Decoding the PowerLink31 Messages

V1.1 – 2024-07-10

CHANGES FROM PREVIOUS VERSION

- Amended naming of message types
- Added section for type 0 and 4 messages – needs completing.
- Added command 19 in B0 Commands

PREFACE

The following is my explanation of how the messages from a PowerLink31 device are constructed. It has been derived from trial and error in working through decoding these messages and may have errors or omissions. Any use of this information is subject to the usual, no warranties, no guarantees, no fault and is at your own risk.

The following parts, I think are pretty robust and seem to work for all messages I have seen so far. That is not to say there are messages I have not seen that fail this logic.

- Decoding the Powerlink31 Wrapper
- Decoding the B0 message

However, what each of the messages actually mean and the translation of the data maybe subject to change as progression is made.

- B0 Commands
- Command 35 Settings
- Command 42 Settings
- Data Indexes

I do have working code that can build and decode these messages based on the below information and also sit as a MITM between a panel and a PowerManage server to decode the back and forth between them.

BASIC MESSAGE STRUCTURE

The powerlink31 message is of the following format (in hex)

0a 37 36 45 32 30 30 32 32 22 56 49 53 2d 41 43 4b 22 30 31 34 32 4c 30 30 31 32 33 35 23 32 41 34 43 43 33 5b 0d 02 43 ba 0a 5d 0d

Which translates to

76E20022"VIS-ACK"0142L001235#AAAAAA[0d 02 43 ba 0a]\r

This consists of 2 parts. The Powerlink31 wrapper and the actual message (the bit wrapped in square brackets). The below will deal with each part

DECODING THE POWERLINK31 WRAPPER

NOTE: Some of the byte references may be slightly different depending on the length of some of the constituent parts of the message. These refer specifically to the example data above, but the description comments make note of those that can vary.

Bytes	Example Data	Descritption
0 -> 1	76 E2	CRC16ARC checksum of bytes from 4 to end of message
2 -> 3	00 22	Message length from byte 4 to end of message
4 -> 12	"VIS-ACK"	The type of message. This can vary in length and should be decoded by looking for data between 2 quotes ("). I have seen mainly "VIS-ACK" (ACK message), "VIS-BBA" (command and response message), "*ADM-CID" and "*ACK" – on rare occasions and seem to happen with an alarm tamper.
13 -> 16	0142	This is an incrementing message counter. Zero prefixed and always 4 digits
17	L	Account number start marker
18 -> 23	001235	Account number (as set in the Broadband communication settings of your alarm). Messages from a PowerManage server send this as a 0, but the alarm sends the full 6 digits. Therefore when decoding, this is the data between L and #
24	#	Alarm serial start marker
25	AAAAAA	Your alarm serial number
26	[	Start of wrapped message marker
27 -> 31	0d 02 43ba 0a	Message – this can vary in size a lot. This example is an ACK message. You should decode on the basis of the data between [and]
32	]	End of wrapped message marker
33	\r	End of PowerLink message

DECODING THE WRAPPED MESSAGE

The above PowerLink31 wrapper is relatively consistent and straight forward to decode. The wrapped message, however, breaks down into a number of parts that have a specific decoding requirement each.

We will use a different base example here, which is just the wrapped message inside the [] of the PowerLink31 message.

0d b0 03 51 06 ff 08 ff 01 18 ef 43 a0 0a

Bytes	Example Data	Description
0	0d	Message start indicator. Always 0d
1	b0	Message class indicator. This is a b0 message (see b0 messages below), but this can be other types too (see Standard messages below)
2 -> 11	03 51 06 ff 08 ff 01 18 ef 43 a0	Message data. Refer to above.
13	0a	Message end indicator. Always 0a

DECODING THE STANDARD MESSAGE

Work in progress…

DECODING THE B0 MESSAGE

From our example above

0d b0 03 51 06 ff 08 ff 01 18 ef 43 a0 0a

Bytes	Example Data	Description
0	0d	Start of message marker
1	b0	B0 message indicator
2	03	Message type: 0 – Add 1 - Request 2 - Paged Response 3 - Response 4 – Remove 5 - Unknown
3	51	Message command. If this is 35 or 42, see below in variations. Also see known B0 commands for what they each represent.
4	06	Length of remaining message from byte 4 to 2 before end.

