



PeerSec MatrixSSH™ Dev Guide

Overview	1
API Documentation	2
matrixSshOpen()	2
matrixSshClose()	2
matrixSshReadKeysMem()	2
matrixSshFreeKeys()	3
matrixSshNewSession()	3
matrixSshDeleteSession()	5
matrixSshRegisterDataCallback()	5
matrixSshQueueServerIdent()	7
matrixSshQueueOutgoingData()	8
matrixSshQueueCloseSession()	9
matrixSshDecode()	9
matrixSshGetReadBuffer()	10
matrixSshUpdateWriteBuffer()	11
matrixSshGetWriteBuffer()	11
matrixSshHandleTimeouts()	12

API Version: MATRIXSSH_2_0



Overview

MatrixSSH

MatrixSSH is an embedded C code library used to create SSH 2.0 server applications. The cryptography code is implemented using the MatrixSSL library. MatrixSSH supports password authentication and RSA public key authentication methods. The supported symmetric ciphers are AES and 3DES and the supported Message Authentication hash algorithms are HMAC-SHA1 and HMAC-MD5.

The library APIs operate at the IO buffer level and do not assume any particular transport mechanism. However, a sample application using POSIX sockets is provided in the MatrixSSH package and may be used as a framework for release products.

The example server supports one channel per SSH connection.

About PeerSec Networks

PeerSec Networks has had security and encryption products on the market for over seven years. With nearly 100 customers worldwide and millions of devices secured, PeerSec is a global leader in embedded security. PeerSec Networks provides security solutions with fully owned Intellectual Property and a dedicated engineering staff for rapid support and custom enhancements.



API Documentation

matrixSshOpen()

Prototype

```
int matrixSshOpen(void);
```

Parameters

Return Value	Output	0 on success, < 0 on failure
--------------	--------	------------------------------

This initialization function must be called once for each server process. The internal library structures are allocated and setup at this time.

matrixSshClose()

Prototype

```
void matrixSshClose(void);
```

Parameters

none

This shutdown function must be called at the exit of each server process. The internal library resources are freed at this time.

matrixSshReadKeysMem()

Prototype

```
int matrixSshReadKeysMem(sshRsaKey_t **keyStruct, const unsigned char *key, int keyLen, int keyType);
```

Parameters

keyStruct	Output	Successful return will populate keyStruct for input to subsequent calls to matrixSshNewSession.
-----------	--------	---



key	Input	ASN.1 DER-encoded private key
keyLen	Input	Length in bytes of the key
keyType	Input	Must be MATRIXSSH_SIGNKEY_RSA
Return Value	Output	0 on success, < 0 on failure

This function is used to parse and load the private key of the server into the `sign_key` structure that will then be passed to `matrixSshNewSession` for each new SSH session. The SSH server must always nominate a private key that enables connecting clients to authenticate against.

This function takes a single private key in ASN.1 DER-encoded format and converts it to a `sign_key` type. The `keyStruct` is freed using the `matrixSshFreeKeys` API. However, the key must not be freed until the session is closed as `matrixSshNewSession` references the memory directly rather than creating a copy.

MatrixSSH supports industry-standard RSA keys.

matrixSshFreeKeys()

Prototype

```
void matrixSshFreeKeys(sshRsaKey_t *keyStruct);
```

Parameters

keyStruct	Input	Private key returned from a previous successful call to <code>matrixSshReadKeysMem</code>
-----------	-------	---

Free the `keyStruct` returned from a previous call to `matrixSshReadKeysMem`. However, the key must not be freed until the session is closed as `matrixSshNewSession` references the memory directly rather than creating a copy.

matrixSshNewSession()

Prototype

```
int matrixSshNewSession(sshSession_t **ses, sshRsaKey_t *keyStruct, matrixLogCb_t *logCback, matrixPassCb_t *passwordCback, sshAuthKey_t *authKeys, char *banner, char *motd);
```

