stream = ssh2_shell($connection);
$stderr_stream = ssh2_fetch_stream($stdio_stream, SSH2_STREAM_STDERR);
?>
```

See Also

- ssh2_shell()
- ssh2_exec()
- ssh2_connect()

ssh2_fingerprint

ssh2_fingerprint -- Retrieve fingerprint of remote server

Description

string **ssh2_fingerprint** (resource `$session` [, int `$flags`])

Returns a server hostkey hash from an active session.

Parameters

`session`
An SSH connection link identifier, obtained from a call to ssh2_connect().

`flags`
`flags` may be either of **SSH2_FINGERPRINT_MD5** or **SSH2_FINGERPRINT_SHA1** logically ORed with **SSH2_FINGERPRINT_HEX** or **SSH2_FINGERPRINT_RAW**. Defaults to **SSH2_FINGERPRINT_MD5** | **SSH2_FINGERPRINT_HEX**.

Return Values

Returns the hostkey hash as a string.

Examples

Example #8 - Checking the fingerprint against a known value

```
<?php
$known_host = '6F89C2F0A719B30CC38ABDF90755F2E4';

$connection = ssh2_connect('shell.example.com', 22);

$fingerprint = ssh2_fingerprint($connection,
              SSH2_FINGERPRINT_MD5 | SSH2_FINGERPRINT_HEX);

if ($fingerprint != $known_host) {
 die("HOSTKEY MISMATCH!\n" .
     "Possible Man-In-The-Middle Attack?");
}
?>
```

ssh2_methods_negotiated

ssh2_methods_negotiated -- Return list of negotiated methods

Description

array **ssh2_methods_negotiated** (resource `$session`)

Returns list of negotiated methods.

Parameters

`session`
An SSH connection link identifier, obtained from a call to ssh2_connect().

Return Values

Examples

Example #9 - Determining what methods were negotiated

```
<?php
$connection = ssh2_connect('shell.example.com', 22);
$methods = ssh2_methods_negotiated($connection);

echo "Encryption keys were negotiated using: {$methods['kex']}\n";
echo "Server identified using an {$methods['hostkey']} with ";
echo "fingerprint: " . ssh2_fingerprint($connection) . "\n";

echo "Client to Server packets will use methods:\n";
echo "\tCrypt: {$methods['client_to_server']['crypt']}\n";
echo "\tComp: {$methods['client_to_server']['comp']}\n";
echo "\tMAC: {$methods['client_to_server']['mac']}\n";

echo "Server to Client packets will use methods:\n";
echo "\tCrypt: {$methods['server_to_client']['crypt']}\n";
echo "\tComp: {$methods['server_to_client']['comp']}\n";
echo "\tMAC: {$methods['server_to_client']['mac']}\n";

?>
```

See Also

- ssh2_connect()

ssh2_publickey_add

ssh2_publickey_add -- Add an authorized publickey

Description

bool **ssh2_publickey_add** (resource `$pkey`, string `$algoname`, string `$blob` [, bool `$overwrite` [, array `$attributes`]])

Note
The public key subsystem is used for managing public keys on a server to which the client is *already* authenticated. To authenticate to a remote system using public key authentication, use the ssh2_auth_pubkey_file() function instead.

Parameters

`pkey`
: Publickey Subsystem resource created by ssh2_publickey_init().

`algoname`
: Publickey algorithm (e.g.): ssh-dss, ssh-rsa

`blob`
: Publickey blob as raw binary data

`overwrite`
: If the specified key already exists, should it be overwritten?

`attributes`
: Associative array of attributes to assign to this public key. Refer to ietf-secsh-publickey-subsystem for a list of supported attributes. To mark an attribute as mandatory, precede its name with an asterisk. If the server is unable to support an attribute marked mandatory, it will abort the add process.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Examples

Example #10 - Adding a publickey with ssh2_publickey_add()

```
<?php
$ssh2 = ssh2_connect('shell.example.com', 22);
ssh2_auth_password($ssh2, 'jdoe', 'password');
$pkey = ssh2_publickey_init($ssh2);