Message Types 0 (Add) and 4 (Remove) – Sent to panel

To document properly!!

Think 0 is add and 4 is remove as if enrolling new device or setting bypass, sends a 0, removing these things sends a 4.

2 variations of message seen

0d b0 00 25 10 aa aa 01 ff 08 ff 09 31 34 30 35 30 31 39 07 00 43 03 0a -> Enrol device

Seems similar to a response except download code in 5 & 6, data length in 11 (09) Data is ascii encoded enrolment id from 12.

0d b0 04 25 09 aa aa 01 ff 08 03 02 08 00 43 6e 0a -> remove device

0d b0 00 19 0f aa aa 00 ff 01 03 08 08 00 00 00 00 00 00 00 43 7a 0a -> Set bypass

0d b0 04 19 0f aa aa 00 ff 01 03 08 08 00 00 00 00 00 00 00 43 76 0a -> remove bypass

(index based in a chunk (byte 9 is data type (01 - bits), byte 10 is index (03 - zones), byte (11 is chunk data length, rest up to 43 is data) as per ‘byte7 is not ff’ variation in responses, except download code is in 5 & 6

Probably more commands of this type – need investigating.

Message Type 1 (Request) – Sent to panel

There are 2 possible options here. Those that have parameters (command 35 and 42 – maybe more) and those that don't.

No Parameters (byte 4 is 01)

0d b0 01 1d 01 05 43 e7 0a

Bytes	Example Data	Description
5	05	Data – seems always 05
6	43	End of data marker. Always 43.
7	A0	Checksum of b0 to byte 10 (or last byte not including checksum)
8	0a	End of message marker

Has Parameters (byte 4 is not 01)

A request can have 1 or more settings in the message. For each setting requested an individual response is provided.

1 Parameter

0d b0 01 35 07 02 ff 08 ff 02 00 00 43 c2 0a

2 Parameters

0d b0 01 35 09 02 ff 08 ff 04 00 00 01 00 43 bd 0a

Bytes	Example Data	Description
5	02	Parameter size. Ie in this case parameters are 2 bytes each
6	ff	Always ff
7	08	Data type – 08 is bytes
8	ff	Start of data marker
9	04	Length of data. Divide this by parameter size to get number of parameters
10 -> length of data	00 00 or 00 00 01 00	Parameters
11	43	End of data marker. Always 43.
12	a0	Checksum of b0 to byte 10 (or last byte not including checksum)
13	0a	End of message marker

Message Types 2 (Paged Response) and 3 (Response) – Sent from panel

0d b0 03 51 06 ff 08 ff 01 18 da 43 b5 0a

We will cover paged messages specifically below, but it is worth noting here that if any response returns a paged response, the first response will be type 2, followed by more type 2s or a type 3.

Paged (type 2) messages are always finished with a type 3 message with the last part of the data. This paged approach is designed to overcome some limitation of how long each message can be and therefore it is broken down into paged responses.

Bytes	Example Data	Description
5	ff	Page number. For a message type 3, this is always ff, for a message type 2, this is the page number.
6	08	This is the type of data and represents the size in bits of each data element in the following data. If this is 0, see below in variations.
7	ff	Start of data marker. If this is not ff, see Byte 7 is not FF in the variations below.
8	01	Length of data
9	18	Data. This can vary based on the value in byte 7.
10	ef	Incremental message counter
11	43	End of data marker. Always 43.
12	a0	Checksum of b0 to byte before checksum. In this example byte 11
13	0a	End of message marker

VARAIATIONS

There are some variations to this basic structure, which are shown below.

Message Request (where byte 1 is 1)

Byte 6 is 0 (Seen twice – command 64 & 69)

This seems that the 0 data type indicates the data type is in 2 bytes in 8 and 9.