Parameters

ses	Output	Handle to a new SSH session
-----	--------	-----------------------------



keyStruct	Input	Private key created from a previous call to <code>matrixSshReadKeysMem</code>
logCbck	Input	User callback routine to receive SSH log messages from the library. See <i>prototype below</i> .
passwordCbck	Input	User callback to receive and process username/password combinations from client login attempts. Return 0 on successful authentication, < 0 for invalid user or password. See <i>prototype below</i> .
authKeys	Input	List of usernames and public keys to enable Public Key authentication of connecting clients (if applicable). NULL otherwise
banner	Input	Optional banner string to be displayed to client prior to login. NULL otherwise
motd	Input	Optional message of the day string to be displayed to client after a successful login. NULL otherwise
Return Value	Output	0 on success, < 0 on failure

```
typedef void matrixLogCb_t(sshSession_t *ses, int priority, const char *format,
    va_list param);
```

ses	Input	Handle to this SSH session
priority	Input	Severity of the message from 0 to 7 inclusive, with 0 being the highest priority (emergency) and 7 the lowest (debug).
format	Input	sprintf style format string.
param	Input	Variable length arguments matching format string



```
typedef int matrixPassCb_t(unsigned char *username, unsigned char *password);
```

username	Input	Client's given username
password	Input	Client's password to be validated
Return Value	Output	0 if username and password validation successful, < 0 otherwise.

Start a new SSH server session. This function should be called for each new incoming SSH client connection to assign the session parameters and return the session handle that will be used as the input parameter to all other MatrixSSH APIs.

If the server wishes to perform Public Key Authentication for authentication connecting clients (as opposed to Password Authentication) the authKeys parameter should be populated with username/key combinations in an sshAuthKey_t array. For a working example, see the ./matrixssh/examples/matrixssh.c server and the accompanying authorizedkeys.h file for expected formatting.

If password authentication is used, the user 'passwordCback' callback will be called to validate the username and password typed at the client login prompt. The number of password attempts is defined in options.h as MAX_AUTH_TRIES, with a default of 3 attempts until the session will be closed. There is also a time delay period after an unsuccessful attempt that is governed by SSH_AUTH_DELAY in options.h and its default is 2 seconds.

matrixSshDeleteSession()

Prototype

```
void matrixSshDeleteSession(sshSession_t *ses);
```

Parameters

ses	Input	Session handle from a previous successful call to matrixSshNewSession
-----	-------	---

This is a session closure function that must be called on both successful session termination and on unrecoverable failures (provided matrixSshNewSession has been called previously).

The ses parameter will no longer be valid after this function is called.

matrixSshRegisterDataCallback()

**Prototype**

```
int matrixSshRegisterDataCallback(sshSession_t *ses, matrixDataCb_t *dataCb,
    matrixFreeCb_t *freeCb, void *data, char *prompt, int flags);
```

Parameters

ses	Input	Handle to this SSH session generated from a previous call to matrixSshNewSession
dataCb	Input	A callback function to handle incoming client data, commands, etc.
freeCb	Input	An optional callback function that will be invoked at session closure to free any resources associated with the 'data' user context parameter
data	Input	Optional user context that will be passed into the dataCb function for association with the session.
prompt	Input	Null terminated string specifying the command line prompt the client will be presented with.
flags	Input	If SSH_RAW_DATA is passed as a flag, the dataCb will be called with the raw bytes decoded from the SSH stream, including the terminal control characters. Otherwise, the callback will be called only when an entire line is parsed.
Return Value	Output	0 on success, < 0 on failure

```
typedef int matrixDataCb_t(sshSession_t *ses, void *data, char *buf, int len);
```

ses	Input	Handle to this SSH session generated from a previous call to matrixSshNewSession
-----	-------	--



data	Input	The user context value that was passed in as 'data' to matrixSshRegisterDataCallback
buf	Input	Pointer to data sent from an SSH client available to process.
len	Input	Length of 'buf' in bytes
Return Value	Output	The user callback can return 0 on success, or < 0 for errors that should cause the connection to close.

```
typedef void matrixFreeCb_t(void *data);
```

data	Input	Provides an opportunity for the user to free the user context value that was passed in as 'data' to matrixSshRegisterDataCallback
-------------	-------	---

This function is used to register a data handler for the session that will process application-specific requests received from the client (what the user is typing at the prompt). This dataCback function will be invoked after a successful SSH connection and the client begins sending command requests. The data will have been unencrypted at this point and ready for parsing.