$keyblob = base64_decode('
AAAAB3NzaC1yc2EAAAABIwAAAIEA5HVt6VqSGd5PTrLRdjNONxXH1tVFGn0
Bd26BF0aCP9qyJRlvdJ3j4WBeX4ZmrveGrjMgkseSYc4xZ26sDHwfL351xj
zaLpipu\BGRrw17mWVBhuCExo476ri5tQFzbTc54VEHYckxQ16CjSTibI5X
69GmnYC9PNqEYq/1TP+HF10=');

ssh2_publickey_add($ssh2, 'ssh-rsa', $keyblob, false, array('comment'=>"John's
Key"));
?>
```

See Also

- ssh2_publickey_init()
- ssh2_publickey_remove()
- ssh2_publickey_list()

ssh2_publickey_init

ssh2_publickey_init -- Initialize Publickey subsystem

Description

resource **ssh2_publickey_init** (resource `$session`)

Request the Publickey subsystem from an already connected SSH2 server.

The publickey subsystem allows an already connected and authenticated client to manage the list of authorized public keys stored on the target server in an implementation agnostic manner. If the remote server does not support the publickey subsystem, the ssh2_publickey_init() function will return **FALSE**.

Parameters

`session`

Return Values

Returns an *SSH2 Publickey Subsystem* resource for use with all other ssh2_publickey_*() methods, or **FALSE** on failure.

Notes

Note
The public key subsystem is used for managing public keys on a server to which the client is *already* authenticated. To authenticate to a remote system using public key authentication, use the ssh2_auth_pubkey_file() function instead.

See Also

- ssh2_publickey_add()
- ssh2_publickey_remove()
- ssh2_publickey_list()

ssh2_publickey_list

ssh2_publickey_list -- List currently authorized publickeys

Description

array **ssh2_publickey_list** (resource `$pkey`)

List currently authorized publickeys.

Parameters

pkey
Publickey Subsystem resource

Return Values

Returns a numerically indexed array of keys, each of which is an associative array containing: name, blob, and attrs elements.

Publickey elements

Array Key	Meaning
name	Name of algorithm used by this publickey, for example: *ssh-dss* or *ssh-rsa*.
blob	Publickey blob as raw binary data.
attrs	Attributes assigned to this publickey. The most common attribute, and the only one supported by publickey version 1 servers, is *comment*, which may be any freeform string.

Examples

Example #11 - Listing authorized keys with ssh2_publickey_list()

```
<?php
$ssh2 = ssh2_connect('shell.example.com', 22);
ssh2_auth_password($ssh2, 'jdoe', 'secret');
```

```
$pkey = ssh2_publickey_init($ssh2);

$list = ssh2_publickey_list($pkey);

foreach($list as $key) {
 echo "Key: {$key['name']}\n";
 echo "Blob: " . chunk_split(base64_encode($key['blob']), 40, "\n") . "\n";
 echo "Comment: {$key['attrs']['comment']}\n\n";
}
?>
```

The above example will output:

```
Key: ssh-rsa
Blob: AAAAB3NzaC1yc2EAAAABIwAAAIEA5HVt6VqSGd5P
TrLRdjNONxXH1tVFGn0Bd26BF0aCP9qyJRlvdJ3j
4WBeX4ZmrveGrjMgkseSYc4xZ26sDHwfL351xjza
Lpipu\BGRrw17mWVBhuCExo476ri5tQFzbTc54VE
HYckxQ16CjSTibI5X69GmnYC9PNqEYq/1TP+HF10
Comment: John's Key

Key: ssh-rsa
Blob: AAAAB3NzaHVt6VqSGd5C1yc2EAAAABIwA232dnJA
AIEA5HVt6VqSGd5PTrLRdjNONxX/1TP+HF1HVt6V
qSGd50H1tVFGn0BB3NzaC1yc2EAd26BF0aCP9qyJ
RlvdJ3j4WBeX4ZmrveGrjMgkseSYc4xZ26HVt6Vq
SGd5sDHwfL351xjzaLpipu\BGB3NzaC1yc2EA/1T
Comment: Alice's Key
```

Notes

Note
The public key subsystem is used for managing public keys on a server to which the client is *already* authenticated. To authenticate to a remote system using public key authentication, use the ssh2_auth_pubkey_file() function instead.

See Also

- ssh2_publickey_init()
- ssh2_publickey_add()
- ssh2_publickey_remove()

ssh2_publickey_remove

ssh2_publickey_remove -- Remove an authorized publickey

Description

bool **ssh2_publickey_remove** (resource `$pkey`, string `$algoname`, string `$blob`)

Removes an authorized publickey.

Parameters

`pkey`
Publickey Subsystem Resource

`algoname`
Publickey algorithm (e.g.): ssh-dss, ssh-rsa

`blob`
Publickey blob as raw binary data

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Notes

Note
The public key subsystem is used for managing public keys on a server to which the client is *already* authenticated. To authenticate to a remote system using public key authentication, use the ssh2_auth_pubkey_file() function instead.

See Also

- ssh2_publickey_init()
- ssh2_publickey_add()
- ssh2_publickey_list()

ssh2_scp_recv

ssh2_scp_recv -- Request a file via SCP

Description

bool **ssh2_scp_recv** (resource `$session`, string `$remote_file`, string `$local_file`)

Copy a file from the remote server to the local filesystem using the SCP protocol.

Parameters

`session`
An SSH connection link identifier, obtained from a call to ssh2_connect().

`remote_file`
Path to the remote file.

`local_file`
Path to the local file.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Examples

Example #12 - Downloading a file via SCP

