ElBnetmux

TCP socket protocol description

31 January 2012

Introduction

Overview

The protocol employed by eibnetmux for TCP socket clients is very, very simple. It follows a simple request / response pattern. There are no sessions and there is no connection state kept between two consecutive requests. In fact, a client could theoretically close the TCP connection after receiving a response and reinitialise it for the next request. Obviously, this is not efficient and abuses client and server resources.

For Linux-based systems, there is an even simpler option: use the C client library which provides a few C calls to implement the protocol. Please refer to the man pages contained in the source for documentation.

This document describes the formats of the request and response packets.

Request flow

When talking to eibnetmux over the TCP socket interface, a client should implement the following request pattern:

Category	**Commands**
Connection setup	key exchange diffie-hellman-merkle authenticate name
Request	any request described in this document
Response	acknowledgement response error
Disconnect	exit close TCP connection

Endianness / byte ordering

Care should be taken when passing parameters to eibnetmux. Generally, they are in network byte order (big endian). The majority of PCs out there, x86-based systems, are small endian and will need to convert parameters. Please refer to stabdard C library (clib) functions htons() and htonl(), respectively.

Generic frame format

All request and response packets share the same structure:

Header		Parameters
Command	Param 1	Additional parameters

Field	Description
header	Must always be transmitted in full, even if the command does not require any parameters.
command	This is a single letter whose case is important (commands 'A' and 'a' are very different, indeed, for example). It is 8 bit in size.
param 1	This is a 16 bit numeric parameter. It is in network byte order and mostly used to specify KNX device addresses or data lengths.
additional parameters	Some frames, especially write requests and read responses, need to transmit additional information. This is the corresponding byte stream. Its internal format is command-specific.

KNX addresses

Many commands require the specification of a KNX logical group. These addresses are, commonly, in the form of m/s/g where m=maingroup, s=subgroup, g=group.

When communicating with the eibnetmux TCP socket server, such addresses must be packed in 16 bits. The format is 0mmm msss gggg gggg.

Decoding KNX data

Eibnetmux receives KNX data from the bus in a so-called cemi frame. Its format is defined as follows:

Field	Bits	Description
code	8	Transaction code
zero	8	This field is always 0.
ctrl	8	Priority and other flags.
ntwrk	8	Routing information.

saddr	16	Source address, physical KNX device address.
daddr	16	Destination address, usually logical KNX group address.
length	8	Number of bytes of data contained in frame.
tpci	8	Transport layer data.
apci	8	Application layer data.
data	0 — 112	KNX data formatted according to KNX logical group EIS code.

Some of the commands return the full cemi frame, others only the data part. It is important to note that the data part starts with the apci field.

A KNX logical group's data type, the EIS code, must be known to be able to fully decode the data. EIS stands for EIB Interworking Standards. It defines the following data types:

EIS	Data bytes	Function
1	0	Switching (on/off)
2	0	Dimming (up/down)
3	3	Time
4	3	Date
5	2	Value
6	1	Scaling
7	0	Drive control
8	0	Priority
9	4	Float value
10	2	16-bit counter
11	4	32-bit counter
12	4	Access
13	1	EIB-ASCII-Char

14	1	8-bit counter
15	1 — 14	String

Decoding algorithms are beyond the scope of this document. Please refer to the source code of the PHP client library which contains encoding & decoding algorithms.

Request / response descriptions

Connection setup

Key exchange	request, optional

If user authentication is required, passwords must not be transmitted in clear-text. With this function, a client initialises creation and secure exchange of an appropriate encryption key. It will result in a Diffie-Hellman-Merkle key exchange where the eibnetmux server provides its DHM parameters to the client with the response to this command. Later on, the client will need to send its own parameters using the diffie-hellman-merkle command (see below).

command	K
param 1	
additional parameters	

Server DHM parameters	response

This is the server's response to a key exchange command. It contains the Diffie-Hellman-Merkle parameters (P, G, Ys) needed by the client to calculate the shared enctyption key.

command	K
param 1	length
additional parameters	block of size 'length' containing the server key exchange parameters. It's internal format is defined by PolarSSL's dhm_read_public() function.

