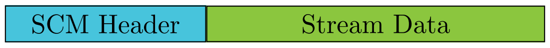
1 Socket Control Model (SCM)

The payload of the USB packet contains any combination of a single SCM packet, two or more SCM packets or a split SCM packet. SCM transfers can be split across USB packets but shall not be split across USB transfers.

1.1 SCM Transfer Format

The packet format defines an SCM packet. For information regarding USB packets refer to [USB2.0] specification. Details packets formats can be found in section 8.4 of the [USB2.0] specification.

An SCM packet consists of a 64-bit header and, depending on the command type, a varible length payload of arbitrary data. The length of this payload is indicated in the header. The payload will always immediately proceed the header.



1.2 SCM Transfer Types

The SCM packet header contains an opcode field (see section 1.3) to denote whether the transfer is an Immediate or Data type. Immediate types only contain the header while Data types contain a buffer of data immediately proceeding the packet of a length denotated in the header.

Table 1: Your first table.

Opcode	Name	Type	Purpose
0x00	OPEN	Cmd	Open a socket on the host
0x01	CONNECT	Data	Connect an open socket to a given address
0x02	CLOSE	Cmd	Disconnect and close a socket
0x03	WRITE	Data	Write data to a connected socket
0x04	ACK	Cmd	Acknowledge a command and indicate success
0x05	REPLY	Data	Similar to ACK but contains data.
0x06	IOCTL	Data	Tells the host to run ioctl on the sock.
0x07	SETOPT	Data	Tells the host to run setsockopt() on a sock.
0x08	GETOPT	Data	Tells the host to run getsockopt() on a sock.

1.3 SCM Packets

All values in SCM packets are little endian. All IDs and integer values are unsigned unless otherwise denoted.

1.3.1 Common Fields

1. **Opcode**: 8-bit unsigned integer identifying what type of packet follows.
2. **Message ID**: 8-bit unsigned integer provided to allow the reciever to identify the recieved message when replying. This is always genrated by the sender, and the reciever may only reply using the given ID once. Sending an ACK or REPLY to an unknown Message ID causes undefiend behavior.
3. **Sock ID**: 8-bit unsigned integer identifying which sock is being sent to. Sock IDs are always created by the device during an OPEN command.

1.3.2 SCM Command Packet

1.3.2.1 OPEN

The OPEN command is awlays initiated by the device to the host. The device will create a new message ID and new sock ID so the device can identify future operations on these objects.

0	8	16	24 31
0x00	Message ID	Sock ID	Reserved
Addr Family		Protocol	

Address Family is the ID used by the Linux kernel in `linux/socket.h`

Protocol is the ID used by the Linux kernel in `uapi/linux/in.h`

ACK will return as a 32-byte signed integer. On success ACK immediate will be 0, on failure the error code returned from the call.

1.3.2.2 CLOSE

When sent from device to host, disconnects (if connected) and closes a socket using the ID given during creation. If sent from host to device this will serve as a notification that the socket has been closed by the remote peer.

0	8	16	24 31
0x02	Message ID	Sock ID	Reserved
Exit Code (device to host only)			

On success ACK immediate will be 0, on failure the error code returned from the call (host to device only).

1.3.2.3 ACK

Upon completion of a message the reciever will send this back to acknowledge reciept and indicate whether the operation was a succes or a failure. Once USB has acknowledged reciept, the sender of an ACK will not wait for further confirmation that the recipient has recieved the message.

0	8	16	24 31
0x04	Message ID	Sock ID	Reserved
See ACK section on commands			

1.3.3 SCM Data Packet

Data packets contain arbitrary data immediately after the header of whatever length is contained in the headers length field. The data segment of specific commands may represent structures with endianness, all fields are individually converted to little endian before sending and converted back when recieved.

Data Length will always be an unsigned 32-bit integer.

1.3.3.1 CONNECT

Connect tells the host to connect a created socket to a given address. The address information passed will vary by protocol, the host and device should know which structs the other side will send and process those (these are typically the same on both ends).

Table 2: Connection payload struct defs.

Family	Protocol	Header	Struct name
IPv4	TCP	sockaddr_in	linux/socket.h

0	8	16	24 30 31
0x01	Message ID	Sock ID	Reserved
Length			
Address			
Address (cont...)			