```
<?php
$connection = ssh2_connect('shell.example.com', 22);
ssh2_auth_password($connection, 'username', 'password');

ssh2_scp_recv($connection, '/remote/filename', '/local/filename');
?>
```

See Also

- ssh2_scp_send()
- copy()

ssh2_scp_send

ssh2_scp_send -- Send a file via SCP

Description

bool **ssh2_scp_send** (resource `$session`, string `$local_file`, string `$remote_file` [, int `$create_mode`])

Copy a file from the local filesystem to the remote server using the SCP protocol.

Parameters

`session`
An SSH connection link identifier, obtained from a call to ssh2_connect().

`local_file`
Path to the local file.

`remote_file`
Path to the remote file.

`create_mode`
The file will be created with the mode specified by *`create_mode`*.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Examples

Example #13 - Uploading a file via SCP

```
<?php
$connection = ssh2_connect('shell.example.com', 22);
ssh2_auth_password($connection, 'username', 'password');

ssh2_scp_send($connection, '/local/filename', '/remote/filename', 0644);
?>
```

See Also

- ssh2_scp_recv()
- copy()

ssh2_sftp_lstat

ssh2_sftp_lstat -- Stat a symbolic link

Description

array **ssh2_sftp_lstat** (resource `$sftp`, string `$path`)

Stats a symbolic link on the remote filesystem *without* following the link.

This function is similar to using the lstat() function with the ssh2.sftp:// wrapper in PHP5 and returns the same values.

Parameters

`sftp`

`path`
Path to the remote symbolic link.

Return Values

See the documentation for stat() for details on the values which may be returned.

Examples

Example #14 - Stating a symbolic link via SFTP

```
<?php
$connection = ssh2_connect('shell.example.com', 22);
ssh2_auth_password($connection, 'username', 'password');

$sftp = ssh2_sftp($connection);
$statinfo = ssh2_sftp_lstat($sftp, '/path/to/symlink');

$filesize = $statinfo['size'];
$group = $statinfo['gid'];
$owner = $statinfo['uid'];
$atime = $statinfo['atime'];
$mtime = $statinfo['mtime'];
$mode = $statinfo['mode'];
?>
```

See Also

- ssh2_sftp_stat()
- lstat()
- stat()

ssh2_sftp_mkdir

ssh2_sftp_mkdir -- Create a directory

Description

bool **ssh2_sftp_mkdir** (resource `$sftp`, string `$dirname` [, int `$mode` [, bool `$recursive`]])

Creates a directory on the remote file server with permissions set to *`mode`*.

This function is similar to using mkdir() with the ssh2.sftp:// wrapper.

Parameters

`sftp`
: An SSH2 SFTP resource opened by ssh2_sftp().

`dirname`
: Path of the new directory.

`mode`
: Permissions on the new directory.

`recursive`
: If *`recursive`* is **TRUE** any parent directories required for *`dirname`* will be automatically created as well.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Examples

Example #15 - Creating a directory on a remote server

```
<?php
$connection = ssh2_connect('shell.example.com', 22);
ssh2_auth_password($connection, 'username', 'password');
$sftp = ssh2_sftp($connection);

ssh2_sftp_mkdir($sftp, '/home/username/newdir');
/* Or:  mkdir("ssh2.sftp://$sftp/home/username/newdir"); */
?>
```

See Also

- mkdir()
- ssh2_sftp_rmdir()

ssh2_sftp_readlink

ssh2_sftp_readlink -- Return the target of a symbolic link

Description

string **ssh2_sftp_readlink** (resource `$sftp`, string `$link`)

Returns the target of a symbolic link.

Parameters

`sftp`
An SSH2 SFTP resource opened by ssh2_sftp().

`link`
Path of the symbolic link.

Return Values

Returns the target of the symbolic *`link`*.

Examples

Example #16 - Reading a symbolic link

