

Описание протокола обмена данными WEB-JSON (WEB-Socket)

Оглавление

1. Общие сведения	3
1.1 Что такое WEBJSON?	3
1.2 Что такое WebSocket?	3
1.3 Различие между WebSocket и WebJSON.....	3
1.4 Тестовый сервер	4
2. Сообщения, отправляемые контроллером.....	4
2.1 POWER_ON	5
2.2 CHECK_ACCESS	5
2.3 PING.....	6
2.4 EVENTS.....	7
3. Сообщения, отправляемые сервером	8
3.1 SET_ACTIVE	8
3.2 OPEN_DOOR	9
3.3 SET_MODE	10
3.4 SET_TIMEZONE	10
3.5 SET_DOOR_PARAMS	11
3.6 ADD_CARDS	12
3.7 DEL_CARDS	13
3.8 CLEAR_CARDS	13
3.9 READ_CARDS.....	14
4. WebSocket.....	14
4.1 Что передают внутри WebSocket?	14
4.2 Тестовый сервер	15
5. АВТОРИЗАЦИЯ ПО ЛОГИН/ПАРОЛЬ.....	18
ПРИЛОЖЕНИЕ 1: Коды событий.....	19
ПРИЛОЖЕНИЕ 2: Преобразование кодов карт	20
ПРИЛОЖЕНИЕ 3: Описание флагов.....	21
Флаги пожара.....	21
Флаги охраны.....	21
Флаги режима	22

1. Общие сведения

1.1 Что такое WEBJSON?

Точного официального стандарта или протокола под названием «WebJson» не существует, имеется в виду **обычный загрузчик данных JSON через протокол HTTP**. Чаще всего, когда говорят «WebJson», имеют в виду **Web API (REST API)**, который обменивается данными в текстовом формате **JSON**.

Это вообще не новый протокол, а способ общения через привычный вам **HTTP**:

- **Что происходит:** Браузер просит сервер: «Дай данные о пользователе». Сервер возвращает не красивую страницу, а текст `{"name": "Анна", "city": "Минск"}`.
- **Суть:** Это классический «Клиент-Сервер».

1.2 Что такое WebSocket?

WebSocket — это протокол связи, который позволяет установить **постоянное двустороннее соединение** между браузером (или другим клиентом) и сервером.

Если говорить просто: это «длинный провод», который держится открытым, пока вы общаетесь. В отличие от обычного интернета (HTTP), где сервер отвечает только когда его спросили, здесь **сервер может сам отправлять данные клиенту в любой момент**.

1.3 Различие между WebSocket и WebJSON

Ключевое различие между **WebSocket** и **WebJSON** (или JSON поверх HTTP) заключается в **характере соединения и направлении передачи данных**.

На самом деле, «WebJson» — это не отдельный протокол, а скорее **формат данных (JSON)**, который чаще всего передаётся внутри *обычных HTTP-запросов*.

Вот простое сравнение:

1. WebSocket (Полноценный протокол)

Это постоянно открытая «труба» между браузером и сервером.

- **Связь:** Двухнаправленная (сервер сам может слать данные клиенту без запроса).
- **Скорость:** Очень высокая (нет накладных расходов на заголовки HTTP после установки связи).
- **Живучесть:** Одно соединение живет минуты или часы.
- **Трафик:** Минимальный.
- **Когда нужен:** Онлайн-программы, где важна скорость и push-уведомления.

2. Web + JSON (Обычный HTTP REST API)

Это стандартный способ запрос-ответ, как загрузка страницы.

- **Связь:** Однонаправленная (клиент попросил -> сервер ответил -> связь разорвана).
- **Скорость:** Средняя (каждый раз передаются HTTP-заголовки и заново устанавливается соединение).
- **Живучесть:** Доли секунды (на один запрос).
- **Трафик:** Большой (заголовки кук, кэша, типа контента).
- **Когда нужен:** Обычный сайт, мобильное приложение.

Итог. Если вам нужно **в реальном времени** (10 раз в секунду) и чтобы сервер сам присылал обновления — берите **WebSocket**. Если просто списком запросов раз в минуту — берите **HTTP + JSON**.

Контроллер соединяется с сервером по протоколу HTTP(s) и методом POST запросов отправляет имеющиеся у него данные, запросы приходят не в параметрах, а в теле POST запроса, поэтому `_POST` пустая. Надо читать входной поток. Сервер в ответ на запрос присылает управляющие сообщения для контроллера. Размер пакета, отправляемого контроллеру, не должен превышать 2 кБ.

Если сервер поддерживает ONLINE проверку доступа, то при поднесении карты делается запрос на сервер для проверки разрешения на проход. При этом все остальные функции (события, запись карт и т.д.) работают.

При невозможности ONLINE проверки контроллер (сервер недоступен или возвращает ошибку) переходит в OFFLINE режим и работает с картами, записанными в его память. Периодически контроллер проверяет доступность сервера.