The dataCback function should return <0 if there is an unrecoverable error in the parsing of the data or if the user wishes to exit based on received data.

If the dataCback handler has reply data to be sent to the client the matrixSshQueueOutgoingData function should be used to encrypt and queue the outgoing message.

The user may also nominate an associated data structure using the void *data parameter. If not NULL, the dataCback handler will be invoked with the data pointer for any custom storage and accounting the user wants to maintain for the session.

If the data parameter is used, a freeCback handler may also be nominated to perform any necessary cleanup at session closure. The freeCback handler will be invoked when matrixSslDeleteSession is called.

matrixSshQueueServerIdent()

Prototype

```
int matrixSshQueueServerIdent(sshSession_t *ses, char *remotehost);
```



Parameters

ses	Input	Session handle from a previous successful call to <code>matrixSshNewSession</code>
remotehost	Input	IP or hostname string identifying the connecting client. This value is used in log messages only.
Return Value	Output	0 on success, < 0 on failure

This function is used to construct the initial identification protocol message that the server will send to the client. This function should be called one time for each session after `matrixSshNewSession` has been called.

The identification message will eventually be sent as part of the main application loop in which calls to `matrixSshGetWriteBuffer` are used to retrieve data that should be sent over the wire to the client.

`matrixSshQueueOutgoingData()`

Prototype

```
int matrixSshQueueOutgoingData(sshSession_t *ses, unsigned char *buf, int len);
```

Parameters

ses	Input	Session handle from a previous successful call to <code>matrixSshNewSession</code>
buf	Input	The user data to be encrypted and queued to be sent to the client.
len	Input	Input. Length in bytes of the <code>buf</code> parameter.
Return Value	Output	0 on success, < 0 on failure

This function is used to encrypt and queue application data that is being sent over the wire to the client. This function must only be called from within the user data callback routine that is nominated with the `matrixSshRegisterDataCallback` function.

As with all data that will eventually be sent over the wire, this queued data will be retrieved by the main application loop via `matrixSshGetWriteBuffer`.



matrixSshQueueCloseSession()

Prototype

```
int matrixSshQueueCloseSession(sshSession_t *ses);
```

Parameters

ses	Input	Session handle from a previous successful call to matrixSshNewSession
Return Value	Output	0 on success, < 0 on failure

This function encrypts and queues a session closure message that will be sent to the client for clean session termination.

As with all data that will eventually be sent over the wire, this queued data will be retrieved by the main application loop via matrixSshGetWriteBuffer.

matrixSshDecode()

Prototype

```
int matrixSshDecode(sshSession_t *ses, unsigned char *readBuf, int bytesRead);
```

Parameters

ses	Input	Session handle from a previous successful call to matrixSshNewSession
readBuf	Input	The encrypted data received from the SSH client (typically via sockets)
bytesRead	Input	Input. The length in bytes of 'readBuf'



Return Value	Output	SSH_SUCCESS on success SSH_FATAL on failure SSH_PARTIAL - An incomplete data record was passed in readBuf. The user must obtain a new buffer with matrixSshGetReadBuffer, read more client data from the socket, and call matrixSshDecode again.
---------------------	--------	--

This function is called to process all data received from the client side during the SSH session. This decode function is used in conjunction with matrixSshGetReadBuffer to retrieve an available buffer location for incoming data and to process the received readBuf data.