0d b0 03 64 18 ff 00 ff 13 00 ff 10 4a 53 37 30 33 36 34 36 20 4b 32 30 2e 32 31 34 8a 43 72 0a

Byte	Example	Description
0 -> 5		Same as main description
6	00	Think says data type in bytes 8 & 9
7	ff	
8 -> 9	13 00	Think data type
10	ff	
11	10	Data length
12 -> data length	4a 53 37 30 33 36 34 36 20 4b 32 30 2e 32 31 34	Data
End of data	8a 43 72 0a	Same as main description from byte 10

Command is 0f (Only seems to be this command)

This seems to be a unique format that does not follow anything else. I have not found another message that does this.

0d b0 03 0f 0f 19 08 0f 00 00 00 02 03 00 83 00 03 00 03 e6 43 45 0a

Byte	Example	Description
0 -> 3		Same as main description
4	0f	Data length
5	19	Don't know
6	08	Data type
7 -> data length – 4	00 00 00 02 03 00 83 00 03 00 03	Data
End of data	E6 43 45 0a	Same as main description from byte 10

Command is 35 or 42

These are special settings responses and contain the setting identifier in the response.

0d b0 03 35 18 ff 08 ff 13 2d 00 06 4a 53 37 30 33 36 34 36 20 4b 32 30 2e 32 31 34 1a 43 e6 0a

Byte	Example	Description
0 -> 8		Same as main description
9 -> 10	2d 00	Setting. See Command 35 Settings and Command 42 Settings
11	06	Data type – See Command 35 Data Types and Command 42 Data Types
12 -> data length -3	4a 53 37 30 33 36 34 36 20 4b 32 30 2e 32 31 34	Data
End of data	1a 43 e6 0a	Same as main description from byte 10

Byte 7 is not ff (Many commands)

This is a data format that contains multiple data elements in chunks. In the raw data, you can see each chunk separated by ff as the starting marker (except the first chunk in a paged response as this is the page number)

0d b0 03 02 2d ff 01 00 01 00 ff 01 01 02 00 00 ff 01 02 01 00 ff 01 03 08 00 00 00 00 00 00 00 00 ff 01 04 04 00 00 00 00 ff 01 05 04 00 00 00 00 43 43 6d 0a

First message has to be split into these chunks on the following basis. Note, it is dangerous to use the ff marker as some data can be ff too.

1. Iterate from byte 5 to the message length (from byte 4)

2. Byte 8 for first chunk data length – this is only the data size but we chunk each section before decoding chunk so we add 3 to our chunk length and then:
3. Chunk is byte 6 to chunk length + (6 + 3)
4. Keep going adding chunk data length + 4 for the start of the next chunk

This gives us a list of chunks.

Each chunk is then decoded as follows:

Byte	Example	Description
0	08	Data type – see data types
1	00	Data index – These relate to a data category. Ie 3 is zones. See Data Indexes
3	04	Data size
4 -> data size	07 07 07 07	Data

Example

0d b0 02 01 94 ff 08 00 04 07 07 07 07 ff 08 01 0f 06 06 06 06 06 06 06 06 06 06 06 06 06 06 06 ff 08 02 08 02 07 07 07 07 07 07 07 ff 08 03 40 02 02 02 02 02 02 02 00 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 ff 08 04 20 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 01 08 05 00 8f 43 07 0a

So,

- Message class is b0
- Message type is 02 (Paged response)
- Message command is 01
- Message length 148 (0x94)

Chunked data is:

[

“08 00 04 07 07 07 07”,

“08 01 0f 06 06 06 06 06 06 06 06 06 06 06 06 06 06 06”

“08 02 08 02 07 07 07 07 07 07 07”,

“08 03 40 02 02 02 02 02 02 02 00 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07”,

“08 04 20 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07”

]

DEALING WITH PAGED RESPONSES

So, one thing that seems to happen is that a paged response doesn't split a chunk. Ie, if the data is chunked into 10 bytes, each page will have a number of 10 byte chunks. No partial chunks exist at the end.

However, (especially with zone related data on a panel with 64 zones – index 3) an indexed chunk can be split and therefore some of index X is in one message and some in another.

Therefore, to rebuild a page, we can just keep adding the chunks in each message, noting that if it is an indexed chunk, we need to add that data (minus the first 3 bytes (data type, index and size)) to any existing chunk of that index.