1.4 Тестовый сервер

Для тестирования работы контроллера можно использовать тестовый сервер - <https://json5.help8.ru/5/> (адрес вводить в браузере).

В настройках контроллера в web-интерфейсе на вкладке «Режим работы» указать «Web-Json», HTTP и указать адрес <https://json5.help8.ru/>.

Тестовый сервер имеет внешний вид:

Controller	Stop ☐ Interval 10
Server	Pause ☐ Interval 1000
active ☒ online ☒ open ☐	set_active
input ☒ output ☐	open_door
0 normal ▼	set_mode
bank	set_timezone
zone 0	set_door_params
begin 00:00	
end 23:59	
days 11111110	add_cards
open 30	del_cards
open_c 50	get_state
close_c 50	clear_cards
card 00B5009EC1A8	read_cards
flags 32	
tz 255	
card 00B5009EC1A8	

2. Сообщения, отправляемые контроллером

Все сообщения от контроллера имеют строчный вид. Для наглядности этапы запроса расписаны по элементам в столбик:

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

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

type - тип контроллера.

sn - серийный номер контроллера.

messages - массив сообщений от контроллера.

id - уникальный идентификатор сообщения.

2.1 POWER_ON

Сообщение [«power_on»](#) посылается при первом соединении после включения питания контроллера и продолжает посылаться до тех пор, пока сервер не пришлет [«SET_ACTIVE»](#).

Пример сообщений с тестового сервера (синий – сервер, красный – контроллер):

```
{ "date": "2026-04-02 14:46:26", "interval": 10, "messages": [] }
{ "type": "Z5-R WEB BT", "sn": 45010964, "messages": [ { "id": 488369967, "operation": "power_on", "fw": "2.38", "conn_fw": "1.45", "active": 0, "mode": 0, "controller_ip": "10.32.36.130", "auth_hash": "593e737335c89ff739e4445246a40379" } ] }
```

Текстовый пример запроса:

```
{
  "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 - название операции.

fw - версия прошивки контроллера.

conn_fw - версия прошивки модуля связи.

active - признак активированности контроллера.

mode - режим работы контроллера (смотри SET_MODE).

controller_ip - IP адрес контроллера в локальной сети.

Ответ:

Без ответа (смотри SET_ACTIVE).

2.2 CHECK_ACCESS

Сообщение [«check_access»](#) посылается контроллером в режиме ONLINE проверки доступа при поднесении карты к считывателю. Проверяется доступ для карточки, разрешен проход или нет.

Запрос:

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

Пример запроса контроллера:

```
{ "type": "Z5-R WEB BT", "sn": 45001320, "messages": [{"id": 570816802, "operation": "check_access", "card": "004D0077A224", "reader": 1}, {"id": 170285823, "operation": "ping", "active": 1, "mode": 0}] }
```

card - номер карты в шестнадцатеричном виде (см. ПРИЛОЖЕНИЕ 2).

reader - считыватель, к которому приложена карта. 1 - вход, 2 - выход.

Ответ:

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

Пример ответа с тестового сервера:

```
{ "date": "2026-04-10 15:19:10", "interval": 8, "messages": [{"id": 570816802, "operation": "check_access", "granted": 0}] }
```

granted - 1 - проход разрешен, 0 – запрещен.

2.3 PING

Сообщение «ping» посылается контроллером периодически при отсутствии событий. Интервал передачи настраивается в WEB-интерфейсе.

Пример сообщений с тестового сервера (синий – сервер, красный – контроллер):

```
{ "date": "2026-04-02 15:05:40", "interval": 10, "messages": [] }
{ "type": "Z5-R WEB BT", "sn": 45010964, "messages": [{"id": 1598257455, "operation": "ping", "active": 1, "mode": 0}] }
{ "date": "2026-04-02 15:05:51", "interval": 10, "messages": [] }
{ "type": "Z5-R WEB BT", "sn": 45010964, "messages": [{"id": 1777079854, "operation": "ping", "active": 1, "mode": 0}] }
{ "date": "2026-04-02 15:06:01", "interval": 10, "messages": [] }
{ "type": "Z5-R WEB BT", "sn": 45010964, "messages": [{"id": 455161765, "operation": "ping", "active": 1, "mode": 0}] }
{ "date": "2026-04-02 15:06:12", "interval": 10, "messages": [] }
{ "type": "Z5-R WEB BT", "sn": 45010964, "messages": [{"id": 43106934, "operation": "ping", "active": 1, "mode": 0}] }
```

Запрос:

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

active, mode - смотри POWER_ON.

Ответ:

Сообщения для контроллера.

Если сервер несколько раз подряд (до 10 раз) не отвечает контроллеру на запрос "Ping", контроллер считает, что связи нет и отправляет запрос "Power on".

2.4 EVENTS

Сообщение «events» посылается при появлении новых событий в контроллере. При ответе “success” = N, N событий считаются обработанными. При ответе “success” = 0 или отсутствии ответа, повторяется отправка.

Пример ответа контроллера на тестовый сервер:

```
{ "type": "Z5-R WEB BT", "sn": 45010964, "messages": [{ "id": 308, "success": 1 }, { "id": 132615087, "operation": "events", "events": [{ "flag": 0, "event": 8, "time": "2026-04-02 15:05:29", "card": "000000000000"}, { "flag": 0, "event": 16, "time": "2026-04-02 15:05:29", "card": "000000000000"} ], "last_event": 116811 }
```

Запрос:

```
{
  "id": 123456789,
  "operation": "events",
  "events": [
    { "flag": 0,
      "event": 4,
      "time": "2015-06-25 16:36:01",
      "card": "00B5009EC1A8"},
    { "flag": 0,
      "event": 16,
      "time": "2015-06-25 16:36:02",
      "card": "00BA00FE32A2"}
  ],
  "last_event": 3160 - это поле есть только в режиме Socket
}
```

events - массив событий.

event - тип события (смотри ПРИЛОЖЕНИЕ 1).

card - номер карты в шестнадцатеричном виде (для событий с картой) (см. ПРИЛОЖЕНИЕ 2).

time - время события.

flag - флаги события (для событий с флагами) (смотри ПРИЛОЖЕНИЕ 3).

Ответ:

в режиме JSON

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

events_success - количество успешно принятых событий.

в режиме Socket

```
{
  "id": 123456789,
  "operation": "events",
  "events_success": 3160
}
```

events_success – номер последнего принятого сообщения, указан в поле last_event опроса контроллера.

3. Сообщения, отправляемые сервером

Посылки от сервера имеют строчный вид. Для наглядности команды разбиты по элементам:

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

date - текущее время на сервере, время в контроллере устанавливается равным присланному времени.

interval - период между соединениями, установленный сервером.

3.1 SET_ACTIVE

Сообщение [«set_active»](#) активирует/деактивирует работу контроллера с сервером. Не активированный контроллер не передаёт события и не принимает управляющие посылки. Также сервер сообщает контроллеру, поддерживает ли он ONLINE проверку доступа.

```
[{"id":9656,"operation":"set_active","active":1,"online":1,"open":null}]
0
```

Запрос:

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

Пример запроса с тестового сервера:

```
["date":"2026-04-02 14:46:36","interval":8,"messages":[{"id":9656,"operation":"set_active","active":1,"online":1},{"id":488369967,"operation":"set_active","active":1,"online":1}]]
```

active - 1 - активация контроллера, **0** – деактивация.

online - 1 - сервер поддерживает режим ONLINE, **0** - не поддерживает.

Ответ:

```
{
  "id":123456789,
```

```
"success ":1
}
```

Пример ответа контроллера:

```
{"type":"Z5-R WEB BT","sn":45010964,"messages":[{"id":9656, "success":1}
```

success - 1 - команда принята, **0** – ошибка.

События будут передаваться только после команды сервера "set_active". До этого контроллер будет посылать серверу сообщения "power_on". Ничего не посылать контроллеру не получится. Сервер обязан в ответе на POST запрос контроллера присылать валидную команду. Если никаких команд для контроллера нет, то ответ должен иметь вид:

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

Если контроллер не получит валидного ответа, он перейдёт в автономный режим, и опять начнёт слать "power_on".

3.2 OPEN_DOOR

Сообщение «open_door» вызывает срабатывание выходного каскада контроллера в заданном направлении (открытие двери).

```
[{"id":6620,"operation":"open_door","direction":0}]
0
```

Запрос:

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

Пример запроса с тестового сервера:

```
{"date":"2026-04-02 15:05:29","interval":10,"messages":[{"id":308,"operation":"open_door","direction":0}]}
```

direction - 0 - вход, **1** – выход.

Ответ:

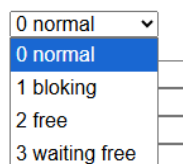
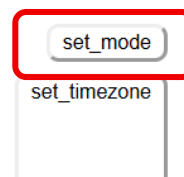
```
{
  "id":123456789,
  "success ":1
}
```

Пример ответа контроллера:

```
{"type":"Z5-R WEB BT","sn":45010964,"messages":[{"id":308, "success":1}
```

3.3 SET_MODE

Сообщение «set_mode» устанавливает режим работы контроллера (Норма, Блокировка, Свободный проход, Ожидание Свободного прохода).

```
[{"id":1454,"operation":"set_mode","mode":2}]
```

Запрос:

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

Пример запроса с тестового сервера:

```
{ "date": "2026-04-10 15:30:37", "interval": 8, "messages": [{"id": 1454, "operation": "set_mode", "mode": 2}] }
```

возможные режимы: 0 - норма, 1 - блок, 2 - свободный проход, 3 - ожидание свободного прохода.

Ответ:

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

Пример ответа контроллера:

```
{ "type": "Z5-R WEB BT", "sn": 45001320, "messages": [{"id": 257, "success": 1}] }
```

3.4 SET_TIMEZONE

Сообщение «set_timezone» устанавливает параметры временной зоны контроллера.