If the incoming readBuf is part of the internal SSH protocol the library will parse and queue any necessary replies for later retrieval by the matrixSshGetWriteBuffer function.

If the incoming readBuf is user input from the client after a successful connection the library will decrypt the data and invoke the callback data handler that was registered with matrixSshRegisterDataCallback.

matrixSshGetReadBuffer()

Prototype

```
int matrixSshGetReadBuffer(sshSession_t *ses, unsigned char **buf, int *bufLen);
```

Parameters

ses	Input	Session handle from a previous successful call to matrixSshNewSession
buf	Output	Returns a pointer to allocated memory which encrypted client data off the wire should be stored
bufLen	Output	The number of available bytes that may safely be written to 'buf'
Return Value	Output	0 on success, < 0 on failure

If the server is expecting incoming client data (or has woken up on a read event) this function must be used to retrieve buffer storage for that data read.



This function is part of the main application read loop that obtains incoming buffer storage (this function), reads data from the wire (platform specific), and processes the populated buffer with `matrixSshDecode`. If `matrixSshDecode` returns `SSH_PARTIAL` this function should be called again and more data read from the wire.

`matrixSshUpdateWriteBuffer()`

Prototype

```
int matrixSshUpdateWriteBuffer(sshSession_t *ses, int bytesWritten);
```

Parameters

<code>ses</code>	Input	Session handle from a previous successful call to <code>matrixSshNewSession</code>
<code>bytesWritten</code>	Input	Number of bytes successfully sent out the wire from the buffer returned by <code>matrixSshGetWriteBuffer</code>
Return Value	Output	0 on success, < 0 on failure

This function must be called after each wire send to update the internal write buffer.

The main application write loop will look to see if there is data to send with a call to `matrixSshGetWriteBuffer`, send as much of that data as possible out the wire, and then inform the library how many bytes were written with a call to this function. This write loop will continue until `matrixSshGetWriteBuffer` returns 0.

`matrixSshGetWriteBuffer()`

Prototype

```
int matrixSshGetWriteBuffer(sshSession_t *ses, unsigned char **buf, int *bufLen);
```

Parameters

<code>ses</code>	Input	Session handle from a previous successful call to <code>matrixSshNewSession</code>
<code>buf</code>	Output	Pointer to the buffer containing encrypted data that is to be sent out the wire
<code>bufLen</code>	Output	Length in bytes of 'buf'



Return Value	Output	0 on success, < 0 on failure
---------------------	--------	------------------------------

This function is used to retrieve data that is to be written out to the wire. After the platform specific write the user must call `matrixSshUpdateWriteBuffer` with the number of bytes successfully written.

This function will return that there are no bytes to be written if called within `SSH_AUTH_DELAY` seconds of a failed login attempt. This feature is useful in combating automated clients attempting to guess username/password combinations. The `SSH_AUTH_DELAY` define can be found in the `options.h` file and has a default value of 2 seconds. This feature can be disabled by setting the define value to 0.

matrixSshHandleTimeouts()

Prototype

```
int matrixSshHandleTimeouts(sshSession_t *ses);
```

Parameters

ses	Input	Session handle from a previous successful call to <code>matrixSshNewSession</code>
Return Value	Output	0 on success, < 0 on failure

This function verifies that the session has not timed out. If it has, a negative return code will be returned, and the session will be closed.

This function also will initiate a new key exchange with the client if a certain amount of time has passed on the connection, or a certain amount of data has been sent. These values are adjustable in `options.h` as parameters `KEX_REKEY_TIMEOUT` and `KEX_REKEY_DATA` respectively. By default these values are 1 hour and 1 GB.

This API can also queue keep-alive messages that will be sent according to `DEFAULT_KEEPALIVE` in `options.h`. By default this setting is 0, and keep-alive messages are not sent.

Because this API handles various timely situations, **it should be called periodically** (approximately once per second) by the server application. Typically this can be done in the main event loop if it is known that an event will cause the loop to execute periodically.