M1N1IB storage device manual

Congratulations on buying the M1N1IB storage device. The M1N1IB storage device is your small storage companion that can be carried in your pocket and accessed locally or even connected to the internet.

The M1N1IB storage device features 5 independent slots which chain multiple sectors to store data. Every slot can hold up to 65535 bytes of raw data that represents a file of any format over multiple sectors of up to 12 bytes in size.

We hope that you will enjoy your M1N1IB storage device.

Summary

Device communication options

The M1N1IB storage device can be accessed in a number of ways. Supported access:

- Local access *
  - Firewire
  - USB 2.0
  - USB 3.0
- Network access
  - TCP
  - HTTP **

This manual informs on how to communicate with the device over a TCP connection. Note that network access is limited to read operations only due to security reasons.

** Please refer to the separate access manual for more information on local access.*

*** HTTP is supported in alpha state, please refer to the separate network manual for more information.*

TCP communication

After enabling the TCP communication over the M1N1IB Device Control Panel, the device will listen on its eth0 interface binding to 0.0.0.0 on port 9000. The TCP packet data size of the request and response is fixed to maximum 16 bytes.

TCP request format specification

The M1N1IB device protocol specifies a static request format with possibility of additional request data that depends on the request being issued to the device.

Request format

RQ	Session ticket (5 bytes)	OC	OS	Additional request data (up to 8 bytes)*

- RQ (1 byte) – indication of a request
- OC (1 byte) – operation category
- OS (1 byte) – operation sequence

Fixed bytes in a request will always be: RQ, Session ticket (except when initiate session operation is performed), OC and OS. The RQ byte is a fixed byte and will always correspond to 0x52. The operation category and operation sequence bytes depend on the operation being performed. The bytes needed to perform a certain device operation is listed in the Operations section under each operation. Some operations require sending additional request data. The additional request data is specified under the Operations section under each operation.

TCP response status codes and error messages

Every response sent by the device will note one of the TCP status codes. In essence, the TCP status code can either be an “OK” or an “ERROR” for which a certain error message may be present.

The supported status codes are:

- STATUS_OK – 0x6f
- STATUS_ERR – 0x77

Note that in case of a STATUS_OK, the remaining 15 bytes will correspond to a response format as specified by the operation being performed.

In case of a STATUS_ERR, the response format is:

Error response format

ST	ET	Optional error message (14 bytes)

- ST (1 byte) – status code, in case of STATUS_ERR corresponds to 0x77
- ET (1 byte) – error code

The response will contain a status code and an error code at a minimum. The remaining may or may not hold an optional error message or trace for users to identify the error. Note that error codes itself may already suggest the more information on the error.

Error codes

Possible error codes in a STATUS_ERR are:

- ERR_MESSAGE – 0x6d
  - indicates an issue with message sent by client (size, format, etc.)
- ERR_DEVICE – 0x63
  - indicates an issue with the device performing the operation

Supported operations over TCP

Session ticket operations

Operation category: *0x73*

Initiate session

Operation sequence: *0x69*
Additional request data:

- *None*

In order to perform device operations you must acquire a session ticket. Session tickets are destroyed after 5 minutes of inactivity if not destroyed by developers themselves. It is suggested that developers destroy session tickets when they are no longer needed rather then let them expire due to possible memory-in-use increase on the device.

Response format

ST	Session ticket (5 bytes)	

- ST (1 byte) – status code

Destroy session

Operation sequence: *0x64*
Additional request data:

- *Session ticket (5 bytes)*

Destroys previously initiated session.

Response format

ST	Session ticket (5 bytes)	

- ST (1 byte) – status code

Read operations

Operation category: *0x72*

Reading entry slots metadata

Operation sequence: *0x65*
Additional request data:

- *None*

Response format

ST	S1	S2	S3	S4	S5

- ST (1 byte) – status code
- S# (3 bytes) – entry slot meta information

Entry slot S# 3-byte layout

Entry slot metadata (1 byte)	Entry slot start sector address (2 bytes)

Entry slot metadata byte bits layout

SK	Size of data stored in the slot (7 bits)

- SK (1 bit) – indicates whether there is data stored in the slot

Reading sectors

Operation sequence: *0x73*
Additional request data:

- *Sector address (2 bytes)*

Response format

ST	SF	Sector data (12 or 8 bytes)*	*Metadata (4 bytes)**	NS

- ST (1 byte) – status code
- SF (1 byte) – sector flags
- NS (2 bytes) – address of next sector

Sector data can include both data and additional sector metadata introduced by developers. If no metadata is present, sector data can contain up to 12 bytes in one sector, otherwise sector data length is reduced to 8 bytes (8 bytes data and 4 bytes sector metadata). Refer to sector flags (SF) to get information whether the sector contains only data or both data with metadata.

Sector flags (SF) byte bit layout

R	R	L	M	N	B	R	R

- R – reserved bit
- L – lock flag, indicates if sector is locked for writing
- M – metadata flag, indicates if sector contains metadata
- N – next sector flag, indicates whether next sector exists or not
- B – bad sector flag, indicates whether sector might be corrupted now or in near future