```
[{"id":871,"operation":"set_timezone","zone":0,"begin":"00:00","end":"23:59","days":"11111110"}]
```

Запрос:

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

Пример запроса с тестового сервера:

```
{"date":"2026-04-10 15:33:40","interval":8,"messages":[{"id":871,"operation":"set_timezone","zone":0,"begin":"00:00","end":"23:59","days":"11111110"}]}
```

bank - номер банка памяти при использовании двойных временных зон (0-1) (необязательный параметр, при его отсутствии, запись произойдёт в оба банка).

zone - номер временной зоны (0 - 8) (зоны 7 и 8 - зоны переключения режимов)

begin - время начала действия зоны

end - время окончания действия зоны

days - маска дней недели для зоны (0 - зона выключена, 1 - включена), понедельник - 1-й

mode - режим для переключения режимов (зоны 7 и 8, для остальных зон не используется) **0** - Норма, **1** - Блокировка, **2** - Свободный, **3** - Ожидание.

Ответ:

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

Ответ контроллера на тестовом сервере:

```
{"type":"Z5-R WEB BT","sn":45001320,"messages":[{"id":871,"success":1},
```

3.5 SET_DOOR_PARAMS

Сообщение «[set_door_params](#)» устанавливает параметры открывания и контроля состояния двери.

```
{"id":9032,"operation":"set_door_params","open":30,"open_control":50,"close_control":50}
```

Запрос:

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

Пример запроса с тестового сервера:

```
{"date":"2026-04-10 15:35:17","interval":8,"messages":[{"id":9032,"operation":"set_door_params","open":30,"open_control":50,"close_control":50}]}
```

open - время подачи сигнала открывания замка (в 1/10 секунды).

open_control - время контроля открытия двери (в 1/10 секунды).

close_control - время контроля закрытия двери (в 1/10 секунды).

Ответ:

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

При неверных настройках параметров открытия, если время задано слишком короткое, контроллер может выдать ошибку код 40.

Обратите внимание, что у турникетов бывает импульсное и потенциальное управление. В случае потенциального управления "время импульса открытия" и "время ожидания прохода" должны быть равны. В случае импульсного управления "время ожидания прохода" должно быть равно времени, заданному в турникете.

3.6 ADD_CARDS

Сообщение «[add_cards](#)» добавляет карты в память контроллера. Если в памяти контроллера уже имеется карта с таким-же номером, для этой карты обновляются флаги и временные зоны.

```
[{"id":9726,"operation":"add_cards","cards":[{"card":"004D0077A224","flags":0,"tz":0}]}
```

Запрос:

```
{
  "id":123456789,
  "operation":"add_cards",
  "cards": [
    {
      "card": "00B5009EC1A8",
      "flags": 0,
      "tz": 255
    },
    {
      "card": "0000000FE32A2",
      "flags": 32,
      "tz": 255
    }
  ]
}
```

Пример запроса с тестового сервера:

```
{"date":"2026-04-10 15:40:38","interval":10,"messages":[{"id":9726,"operation":"add_cards","cards":[{"card":"004D0077A224","flags":0,"tz":0}]}]}
```

Ответ:

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

Ответ контроллера на тестовом сервере:

```
{"type":"Z5-R WEB BT","sn":45001320,"messages":[{"id":9726,"success":1}]}
```

cards - массив карт для добавления.

card - номер карты в шестнадцатеричном виде (см. ПРИЛОЖЕНИЕ 2).

flags - флаги для карты (8 - блокирующая карта, 32 - короткий код карты (три байта)).

tz - временные зоны для карты (битовая маска разрешённых для карты зон).

Добавление карты по временным зонам(**tz**):

Всего существует семь временных зон: от 0 до 6, хотим задать зоны 1,2,6:

1(dec) = 00000001(bin) - нулевая зона

2(dec) = 00000010(bin) - первая зона

4(dec) = 00000100(bin) - вторая зона
 8(dec) = 00001000(bin) - третья зона
 16(dec) = 00010000(bin) - четвертая зона
 32(dec) = 00100000(bin) - пятая зона
 64(dec) = 01000000(bin) - шестая зона
 255(dec) = 11111111(bin) - полный доступ

Соответственно для 1,2,6 зон tz = 01000110(bin) = 70(dec)

3.7 DEL_CARDS

Сообщение «sel_cards» удаляет конкретную карту из памяти контроллера. Номер конкретной карты указывается на сервере.