Diffie-Hellman-Merkle	request, optional

To conclude the DHM key exchange, the client sends its own parameters to the server. This command must only be sent after receiving a Server DHM parameters response.

command	D
param 1	length

additional parameters	block of size 'length' containing the client key exchange parameters. It's internal format is defined by PolarSSL's dhm_read_public() function.

DHM ok — response

Finally, the server responds with a positive status code.

command	D
param 1	0
additional parameters	

Authenticate — request, optional

If eibnetmux has been setup with active security checks, the user must authenticate to be allowed to perform KNX bus functions.

command	A
param 1	length
additional parameters	Streamof length bytes consisting of username and encrypted password, both terminated by a NUL byte. The length includes the terminator characters.

Authentication ok — response

If username and password match, eibnetmux returns this status.

command	A
param 1	0
additional parameters	

Name — request, mandatory

The client must set its identifier. It is used in status reports.

command	a

param 1	length
additional parameters	client identifier, usually this is the client type, e.g. 'webmon'. The identifier can be of any length but only the first 64 characters are used.

Name ok	response

Confirm setting of the client identifier.

command	a
param 1	0
additional parameters	

Get protocol version

Version	request

Get version of eibnetmux TCP socket protocol.

command	V
param 1	
additional parameters	

Version data	response

The TCP socket API protocol version number.

command	V
param 1	Version number. Currently, this value is 3.
additional parameters	

Reading data from KNX

Read	request

Read the current value of a KNX logical group and return it.

command	R
param 1	KNX logical group address
additional parameters	

Read once — request

Read the current value of a KNX logical group and return it. Immediately afterwards, the server closes the TCP connection.

command	r
param 1	KNX logical group address
additional parameters	

Read data — response

The eibnetmux server returns data retrieved from the KNX bus to the client.

command	R
param 1	length
additional parameters	KNX data starting with the apci field of the cemi frame

Monitor — request

Instruct eibnetmux to monitor the KNX bus for data packets sent to one or more KNX logical groups and forward the packets to the client.
After specifying this command, the client must not send another command (except Exit) but keep the TCP connection open and wait for eibnetmux responses.

command	M
param 1	KNX logical group address mask This is a bitmask of KNX logical group addresses which are monitored. If you want to monitor everything, specify 0xffff.
additional parameters	

Monitor data	response

Each KNX request seen on the bus is forwarded to the eibnetmux client (as long as the address matches the mask).

command	M
param 1	length
additional parameters	full cemi frame

Writing data to KNX

Write	request

Write new data to a KNX logical group.

command	W
param 1	KNX logical group address
additional parameters	length, value Length is a 16-bit value specifying the number of data bytes to follow for the value. The value itself must be correctly formatted to contain the apci and data fields of a cemi frame.

Write once	request

Same command as Write except that eibnetmux closes the TCP connection after sending the value to the KNX bus.

command	w
param 1	KNX logical group address
additional parameters	length, value Length is a 16-bit value specifying the number of data bytes to follow for the value. The value itself must be correctly formatted to contain the apci and data fields of a cemi frame.

Write confirmation	response

If the write command was executed successfully, this confirmation is returned to the client.

command	W
param 1	0
additional parameters	

Terminate connection

Exit	request

Instruct eibnetmux to close the TCP socket connection.

command	X
param 1	
additional parameters	

Exit confirmation	response

After returning this confirmation, the TCP socket is closed immediately.

command	X
param 1	0
additional parameters	

Error response

Error	response

If an error occurs, eibnetmux will return an error code.

command	E
param 1	error code 0x00 - no error 0x01 - socket closed 0x02 - connection table full 0x03 - bad request 0x04 - unknown command 0x05 - timeout 0x06 - unauthorised 0x07 - user name or password failure

0x08 - dhm operation failed
0x09 - wrong or missing parameter
0x0a - general communication failure

additional parameters

Additional commands

There is a list of additional commands used for management purposes.

Command	Description
connect	Connect eibnetmux' eibnet/ip client to remote server, e.g. an N148/21.
disconnect	Disconnect eibnetmux' eibnet/ip client from remote server.
get log level	Return current log level.
set log level	Set new log level.
status	Return eibnetmux' status.
get access block	Return current access block level.
block accesses	Block access above this level.
close connection	Forcibly terminate a client connection.

The documentation for these functions has not yet been completed. Please refer to the source code of eibnetmux for detailed information.