

The controller connects to the server via HTTP (s) and the POST method sends its data. The server in response to the request sends control messages to the controller.

If the server supports ONLINE access checking, then when the card is presented, a request is made to the server to verify permission to pass. In this case, all other functions (events, recording cards, etc.) work.

If ONLINE checks are impossible, the controller (server is unavailable or returns an error) goes to the OFFLINE mode and works with the cards recorded in its memory. Periodically controller checks server availability

1. Messages sent by the controller

All parcels are as follows:

```
{
  "type": "Z5RWEB",
  "sn": 50001,
  "messages": [
    {
      "id": 10,
      ...
    },
    {
      "id": 20,
      ...
    },
    ...
    {
      "id": N,
      ...
    }
  ]
}
```

type - controller type

sn - controller serial number

messages - array of messages from the controller

id - unique message identifier

1.1 POWER_ON

It is sent at the first connection after powering the controller and continues to be sent until the server sends [SET_ACTIVE](#)

Request:

```
{
  "id": 123456789,
  "operation": "power_on",
  "fw": "1.0.1",
  "conn_fw": "2.0.2",
  "active": 0,
  "mode": 0,
  "controller_ip": "192.168.0.222"
}
```

operation - operation name

fw - controller firmware version

conn_fw - communication module firmware version
active - indication of controller activation
mode - controller operation mode (see SET_MODE)
controller_ip - IP address of the controller in the
local network

Answer:

No answer (see SET_ACTIVE)

1.2 CHECK_ACCESS

Sent by the controller in ONLINE mode to check access when the card is brought to the reader

Request:

```
{
  "id": 123456789,
  "operation": "check_access",
  "card": "00B5009EC1A8",
  "reader": 1
}
```

card - card number in hex (see APPENDIX 2)

reader - the reader to which the card is attached. 1 - input, 2 - output.

Answer:

```
{
  "id":123456789,
  "operation": "check_access",
  "granted":1
}
```

granted - 1 - passage allowed, 0 - is prohibited

1.3 PING

Sent periodically in the absence of events. The transmission interval is configured in the WEB interface.

Request:

```
{
  "id": 123456789,
  "operation": "ping",
  "active": 1,
  "mode": 0
}
```

active, mode - look POWER_ON

Answer:

Messages for the controller..

1.4 EVENTS

Sent when new events appear in the controller. If the answer is "success" = N, N events are considered processed. If the answer is "success" = 0 or no response, sending is repeated.

Request:

```
{
  "id": 123456789,
  "operation": "events",
  "events": [
```

```

    {
      "event": 4,
      "card": "00B5009EC1A8",
      "time": "2015-06-25 16:36:01",
      "flag": 0
    },
    {
      "event": 16,
      "card": "00BA00FE32A2",
      "time": "2015-06-25 16:36:02",
      "flag": 0
    }
  ]
}

```

events - array of events

event - event type (see APPENDIX 1)

card - card number in hexadecimal format (for events with a card) (see APPENDIX 2)

time - event time

flag - event flags (for events with flags)

Answer:

```

{
  "id":123456789,
  "operation": "events",
  "events_success":2
}

```

events_success - number of successfully received events

2. Messages sent by the server

Parcels from the server are as follows:

```

{
  "date": "2017-07-25 10:20:30",
  "interval": 10,
  "messages": [
    {
      "id": 10,
      ...
    },
    {
      "id": 20,
      ...
    },
    ...

    {
      "id": N,
      ...
    }
  ]
}

```

date - current time on the server. the time in the controller is set equal to the time sent

interval - the period between connections set by the server

2.1 SET_ACTIVE

Activates / deactivates the controller to the server. A non-activated controller does not transmit events and does not accept control parcels.

The server also informs the controller if it supports ONLINE access checking.

Request:

```
{
  "id":123456789,
  "operation":"set_active",
  "active":1,
  "online":1
}
```

active - 1 - controller activation, 0 - deactivation

online - 1 - server supports ONLINE mode, 0 - do not support

Answer:

```
{
  "id":123456789,
  "success ":1
}
```

success - 1- command is received, 0 - error

2.2 OPEN_DOOR

Causes the output stage to trigger in the specified direction.

