```
*************************************************************************
*
*      EG-PMS-LAN data exchange protocol with Power Manager client
*
************************* Version 2.1 ***********************************
```

Handshaking and client authorization

Client sends Start condition – 1-byte packet (0x11). EG replies with randomly generated Task (task[4]). Client calculates Solution (res[4]) and sends to EG. EG checks, if Solution is correct or not. If correct EG sends Status, otherwise sends nothing and resets to the initial state. If EG did not receive Solution within 4 seconds after sending Task – it goes to initial state too.

Getting socket states and controlling the sockets

EG sends Status, it is encrypted (statcryp[4]). Client decrypts it and use (stat[4]) for own purpose. After that, and on the every next turn, client might reply nothing, so EG will go back to the initial state in 4 seconds. This may be used for monitoring EG state without controlling it. Alternatively, client sends encrypted Controls (ctrlcryp[4]). EG decrypts it, checks the result (ctrl[4]) for validity (every ctrl shall have valid value: 0x01, 0x02 or 0x04), and, if valid, does socket control. After that, EG sends new encrypted Status (statcryp[4]), which contains updated socket states after implementing controls. Then client might send encrypted schedule (schcryp[]) for one of the sockets. If purpose is to get schedule from EG without setting it, it needs to send “dummy schedule”. As result, the old schedule will not be changed. EG receives it, decrypts and checks the checksum (checksum). If checksum is ok, EG implements new schedule, sends it (schcryp[]) to the client, and goes to the initial state. If checksum is not ok – EG does not reply and goes to the initial state.

Encryption and decryption formulas:

```
res[4] - Solution
task[4] - Task
key[8] - EncryptionKey
stat[4]    - Decrypted status
statcryp[4] – Encrypted status
ctrl[4] - Encrypted control
ctrlcryp[4] - Decrypted control

Initial encryption key after IP configuration reset is
0x20,0x20,0x20,0x20,0x20,0x20,0x20,0x31 (i.e.       1)

res[1:0]=((task[0]^key[2])*key[0])^(key[6]|(key[4]<<8))^task[2] (unsigned
multiply);
res[3:2]=((task[1]^key[3])*key[1])^(key[7]|(key[5]<<8))^task[3] (unsigned
multiply);

statcryp[i]=((stat[3-i]^task[2])+task[3])^key[0]+key[1]

stat[3-i]=(((statcryp[i]-key[1])^key[0])-task[3])^task[2]

ctrlcryp[i]=(((ctrl[3-i]^task[2])+task[3])^key[0])+key[1]

ctrl[3-i]=(((ctrlcryp[i]-key[1])^key[0])-task[3])^task[2]
```

```
schcryp[length-i]=(((sch[i]^task[2])+task[3])^key[0])+key[1];

sch[length-i]=(((schcryp[i]-key[1])^key[0])-task[3])^task[2];

length =
(unsigned int)checksum = 0 - (unsigned int)(sch[0]+sch[1]+sch[2]+...sch[length-
2])

Control and state format:

stat[i]=0x41 – voltage is present on socket i+1
stat[i]=0x82 - voltage is absent on socket i+1

ctrl[i]=0x01 – switch socket i+1 on
ctrl[i]=0x02 - switch socket i+1 off
ctrl[i]=0x04 – no switching socket i+1

Schedule format (decrypted):

sch[0..3]=timestamp, 4 bytes. It is time on device, when the schedule was set up
sch[4..length-9]=up to 45 entries
sch[length-8]=0xE5 , marker of loop period
sch[length-7..length-4]=time of loop period in seconds
sch[length-3]=socket number (1..4) and “dummy schedule” bit (most significant).
If it is set, then the it it “dummy schedule”, which will be omited by device.
sch[length-2..length-1]=checksum, 2 bytes
length = number of bytes of schedule, including the checksum and timestamp. Can
be from 12 up to 238 bytes.

Schedule entry format:

1st byte – control and period attribute.
0x00=switch socket off, once
0x01=switch socket on, once
0x02=switch socket off, periodically
0x03=switch socket on, periodically
2..5 byte – time of entry execution, since 1 January 1970, GMT+0


//// End of document.///
```