```
[{"id":7424,"operation":"del_cards","cards":[{"card":"004D0077A224"}]}
```

Запрос:

```
{
  "id":123456789,
  "operation":"del_cards",
  "cards": [
    {"card":"000000A2BA93"},
    {"card":"000000A2A18A"}
  ]
}
```

Пример запроса с тестового сервера:

```
{
  "date":"2026-04-10 15:46:49",
  "interval":10,
  "messages":[{"id":7424,"operation":"del_cards","cards":[{"card":"004D0077A224"}]}]}
}
```

cards - массив карт для удаления, содержит номера карты в шестнадцатеричном виде (см. ПРИЛОЖЕНИЕ 2).

Ответ:

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

Ответ контроллера с тестового сервера:

```
{
  "type":"Z5-R WEB BT",
  "sn":45001320,
  "messages":[{"id":7424,"success":1}]
}
```

3.8 CLEAR_CARDS

Сообщение «clear_cards» удаляет все карты из памяти контроллера.

```
[{"id":2817,"operation":"clear_cards"}]
```

Запрос:

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

Пример запроса с тестового сервера:

```
{ "date": "2026-04-10 15:48:15", "interval": 10, "messages": [{ "id": 2817, "operation": "clear_cards" }] }
```

Ответ:

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

Ответ контроллера с тестового сервера:

```
{ "type": "Z5-R WEB BT", "sn": 45001320, "messages": [{ "id": 2817, "success": 1 }, ] }
```

3.9 READ_CARDS

Сообщение «read_cards» вычитывает все карты из памяти контроллера.

```
{ "id": 6975, "operation": "read_cards" }
```

Запрос:

```
{ "id": 12312,
  "operation": "read_cards"
}
```

Пример запроса с тестового сервера:

```
{ "date": "2026-04-10 15:44:20", "interval": 10, "messages": [{ "id": 6975, "operation": "read_cards" }] }
```

Ответ:

```
"cards": [
  { "pos": 0, "card": "000000DDD2DC", "flags": 0, "tz": 255 },
  { "pos": 1, "card": "000000030201", "flags": 0, "tz": 255 }
]
```

Ответ контроллера с тестового сервера:

```
{ "type": "Z5-R WEB BT", "sn": 45001320, "messages": [{ "operation": "read_cards", "last_card": 8, "cards": [ { "pos": 0, "card": "0000003372B0", "flags": 32, "tz": 255 }, { "pos": 1, "card": "0000006203D7", "flags": 32, "tz": 255 }, { "pos": 2, "card": "00B5009EC1A8", "flags": 32, "tz": 255 }, { "pos": 3, "card": "0072006203D7", "flags": 0, "tz": 255 }, { "pos": 4, "card": "00000005A097", "flags": 32, "tz": 255 }, { "pos": 5, "card": "7BB2667BB26E", "flags": 4, "tz": 255 }, { "pos": 6, "card": "004D0077A224", "flags": 32, "tz": 255 }, { "pos": 7, "card": "0000D3043BAE", "flags": 0, "tz": 1 } ] } ] }
```

4. WebSocket

4.1 Что передают внутри WebSocket?

WebSocket — это протокол связи, который позволяет установить **постоянное двустороннее соединение** между браузером (или другим клиентом) и сервером.

Как это работает (по шагам)

1. **Рукопожатие (Handshake):** Клиент посылает серверу специальный HTTP-запрос: «Давай переходить на WebSocket».
2. **Апгрейд (Upgrade):** Сервер отвечает: «Ок, переключаемся». С этого момента обычный HTTP-протокол сменяется на WebSocket по тому же самому соединению.

3. **Постоянная связь:** Соединение остается открытым. Теперь оба могут слать сообщения туда-сюда без задержек.
4. **Заккрытие:** Любая сторона может закрыть соединение (например, уходя со страницы).
Протокол работает с любыми данными: текст, бинарные файлы, но чаще всего внутри летает всё тот же **JSON** (чтобы было понятно и структурировано).

Краткий итог:

- **WebSocket** нужен тогда, когда данные нужны «здесь и сейчас», без постоянного дерганья сервера.
- Обычный интернет (**HTTP**) подходит для всего статического: загрузки картинок, переходов по ссылкам, отправки формы раз в минуту.

4.2 Тестовый сервер

На сайте ironlogic.ru в разделе «Интеграция - Режим Web-JSON» необходимо скачать [файл тестового сервера](#).

Полезные ссылки

[Ссылка на файл тестового сервера для windows json-socket](#)

Записи вебинаров:

["Режим Web-JSON для контроллеров Matrix-II Wi-Fi и Z-5R Web"](#)

["Режимы Web Json и WebSocket"](#)

Презентации:

["Режим Web-JSON для контроллеров Matrix-II Wi-Fi и Z-5R Web"](#)

guardlight_10a_230_setup.zip

app.exe

9535bf1a-63bd-4ddb-9189-5e632d9d77a..

После запуска «app.exe» откроется командное окно с информацией и сообщениями контроллера и сервера. В первой строке необходимо скопировать URL Сервера (ws://10.24.9.251:9000) и вставить его в соответствующее поле в Web-интерфейсе контроллера.