Request:

```
{
  "id":123456789,
  "operation":"open_door",
  "direction": 0
}
```

direction - 0 - entrance, 1 - exit

Answer:

```
{
  "id":123456789,
  "success ":1
}
```

2.3 SET_MODE

Sets the mode of the controller (Normal, Easy access, Blocking, Waiting for Easy access)

Request:

```
{
  "id":123456789,
  "operation":"set_mode",
  "mode": 2
}
```

possible modes: 0 - norm, 1 - block, 2 - free passage, 3 - waiting for a free passage

Answer:

```
{
  "id":123456789,
  "success ":1
}
```

```
}
```

2.4 SET_TIMEZONE

Sets the parameters of the time zone of the controller

Request:

```
{
  "id":123456789,
  "operation":"set_timezone",
  "zone": 0,
  "begin":"00:00",
  "end":"23:59",
  "days":"11111110"
}
```

zone - time zone number (0 - 6)

begin - zone start time

end - zone expiration time

days - zone days mask (0 - zone off, 1 -zone enabled), Monday - 1st

Answer:

```
{
  "id":123456789,
  "success ":1
}
```

2.5 SET_DOOR_PARAMS

Sets the parameters for opening and monitoring the door status

Request:

```
{
  "id":123456789,
  "operation":"set_door_params",
  "open":30,
  "open_control":50,
  "close_control":50 }
}
```

open - time for the signal to open the lock (in 1/10 of a second)

open_control - time to control the opening of the door (in 1/10 of a second)

close_control - time to control the closing of a door (in 1/10 of a second)

Answer:

```
{
  "id":123456789,
  "success ":1 }
}
```

2.6 ADD_CARDS

Adds cards to controller memory. If there is already a card with the same number in the controller's memory, flags and time zones are updated for this card..

Request:

```
{
  "id":123456789,
  "operation":"add_cards",
  "cards": [
    {
      "card": "00B5009EC1A8",
      "flags": 0,

```

```

    "tz": 255
  },
  {
    "card": "00000000FE32A2",
    "flags": 32,
    "tz": 255
  }
]
}

```

cards - array of maps to add

card - card number in hexadecimal (see APPENDIX 2)

flags - flags for the card (8 - blocking card, 32 - short code of the card (three bytes))

tz - map time zones

Answer:

```

{
  "id":123456789,
  "success ":1
}

```

2.7 DEL_CARDS

Removes cards from controller memory

Request:

```

{
  "id":123456789,
  "operation":"del_cards",
  "cards": [
    {"card":"0000000A2BA93"},
    {"card":"0000000A2A18A"}
  ]
}

```

cards - array of cards for deletion, contains card numbers in hexadecimal form (see APPENDIX 2)

Answer:

```

{
  "id":123456789,
  "success ":1
}

```

2.8 CLEAR_CARDS

Removes all cards from controller memory

Request:

```

{
  "id":123456789,
  "operation":"clear_cards"
}

```

Answer:

```

{
  "id":123456789,
  "success ":1
}

```

APPENDIX 1: Event Codes

Event	entrance	Exit	Card number	Flags
open button inside	0	1		
key not found in key bank	2	3		
the key is found, the door is open	4	5	available	
Key found, access not allowed	6	7	available	
Opened by network operator	8	9		
the key is found, the door is locked	10	11	available	
door hacked	12	13		
door timeout	14	15		
Passage took place	16	17	available	
Controller reboot	20			
Nutrition	21			0 - gone 1 - appeared
The door is open	32	33	available	
Door closed	34	35		
Switching modes of operation (see Mode)	37			Mode Flags
Fire events (see Fire)	38			Flags of Fire
Security events (see Security)	39			Security Flags
The pass is not made within the specified time.	40	41	available	
Gateway Entry Done	48	49	available	
Gateway blocked (busy)	50	51	available	
Gateway Access Allowed	52	53	available	
Passage blocked (Antipassback)	54	55	available	

APPENDIX 2: Map Code Conversion

When writing a card to the controller and sending events from the controller (as well as online), the card code is transmitted as a string representing 6 bytes of code in the hexadecimal system. The number of significant bytes depends on the protocol of the reader connected to the controller.

Для протокола “iButton” передаются 6 значащих байт:

```
"card": "665544332211"
```

For the “Wiegand” protocol, 3 significant bytes are transmitted:

```
"card": "000000332211"
```

If the controller has the “Wiegand” protocol enabled, all cards written to it are truncated to 3 bytes.

Em-Marine cards and keyfobs usually spell their number in the format “187,01899”. An example of converting such a number into a hexadecimal string:

```
187 (dec) = 0xBB (hex)
1899 (dec) = 0x076B (hex)
```

Card Code:

```
"card": "000000BB076B"
```

This is not the complete code of the Em-Marine card. The full code is 5 bytes, 2 additional bytes on the card are not printed, but the reader sends them to the controller. Therefore, when writing such a code to the controller, it is necessary to add the short code feature:

```
"flags": 32
```

On the cards and key rings Mifare number is written in the form “AB721582”. This is already a hex representation of 4 bytes of the card number. To write to controllers:

```
"card": "0000AB721582"
```