```
<?php
$connection = ssh2_connect('shell.example.com', 22);
ssh2_auth_password($connection, 'username', 'password');
$sftp = ssh2_sftp($connection);

$target = ssh2_sftp_readlink($sftp, '/tmp/mysql.sock');
/* $target is now (e.g.): '/var/run/mysql.sock' */
?>
```

See Also

- readlink()
- ssh2_sftp_symlink()

ssh2_sftp_realpath

ssh2_sftp_realpath -- Resolve the realpath of a provided path string

Description

string **ssh2_sftp_realpath** (resource `$sftp`, string `$filename`)

Translates *`filename`* into the effective real path on the remote filesystem.

Parameters

`sftp`
An SSH2 SFTP resource opened by ssh2_sftp().

`filename`

Return Values

Returns the real path as a string.

Examples

Example #17 - Resolving a pathname

```
<?php
$connection = ssh2_connect('shell.example.com', 22);
ssh2_auth_password($connection, 'username', 'password');
$sftp = ssh2_sftp($connection);

$realpath = ssh2_sftp_realpath($sftp,
'/home/username/../../../..///./usr/../etc/passwd');
/* $realpath is now: '/etc/passwd' */
?>
```

See Also

- realpath()
- ssh2_sftp_symlink()
- ssh2_sftp_readlink()

ssh2_sftp_rename

ssh2_sftp_rename -- Rename a remote file

Description

bool **ssh2_sftp_rename** (resource `$sftp`, string `$from`, string `$to`)

Renames a file on the remote filesystem.

Parameters

`sftp`
An SSH2 SFTP resource opened by ssh2_sftp().

`from`
The current file that is being renamed.

`to`
The new file name that replaces *`from`*.

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Examples

Example #18 - Renaming a file via sftp

```
<?php
$connection = ssh2_connect('shell.example.com', 22);
ssh2_auth_password($connection, 'username', 'password');
$sftp = ssh2_sftp($connection);

ssh2_sftp_rename($sftp, '/home/username/oldname', '/home/username/newname');
?>
```

See Also

- rename()

ssh2_sftp_rmdir

ssh2_sftp_rmdir -- Remove a directory

Description

bool **ssh2_sftp_rmdir** (resource `$sftp`, string `$dirname`)

Removes a directory from the remote file server.

This function is similar to using rmdir() with the ssh2.sftp:// wrapper.

Parameters

`sftp`
An SSH2 SFTP resource opened by ssh2_sftp().

`dirname`

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Examples

Example #19 - Removing a directory on a remote server

```
<?php
$connection = ssh2_connect('shell.example.com', 22);
ssh2_auth_password($connection, 'username', 'password');
$sftp = ssh2_sftp($connection);

ssh2_sftp_rmdir($sftp, '/home/username/deltodel');
/* Or:  rmdir("ssh2.sftp://$sftp/home/username/dirtodel"); */
?>
```

See Also

- rmdir()
- ssh2_sftp_mkdir()

ssh2_sftp_stat

ssh2_sftp_stat -- Stat a file on a remote filesystem

Description

array **ssh2_sftp_stat** (resource `$sftp`, string `$path`)

Stats a file on the remote filesystem following any symbolic links.

This function is similar to using the stat() function with the ssh2.sftp:// wrapper in PHP5 and returns the same values.

Parameters

`sftp`
An SSH2 SFTP resource opened by ssh2_sftp().

`path`

Return Values

See the documentation for stat() for details on the values which may be returned.

Examples

Example #20 - Stating a file via SFTP

```
<?php
$connection = ssh2_connect('shell.example.com', 22);
ssh2_auth_password($connection, 'username', 'password');

$sftp = ssh2_sftp($connection);
$statinfo = ssh2_sftp_stat($sftp, '/path/to/file');