```
Выбрать C:\Downloads\app.exe
Версия 3.1.1
For cotrollers http://10.32.36.177:8080 or ws://10.32.36.177:9000
For admin http://10.32.36.177:8080/simulator/admin.html
Simulator of controller http://10.32.36.177:8080/simulator/index.html
Новый пользователь
controller >> server 45010964 - {"type":"Z5-R WEB BT","sn":45010964,"messages":[{"id":323060340,"operation":"power_on","fw":"2.38","conn_fw":"1.45","active":0,"mode":0,"controller_ip":"10.32.36.130","auth_hash":"593e737335c89ff739e444524640379"}]}
sn - 45010964; id - 0
answer server >> controller 45010964 - {"date":"2026-03-26 09:55:35","interval":8,"messages":[{"operation":"set_active","id":2,"active":1,"online":0}]}
controller >> server 45010964 - {"type":"Z5-R WEB BT","sn":45010964,"messages":[{"id":2,"success":1}]}
answer server >> controller 45010964 - {"date":"2026-03-26 09:55:35","interval":8,"messages":[]}
```

iron iL Logic®

Z-5R WEB BT
SN: 45-010964

Русский ☒
English ☐

Статус

Настройки Соединения

Режим Работы

Общие Настройки

Ключ аутентификации: EB2E0BF9

Режим Работы: WEB-JSON

WEB-JSON

Protocol: ☐ HTTP ☒ Websocket

URL Сервера: ws://10.32.36.177:9000

Интервал: 10

Сохранить

Настройки Контроллера

Расширенные Настройки

Вторая строка – адрес сервера для админа, который нужно ввести в браузере (<http://10.4.9.251:8080/simulator/admin.html>).

```

Выбрать C:\Downloads\app.exe
Версия 3.1.1
For controllers http://10.32.36.177:8080 or ws://10.32.36.177:9000
For admin http://10.32.36.177:8080/simulator/admin.html
Simulator of controller http://10.32.36.177:8080/simulator/index.html
Новый пользователь
controller >> server 45010964 - {"type":"Z5-R WEB BT","sn":45010964,"messages":[{"id":323060340,"operation":"power_on","fw":"2.38","conn_fw":"1.45","active":0,"mode":0,"controller_ip":"10.32.36.130","auth_hash":"593e737335c89ff739e4445246a40379"}]}
sn - 45010964; id - 0
answer server >> controller 45010964 - {"date":"2026-03-26 09:55:35","interval":8,"messages":[{"operation":"set_active","id":2,"active":1,"online":0}]}
controller >> server 45010964 - {"type":"Z5-R WEB BT","sn":45010964,"messages":[{"id":2,"success":1}]}
answer server >> controller 45010964 - {"date":"2026-03-26 09:55:35","interval":8,"messages":[]}

```

В браузере будет отображаться панель с командами для контроллера, аналогичными как из раздела 1.

← → ↻ 10.32.36.177:8080/simulator/admin.html ☆ 📄 ⬇️ 🔗

Контроллер	Z5-R Web	45010964	153	230	192.168.0.222	2021-05-08 11:00:53
Настройки	0	0	0	1	8	
Текущая Карта	123456789012	0	0	0		
Тайм Зона		0	00:00	23:59	1111111	

Контроллеры Карты Логи Состояние Сервер Сброс Старт Стоп set_active open_door set_mode set_timezone set_door_params add_cards read_cards del_cards clear_cards

Лог 45010964

```

C:\Downloads\app.exe
Версия 3.1.1
For controllers http://10.32.36.177:8080 or ws://10.32.36.177:9000
For admin http://10.32.36.177:8080/simulator/admin.html
Simulator of controller http://10.32.36.177:8080/simulator/index.html
Новый пользователь
controller >> server 45010964 - {"type":"Z5-R WEB BT","sn":45010964,"messages":[{"id":614763842,"operation":"power_on","fw":"2.38","conn_fw":"1.45","active":0,"mode":0,"controller_ip":"10.32.36.130","auth_hash":"593e737335c89ff739e4445246a40379"}]}
sn - 45010964; id - 0
answer server >> controller 45010964 - {"date":"2026-03-26 11:01:11","interval":8,"messages":[{"operation":"set_active","id":2,"active":1,"online":0}]}
controller >> server 45010964 - {"type":"Z5-R WEB BT","sn":45010964,"messages":[{"id":2,"success":1}]}
answer server >> controller 45010964 - {"date":"2026-03-26 11:01:11","interval":8,"messages":[]}

```

Протокол WebSocket полностью повторяет по запросам WEB-JSON, кроме следующих пунктов:

В сообщении "events" добавлен параметр "last_event". В ответном сообщении надо присылать "events_success": < last_event >:

```

{
  "type": "Z5RWEB",
  "sn": 48276,
  "messages": [
    {
      "id": 1714636915,
      "operation": "events",
      "events": [
        {
          "flag": 0,
          "event": 5,
          "time": "2019-12-19 12:07:33",
          "card": "0000000030201"
        }
      ]
    }
  ]
}

```