Example – Not Split (Command 1 – Some device status/setting)

Message 1

```
b0 02 01 94 ff 08 00 04 07 07 07 07 ff 08 01 0f 06 06 06 06 06 06 06 06 06 06 06 06 06 06 06 ff 08 02
08 02 07 07 07 07 07 07 07 ff 08 03 40 02 02 02 02 02 02 02 00 07 07 07 07 07 07 07 07 07 07 07 07
07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07
07 07 07 07 07 07 07 07 07 07 07 ff 08 04 20 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 07
07 07 07 07 07 07 07 07 07 07 07 07 07 07 07 01 08 05 00 f2 43 a3 0a
```

You can see message type is 2 (Paged) and it has indexes 00, 01, 02, 03 and 04 (the number after each ff 08.

Message 2

```
b0 03 01 25 ff 08 05 20 06 06 06 06 06 06 06 06 06 06 06 06 06 06 06 06 06 06 06 06 06 06 06 06 06
06 06 06 06 06 06 06 f3 43 01 0a
```

You can see message type is 3 (Response) and it has index 05 only.

Example – Split (Command 4b – Zone last event)

Message 1

```
0d b0 02 4b b4 01 28 03 af 8c a1 8a 66 02 6f 45 8a 66 04 65 8f 8a 66 04 bc a0 8a 66 04 d6 a0 8a 66 04
11 85 89 66 04 87 8f 8a 66 04 41 ec 6d 38 00 41 ec 6d 38 00 41 ec 6d 38 00 41 ec 6d 38 00 41 ec 6d
38 00 41 ec 6d 38 00 41 ec 6d 38 00 41 ec 6d 38 00 41 ec 6d 38 00 41 ec 6d 38 00 41 ec 6d 38 00 41
ec 6d 38 00 41 ec 6d 38 00 41 ec 6d 38 00 41 ec 6d 38 00 41 ec 6d 38 00 41 ec 6d 38 00 41 ec 6d 38
00 41 ec 6d 38 00 41 ec 6d 38 00 41 ec 6d 38 00 41 ec 6d 38 00 41 ec 6d 38 00 41 ec 6d 38 00 41 ec
6d 38 00 41 ec 6d 38 00 41 ec 6d 38 00 41 ec 6d 38 00 85 43 73 0a
```

You can see message type is 2 (paged), page number is 1 (byte 5) and it only has 1 index entry of 3 (byte 7).

Message 2

```
0d b0 03 4b 96 ff 28 03 91 41 ec 6d 38 00 41 ec 6d 38 00 41 ec 6d 38 00 41 ec 6d 38 00 41 ec 6d 38
00 41 ec 6d 38 00 41 ec 6d 38 00 41 ec 6d 38 00 41 ec 6d 38 00 41 ec 6d 38 00 41 ec 6d 38 00 41 ec
6d 38 00 41 ec 6d 38 00 41 ec 6d 38 00 41 ec 6d 38 00 41 ec 6d 38 00 41 ec 6d 38 00 41 ec 6d 38 00
41 ec 6d 38 00 41 ec 6d 38 00 41 ec 6d 38 00 41 ec 6d 38 00 41 ec 6d 38 00 41 ec 6d 38 00 41 ec 6d
38 00 41 ec 6d 38 00 41 ec 6d 38 00 41 ec 6d 38 00 41 ec 6d 38 00 86 43 e4 0a
```

You can see message type is 3 (Response), page number if ff and again, only has 1 index entry of 3 (byte 7).

B0 COMMANDS

As you can see, only some of these have been decoded as to what they are, but the logic for structuring the message works with all of them. More work is needed here to work out what these are by looking at the decoded data chunks.

