

Python Secure Shell

# Introduction

The Python Secure Shell (PSS) will allow a user to open a terminal on a device over the internet using our service. A PSS will be bound to one session id and multiple such PSS instances can be active at a time, even on the same network.

# Shell Session

Each shell session will be a one to one connection. Even though they are all on the same network, commands will be sent from a specific session ID to another session ID. Let’s say we have a Raspberry Pi connected running Python Secure Shell and it was assigned a session ID: 1. From a PC another session has a Session ID: 6. When the RPi received a message from 6 then it will look in its array of active sessions. If 6 is not there then the user needs to start a new session. Otherwise it can continue with this session

The callback called when a new message is received from the CYW module looks as follows:

def data\_received\_callback(self, data, data\_type, from\_who):

The from\_who field contains the session ID of the device that sent the data.

Each shell session must be managed independently. If no data has been exchanged between PSS and a client for a certain amount of time then the session must be terminated. As part of the discovery response a unique key is sent to the client. This key is to make sure data always comes from the same client (similar to how a browser must send the content of a cookie to the server each time so the server can make sure it is getting the data from the correct device).

# Discovery

The client side needs a way to discover all the Python Secure Shell devices on the network. Part of the discovery message will be the discover password. The PSS will only respond if the discover password matches its own.

# Protocol Specification

## Discovery

After a user has connected to a network he needs to discover any PSS devices that are connected to his networks. The session IDs are assigned by the server, it is impossible for any clients to know what the session ID of a PSS device will be.

**Request:**

Data Type: DISCOVER-SSH  
Data: Discovery password

**Response:**

Data Type: DISCOVER-RESPONSE-SSH  
Data: 20 Byte unique key that must be sent with all data to PSS followed by a comma and then the device name. The device name is a user friendly name that can be shown to the user to identify this device.  
From Who: 16 (Session id of the device that was discovered, in this document I will use 16 as the Python Encrypted Shell and 4 for the client)

## Connect

This message must be sent to the device to start a session. The data field must be the unique key received by the discovery response.

**Request:**

Data Type: CONNECT-SSH  
Data: xxxxxxxxxxxxxxxxxxxx  
To Session ID: 16

## Sending Data

Data can be sent at any time and does not need a response. Data can also be received at any time. There needs to be a timeout, if no data was received from the client in a certain amount of time then the connection needs to be closed.

**Request:**

Data Type: REQ-SSH:xxxxxxxxxxxxxxxxxxxx (the unique key needs to be part of each request)  
Data: ls -l  
To Session ID: 16

**Response:**

Data Type: RESP-SSH  
From who: 16  
Data:   
total 9191503  
drwxr-xr-x 1 Hubert 197610 0 Feb 4 2014 ARM/  
-rw-r--r-- 1 Hubert 197610 30 Jun 13 2014 AVScanner.ini  
To Session ID: 4

## Terminate connection

Either side can terminate the connection. The user side can terminate the connection when the user closes the application. The device side can close the connection when the user types exit or if the connection times out.

**Request:**

Data Type: CLOSE-SSH  
Data: xxxxxxxxxxxxxxxxxxxx