```

},
{
  "flag": 0,
  "event": 17,
  "time": "2019-12-19 12:07:33",
  "card": "000000030201"
}
],
"last_event": 3160
}
]
}

```

Добавлена операция чтения всех карт “operation”:read_cards”. Карты отправляются «пакетами» по 11 штук (“pos”: 0 – “pos”: 10 и т.д.), т.к. есть ограничения.

Пакеты могут приходить в разноряд, так как сервер может пропустить пакет. После команды READ_CARDS сервер только слушает и ничего не останавливает. Как понять, что все карты пришли и пакетов больше не будет: для проверки записать все пакет и посмотреть последовательность и все ли карты пришли = сколько их.

Пример команды чтения карт:

```

{
  "date": "2019-12-25 10:20:32",
  "messages": [
    {
      "id": 12312,
      "operation": "read_cards"
    }
  ]
}

```

В ответ присылается одно или несколько сообщений вида:

```

{
  "type": "Z5RWEB",
  "sn": 50001,
  "messages": [
    {
      "cards": [
        {
          "pos": 0,
          "card": "000000DDD2DC",
          "flags": 0,
          "tz": 255
        },
        {
          "pos": 1,
          "card": "000000030201",
          "flags": 0,
          "tz": 255
        }
      ]
    }
  ]
}

```

5. АВТОРИЗАЦИЯ ПО ЛОГИН/ПАРОЛЬ

Аутентификация на сервере: Логин и пароль вводятся в адресной строке.

ext:12345@192.168.0.28

ext= > логин

12345 => пароль доступа к API

192.168.0.28 => IP адрес

Z5-R WEB-JSON Authorization

->

POST / HTTP/1.1

Host: 10.5.0.2:8888

Accept: */*

User-Agent: Z5R WEB

Conection: close

Content-type: application/json

Content-Length: 189

{.....}

<-

HTTP/1.0 401 Unauthorized

Server: BaseHTTP/0.6 Python/3.9.13

Date: Thu, 11 Aug 2022 11:44:33 GMT

WWW-Authenticate: Basic realm="Demo Realm"

Content-type: application/json

->

POST / HTTP/1.0

Authorization: Basic dXNlcjpwYXNzd29yZA==

Host: 10.5.0.2:8888

Accept: */*

User-Agent: Z5R WEB

Conection: close

Content-type: application/json

Content-Length: 189

{.....}

<-HTTP/1.0 200 OK

Server: BaseHTTP/0.6 Python/3.9.13

Date: Thu, 11 Aug 2022 11:44:33 GMT

Content-type: application/json

{.....}

Сервер должен отправить ответ со статусом ошибки 401 и установить заголовок "WWW-Authenticate: Basic realm="Demo Realm"". Это значит, что сервер ждет использования аутентификации вида «Basic», это когда с запросом всегда передается логин и пароль. Указываться может в адресе запроса или в заголовках.

ПРИЛОЖЕНИЕ 1: Коды событий

Событие	дес.		hex		Флаги
	Вход	Выход	Вход	Выход	
Открыто кнопкой изнутри	00	01	0x00	0x01	
Ключ не найден в банке ключей	02	03	0x02	0x03	
Ключ найден, дверь открыта	04	05	0x04	0x05	
Ключ найден, доступ не разрешен	06	07	0x06	0x07	
Открыто оператором по сети	08	09	0x08	0x09	
Ключ найден, дверь заблокирована	10	11	0x0A	0x0B	
Попытка открыть заблокированную дверь кнопкой	10	11	0x0A	0x0B	
Дверь взломана	12	13	0x0C	0x0D	
Дверь оставлена открытой (timeout)	14	15	0x0E	0x0F	
Проход состоялся	16	17	0x10	0x11	
Сработал датчик 1	18		0x12		Состояние
Сработал датчик 2	19		0x13		Состояние
Перезагрузка контроллера	20		0x14		
Питание	21		0x15		0 – пропало 1 – появилось
Заблокирована кнопка открывания	22	23	0x16	0x17	
Было (Антипассбэк) уехал на 36/37	26	27	0x1A	0x1B	
Замок включен (режим Триггер)	28	29	0x1C	0x1D	
Замок выключен (режим Триггер)	30	31	0x1E	0x1F	
Дверь открыта	32	33	0x20	0x21	
Дверь закрыта	34	35	0x22	0x23	
Управление питанием (см Электро)	36		0x24		Флаги Электро
Переключение режимов работы (см Режим)	37		0x25		Флаги Режимов
Пожарные события (см Пожар)	38		0x26		Флаги Пожара
Охранные события (см Охрана)	39		0x27		Флаги Охраны
Проход не совершён за заданное время	40	41	0x28	0x29	
Совершен вход в шлюз	48	49	0x30	0x31	
Заблокирован вход в шлюз (занят)	50	51	0x32	0x33	
Разрешен вход в шлюз	52	53	0x34	0x35	
Заблокирован проход (Антипассбек)	54	55	0x36	0x37	
Hotel (Изменение режима работы)	64		0x40		Флаги (описание)
Hotel (Отработка карт)	65		0x41		Флаги (описание)
Номер ключа (перед некоторыми событиями, которые вызвал ключ)	85		0x55		0, Данные ключа
Номер ключа 7 байт (перед некоторыми событиями, которые вызвал ключ)	86		0x56		7 байт ID ключа