$filesize = $statinfo['size'];
$group = $statinfo['gid'];
$owner = $statinfo['uid'];
$atime = $statinfo['atime'];
$mtime = $statinfo['mtime'];
$mode = $statinfo['mode'];
?>
```

See Also

- ssh2_sftp_lstat()
- lstat()
- stat()

ssh2_sftp_symlink

ssh2_sftp_symlink -- Create a symlink

Description

bool **ssh2_sftp_symlink** (resource `$sftp`, string `$target`, string `$link`)

Creates a symbolic link named *`link`* on the remote filesystem pointing to *`target`*.

Parameters

`sftp`
An SSH2 SFTP resource opened by ssh2_sftp().

`target`
Target of the symbolic link.

`link`

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Examples

Example #21 - Creating a symbolic link

```
<?php
$connection = ssh2_connect('shell.example.com', 22);
ssh2_auth_password($connection, 'username', 'password');
$sftp = ssh2_sftp($connection);

ssh2_sftp_symlink($sftp, '/var/run/mysql.sock', '/tmp/mysql.sock');
?>
```

See Also

- ssh2_sftp_readlink()
- symlink()

ssh2_sftp_unlink

ssh2_sftp_unlink -- Delete a file

Description

bool **ssh2_sftp_unlink** (resource `$sftp`, string `$filename`)

Deletes a file on the remote filesystem.

Parameters

`sftp`
An SSH2 SFTP resource opened by ssh2_sftp().

`filename`

Return Values

Returns **TRUE** on success or **FALSE** on failure.

Examples

Example #22 - Deleting a file

```
<?php
$connection = ssh2_connect('shell.example.com', 22);
ssh2_auth_password($connection, 'username', 'password');
$sftp = ssh2_sftp($connection);

ssh2_sftp_unlink($sftp, '/home/username/stale_file');
?>
```

See Also

- unlink()

ssh2_sftp

ssh2_sftp -- Initialize SFTP subsystem

Description

resource **ssh2_sftp** (resource `$session`)

Request the SFTP subsystem from an already connected SSH2 server.

Parameters

`session`
An SSH connection link identifier, obtained from a call to ssh2_connect().

Return Values

This method returns an *SSH2 SFTP* resource for use with all other ssh2_sftp_*() methods and the ssh2.sftp:// fopen wrapper.

Examples

Example #23 - Opening a file via SFTP

```
<?php
$connection = ssh2_connect('shell.example.com', 22);
ssh2_auth_password($connection, 'username', 'password');

$sftp = ssh2_sftp($connection);

$stream = fopen("ssh2.sftp://$sftp/path/to/file", 'r');
?>
```

See Also

- ssh2_scp_recv()
- ssh2_scp_send()

ssh2_shell

ssh2_shell -- Request an interactive shell

Description

resource **ssh2_shell** (resource `$session` [, string `$term_type` [, array `$env` [, int `$width` [, int `$height` [, int `$width_height_type`]]]]])

Open a shell at the remote end and allocate a stream for it.

Parameters

`session`
An SSH connection link identifier, obtained from a call to ssh2_connect().

`term_type`
`term_type` should correspond to one of the entries in the target system's */etc/termcap* file and defaults to *vanilla*.

`env`
`env` may be passed as an associative array of name/value pairs to set in the target environment.

`width`
Width of the virtual terminal.

`height`
Height of the virtual terminal.

`width_height_type`
`width_height_type` should be one of **SSH2_TERM_UNIT_CHARS** or **SSH2_TERM_UNIT_PIXELS**.

Return Values

Examples

Example #24 - Executing a command

```
<?php
$connection = ssh2_connect('shell.example.com', 22);
ssh2_auth_password($connection, 'username', 'password');
```

```
$stream = ssh2_shell($connection, 'vt102', null, 80, 24,
SSH2_TERM_UNIT_CHARS);
?>
```

See Also

- ssh2_exec()
- ssh2_tunnel()
- ssh2_fetch_stream()

ssh2_tunnel

ssh2_tunnel -- Open a tunnel through a remote server

Description

resource **ssh2_tunnel** (resource `$session`, string `$host`, int `$port`)

Open a socket stream to an arbitrary host/port by way of the currently connected SSH server.

Parameters

`session`
An SSH connection link identifier, obtained from a call to ssh2_connect().

`host`

`port`

Return Values

Examples

Example #25 - Opening a tunnel to an arbitrary host

```
<?php
$connection = ssh2_connect('shell.example.com', 22);
ssh2_auth_pubkey_file($connection, 'username', 'id_dsa.pub', 'id_dsa');

$tunnel = ssh2_tunnel($connection, '10.0.0.101', 12345);
?>
```

See Also

- ssh2_connect()
- fsockopen()