Command (Hex)	Description	Notes
01		Some sort of zone/device data
02		Some sort of zone/device data
03		Some sort of zone/device data
04		Some sort of zone/device data
05		Some sort of zone/device data
06	INVALID COMMAND	Returned by the Alarm if an invalid command number is requested
07		
08		Some sort of zone data – zones I have are 03, rest is 00
09		Some sort of zone/device data but all 00s
0A		Some sort of zone/device data but all 00s
0B		Some sort of zone/device data but all 00s
0C		Some sort of zone/device data but all 00s
0D		Some sort of zone/device data but all 00s
0E		Some sort of zone/device data but all 00s
0F		Some sort of zone/device data but all 00s
10	NOT A COMMAND	
11		Some sort of zone data but all 00s
12		Some sort of zone data but all 00s
13		Some sort of zone data but all 00s
14		Some sort of zone data but all 00s
15		Some sort of zone data but all 00s
16		Some sort of zone data but all 00s
17	FULL STATUS	This makes the alarm send many B0 messages – most of this list!
18		Some sort of zone data but all 00s
19	BYPASSES	In bits per zone, 1 is bypass, 0 no bypass
1A		Some sort of zone data but all 00s
1B		Some sort of zone data but all 00s
1C		Some sort of zone data but all 00s
1D	ENROLLED	Seems like enrolment data for devices
1E		Some sort of zone data but all 00s
1F	DEVICE TYPES	Shows device id for all device types
20		Some sort of device data
21	ASSIGNED NAMES	Provides id of zone name for zones. See CMD35 0D 00 for names
22	SYSTEM CAPABILITIES	Not fully decoded but shows numbers of zones, keypads, fobs, events, partitions etc that Alarm can support
23		Some sort of system data
24	PANEL STATUS	Give date time and panel status. Not decoded all elements.
25		
26		
27		
28		
29		

2A	EVENT LOG	Not yet decoded but downloads 58 pages of data in 10 byte chunks as an event.
2B		Some sort of system capabilities data
2C		
2D	ZONE TYPES	The type of each zone
2E		
2F		
30		
31		
32		
33		
34		
35	SETTINGS	Provides many different settings info. See Command35 Settings
36		Some sort of log
37		Some sort of single log entry
38		
39		
3A		Some sort of zone info
3B	NOT A COMMAND	
3C	NOT A COMMAND	
3D	ZONE TEMPS	Zone temp from sensors that have temp data
3E	NOT A COMMAND	
3F	NOT A COMMAND	
40		Some sort of device setting info. Has 04 in zone data for each zone in use.
41		
42	SETTINGS	Provides many different settings info. See Command42 Settings
43		Some sort of zone info
44		
45		
46		
47		
48		
49		Some sort of zone info
4A		
4B	ZONE LAST EVENT	Shows last event for each zone. 1 – Open, 2 – Close, 3 – Motion, 4 - checkin
4C		
4D		
4E		Some zone info in bits
4F		Some zone info in bits
50		Some zone info in 2 byte words
51	ASK ME	Sent by alarm when event happens/something changes. Its data is the list of commands to request to get the updated data
52	DEVICE COUNTS	Numbers of devices by type enrolled
53		
54		
55		
56		
57		
58		
59		
5A		
5B		
5C		
5D		

5E		
5F		
60		
61		
62		
63		
64	PANEL SW VERSION	Software version running on panel
65		
66		
67		
68		
69	PANEL EPROM & SW VERSION	Software and eprom version running on panel
6A	KEEP ALIVE	Only sent as a request to panel to keep connection alive. Sent every 30s if not other message traffic
6B		
6C		
6D		
6E		
6F		
70		
71		
72		
73		
74		
75		Some sort of log
76		
77	ZONE BRIGHTNESS	Brightness of zone for sensors that have lux sensors. Not lux value but a 0 – Darkness,
78		
79		
7A		
7B		
7C		
7D		
7E		
7F		
80		

NOTE: 81 -> FF still to do. Many above 80 come back as invalid command.

COMMAND 35 SETTINGS

As can be seen, there are a number here that have been decoded. I have only listed ones that have either been fully or partially decoded. There are more setting values available than is on this list. More work is needed here to work them all out.