ПРИЛОЖЕНИЕ 2: Преобразование кодов карт

При записи карты в контроллер и передачи событий от контроллера (а также в режиме online), код карты передаётся в виде строки, представляющей 6 байт кода в шестнадцатеричной системе. Количество значащих байт зависит от протокола считывателя, подключённого к контроллеру.

Для протокола “iButton” передаются 6 значащих байт:

"card": "665544332211"

Для протокола “Wiegand 26” передаются 3 значащих байт:

"card": "000000332211"

Если у контроллера включен протокол “Wiegand”, все записываемые в него карты обрезаются до 3х байт.

На картах и брелоках Em-Marine обычно пишется их номер в формате “187,01899”.

Пример преобразования такого номера в шестнадцатеричную строку:

187 (dec) = 0x**BB** (hex)

1899 (dec) = 0x**076B** (hex)

Код карты:

"card": "000000**BB076B**"

Это не полный код карты Em-Marine. Полный код составляет 5 байт, 2 дополнительных байта на карте не напечатаны, но считыватель передаёт их в контроллер. Поэтому при записи такого кода в контроллер надо добавить признак короткого кода:

"flags": 32

На картах и брелоках Mifare номер пишется в виде "AB721582". это уже шестнадцатеричное представление 4х байт номера карты.

Для записи в контроллеры:

"card": "0000AB721582"

ПРИЛОЖЕНИЕ 3: Описание флагов

Обратите внимание!

В сообщениях JSON числа передаются в десятичном виде. Для определения значения флага, необходимо, предварительно перевести число в шестнадцатеричный формат.

Например: 771 это 0x0303, два байта: 0x03 и 0x03. 0x03 это 0b0011 (в двоичном формате).

Первый байт

Бит0 – состояние пожарного режима – 1 вкл

Бит1 – активен пожарный режим по входу FIRE

Второй байт

0x03 – включено по входу FIRE

Флаги пожара

1-й байт флагов пожара - флаги состояния:

Бит0 – состояние пожарного режима – 1 вкл/0 выкл

Бит1 – активен пожарный режим по входу FIRE

Бит2 – активен пожарный режим по превышению температуры

Бит3 – активен пожарный режим по внешней команде

2-й байт флагов пожара – код условия вызвавшего срабатывание

0x00 – выключено по сети

0x01 – включено по сети

0x02 – выключено по входу FIRE

0x03 – включено по входу FIRE

0x04 – выключено по датчику температуры

0x05 – включено по датчику температуры

Флаги охраны

1-й байт флагов охраны – флаги состояния

Бит0 – состояние охранного режима – 1 вкл/0 выкл

Бит1 – состояние тревоги

Бит2 – тревога по входу ALARM (шлейф охраны)

Бит3 – тревога по тамперу

Бит4 – тревога по датчику двери

Бит5 – состояние «сетевой тревоги»

2-й байт флагов охраны – код условия вызвавшего срабатывание

0x00 – выключено по «сетевой тревоге»

0x01 – включено по «сетевой тревоге»

0x02 – выключено по входу ALARM

0x03 – включено по входу ALARM

0x04 – выключено по тамперу

0x05 – включено по тамперу

0x06 – выключено по датчику двери

0x07 – включено по датчику двери

Пример:

Флаг 1291 (dec) = 0x050B

1й байт = 0x0B = 00001011(bin)

- > Бит0 – состояние охранного режима – 1 вкл/0 выкл

- > Бит1 – состояние тревоги

Бит2 – тревога по входу ALARM (шлейф охраны)

- > Бит3 – тревога по тамперу

Бит4 – тревога по датчику двери

Бит5 – состояние «сетевой тревоги»

2й байт = 0x05

0x00 – выключено по «сетевой тревоге»

0x01 – включено по «сетевой тревоге»

0x02 – выключено по входу ALARM

0x03 – включено по входу ALARM

0x04 – выключено по тамперу

- > 0x05 – включено по тамперу

0x06 – выключено по датчику двери

0x07 – включено по датчику двери

Флаги режима

1-й байт флагов режима – текущий режим (с флагами активизации):

Биты 0 и 1 – Активный режим (0...3)

Бит 2 – включен по временной зоне

Бит 3 – включен командой по сети

Бит 4 – включен картой

Бит 5 – включен внешним сигналом (входом)

2-й байт флагов режима – причины переключения режима:

0x01 .. 0x04 – Установка командой по сети (режим +1)

0x81 .. 0x84 – Отказано оператору по сети

0x05 – Началась временная зона

0x06 – Окончилась временная зона

0x08 – Установка картой

0x88 – Отказано изменению картой

0x09 – Установка внешним сигналом (входом)