ACK will return as a 32-byte signed integer. On success ACK immediate will be 0, on failure the error code returned from the call.

1.3.3.2 WRITE

Sends stream data over a connected socket. This command can be sent by either side.

0	8	16	24 31
0x03	Message ID	Sock ID	Reserved
Payload Length in bytes			
Payload data...			

ACK will return as a 32-byte signed integer. On error the return code (¡0) will be returned. Zero will be returned on success and positive codes will be returned when the transfer was a success but the reciever needs to tell the sender to change sending behavior (TODO: IMPLEMENT POSITIVE CODE BEHAVIOR)

1.3.3.3 REPLY

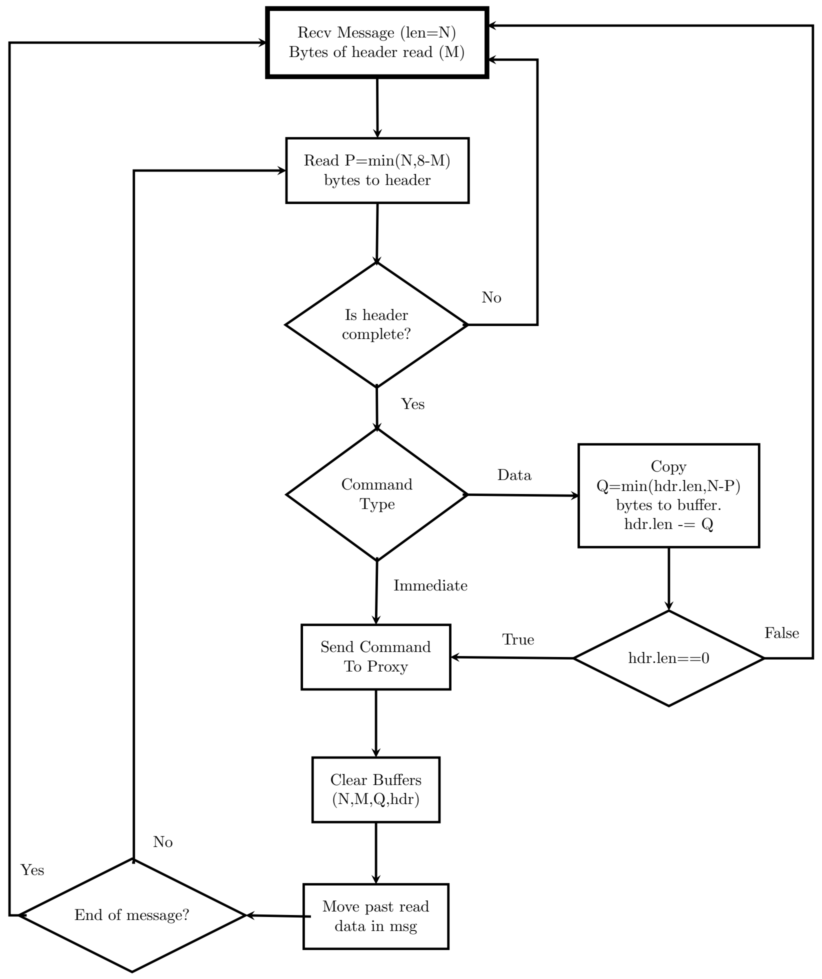
Similar to ACK but contains a data field for internal processing.

0	8	16	24 31
0x05	Message ID	Sock ID	Reserved
Payload length			
Payload data...			

2 Packet Processing

Fig A describes the process for the reciever on both ends to assemble and transmit packets from USB to their respective proxies. The Send Command To Proxy procedure can be assumed to be nonblocking, after which the data passed in can be safely freed.

2.1 Packet Processing Flow (Fig. A)


Recv Message (len=N)
Bytes of header read (M)
Read P=min(N,8-M)
bytes to header
Is header
complete?
No
Yes
Command
Type
Data
Copy
Q=min(hdr.len,N-P)
bytes to buffer.
hdr.len -= Q
Immediate
Send Command
To Proxy
True
hdr.len==0
False
Clear Buffers
(N,M,Q,hdr)
No
Yes
End of message?
Move past read
data in msg