Setting (Hex)	**Description**
00 00	Central station account number 1
01 00	Central station account number 2
02 00	Panel serial number
03 00	Central station IP 1
04 00	Central station IP 1 Port
05 00	Central station IP 2
06 00	Central station IP 2 port
07 00	System capabilities
08 00	User codes
0d 00	Zone names lookup
0f 00	Eprom Download code
15 01	Powerlink software version
24 00	Panel EPROM version
28 00	Capabilities – seems same as 07 00
2b 00	Unknown software version - JS700421 v1.0.02
2c 00	Panel Default Version
2d 00	Panel software version
31 00	Assigned zone types
32 00	Assigned zone name id (links to zone names lookup list)
31 01	Unknown software version - JS703645 K20.213
33 00	Something zones
34 00	Not sure – returns MAP08
35 00	Not sure – returns MAP08
36 00	Something zones
37 00	Something 32 of (keypads/fobs?) – all bytes are 0x01
38 00	Something 32 of (keypads/fobs?) – all bytes are 0x01
39 00	Something 8 of (sirens?) - all 8 bytes are 0x01
3c 00	Panel Hardware Version
3d 00	Panel RSU Version
3e 00	Panel boot version
3f 00	No idea but interesting data - 1, 0, 0, 153, 12, 124, 51, 19, 39, 23, 129, 172, 154, 16, 85, 72
40 00	No idea but interesting data - 1, 0, 0, 197, 11, 71, 0, 79, 124, 21, 153, 29, 0, 255, 255, 255
41 00	No idea – data type 9 (don't know what this is) and data 30 31 30 (hex)
42 00	Custom zone names
45 00	Zone name lookup (again)
46 00	Custom zones names (again)
4e 00	Partitions count??
50 01	Seems to be panel text for troubles/alerts
54 00	Installer code
54 01	Panel IP address
55 00	Unknown 4 digit code – Duress??
56 00	Guard code
61 00	Not sure – returns 153, 152 (int), 99 98 (hex)
71 01	Panel serial
7b 01	Maybe max user codes?
80 00	GPRS APN – data type says int but it is string

81 00	GPRS User – data type says int but string
82 00	GPRS Password – data type says int but string
89 01	Unknown email address
8a 01	Unknown password
8c 00	Central reporting receiver 1 phone number
8d 00	Central reporting receiver 2 phone number
8e 00	Central reporting receiver 1 & 2 SMS No
9d 01	DO NOT USE – Alarm went mad sending data and had to be repowered to reconnect again!
A6 00	Unknown
A7 00	Max zones maybe – returns 64
A8 01	ftp address
A9 01	ftp user
Aa 00	Second download code – shows as int but string (BBBB)
Aa 01	ftp password
E2 00	Not sure – returns 48 bytes – first is 7, rest are 1
E5 00	User codes (again)
E9 00	Accepted chars upper
Ea 00	Accepted chars lower
eb 00	Zone related - investigate
Fc 00	Don’t know – returns 7 – I have 7 zones

COMMAND 42 SETTINGS

Whilst the basic structure decoding applies to command 42 settings, they may have further data that needs decoding within each message. With some small experimentation, I am seeing many of the settings codes produce the same as Command 35, but with a different data formatting.

I.e

00 00

Command 35 – 001235

Command 42 - '02', '00', '18', '00', '00', '00', '01', '1c', '00', '00', '01', '00', '00', '12', '35'

Admittedly, some of the difference is that I am not yet able to decipher format of 42 messages but there is a significant amount of data before the value, which is the same as the command 35.

DATA INDEXES

These have been guessed from the devices/capabilities I have on my PowerMaster 30 panel. Clearly some need working out and it is possible that these are not 100% correct.

Index	**Description**
0	Repeaters
1	X10
2	Sirens
3	Zones
4	Keypads
5	Keyfobs
6	User codes
7	
8	
9	
10	Proximity Tags
11	
12	PGM
13	Powerlink
14	Partitions
15	
16	
17	Events
18	
19	
20	
255	System/None

COMMAND 35 DATA TYPES

The following are what I think the data types are based on using this basis to decode and (in the main) getting sense making results.

Note: I have not fully validated this against any other type of message but seems only command 35 uses this. Command 42 seems to be different.

Data Type	Description	Notes
0	Zero padded string	
1	Direct map string	The hex values are the string
2	Not seen	
3	Double int (little endian)	Can be list of integers
4	Integer	Can be list of integers
5	Not seen	
6	ASCII String	No spaces to strip
7	Don’t know	Seen on Cmd 35 2a 00 but all ff so not sure
8	Space padded ASCII string	Strip spaces
9	Don’t know	See on cmd 35 41 00 – data was 30 31 30 (hex)
10	Space padded ASCII string list	List separation denoted by spaces