EasyHome HTTP API

v 2.5

Оглавление

Общая информация

Система EasyHome имеет возможность управления посредством HTTP 1.1 POST запросов в формате JSON. С версии API 2.2 появилась возможность получать состояние системы используя GET запросы.

Для корректной работы обязательными HTTP заголовками являются «Host» и «Content-Length», остальные заголовки опциональны. Общая длина запроса вместе с заголовками должна быть не менее 60 символов для правильной его обработки.

Порт, предназначенный для API, настраивается через инженерный интерфейс в разделе «расширения EasyHome» и по умолчанию равен 3502.

Пример команды, сформированной через программу Postman:

```
POST / HTTP/1.1
Host: 192.168.1.205:3502
Connection: keep-alive
Content-Length: 204
User-Agent: Mozilla/5.0 (Windows NT 6.1; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko)
Chrome/72.0.3626.121 Safari/537.36
Cache-Control: no-cache
Origin: chrome-extension://fhbjgbiflinjbdggehcddcbncdddomop
Postman-Token: a28d3046-d905-b17c-3b68-64d97966d2ca
Content-Type: multipart/form-data; boundary=----WebKitFormBoundaryl1DbC6gBDszjzhwr
Accept: */*
Accept-Encoding: gzip, deflate
Accept-Language: ru-RU,ru;q=0.9,en-US;q=0.8,en;q=0.7

------WebKitFormBoundaryl1DbC6gBDszjzhwr
Content-Disposition: form-data; name="q"

{"request_type": "set","light_scene": {"3": 1, "4": 101, "5": 100}}
------WebKitFormBoundaryl1DbC6gBDszjzhwr--
```

Получение текущего состояния элементов системы

Шаблоны запроса на получение информации

Через POST запрос:

Получить диапазон элементов:

```
{
      "request_type": "get",
      "<тип запрашиваемого элемента>": {
          "range_begin": "<индекс первого элемента>",
          "range_end": "<индекс последнего элемента>"
      }
}
```

Получить список элементов:

```
{
      "request_type": "get",
       "<тип запрашиваемого элемента>": {
            "list": [<индекс1>, <индекс2>, <индекс3>]
      }
}
```

Для большинства типов запросы можно комбинировать:

```
{
      "request_type": "get",
       "<тип запрашиваемого элемента>": {
          "range_begin": "<индекс первого элемента>",
            "range_end": "<индекс последнего элемента>"
            "list": [<индекс1>, <индекс2>, <индекс3>]
      }
}
```

Через GET запрос:

```
GET /api?request_type=<get>&item_type=<тип запрашиваемого элемента>&range_begin=<индекс первого элемента>&range_end=<индекс последнего элемента>&list=<индекс1>-<индекс2>-<индекс3>
```

Например:

GET /api?request_type=get&item_type=light_state&range_begin=1&range_end=20&list=40-42-46

Для «item_type=bit» запрос будет выглядеть следующим образом:

GET /api?request_type=get&item_type=bit&list=40.0-42.7-46.5

До точки – номер байта, после – номер бита в байте.

Ответ на запрос выглядит следующим образом:

В случае неверного запроса:

```
{
      "response": "bad request"
}
```

Неверным запросом считается так же индекс элемента превышающий их количество в системе (различается для разных типов элементов).

В случае успеха:

```
{
     "response": "ok",
     "<тип возвращаемого элемента>": {
          "<индекс1>": <значение1>,
          "<индекс2>": <значение2>,
          ...
     }
}
```

Формат значений (<значение1>, <значение2>, …) **различается для разных типов** и описывается в подробном списке запросов.

Список возможных "get" запросов

- **Выключатели:**

Пример запроса:

```
{
      "request_type": "get",
      "switch": {

        "range_begin": 1,
        "range_end": 3,
        "list": [54, 83, 255]
      }
}
```

Пример ответа:

```
{
      "response": "ok",
      "switch": {"1": 0, "2": 0, "3": 0, "54": 0, "83": 0, "255": 1}
}
```

- **Датчики движения:**

Запрос и ответ информации о датчиках движения соответствует запросу для выключателей, необходимо заменить «switch» на «pir_sensor».

- **Состояние группы света:**

Пример запроса:

```
{

    "request_type": "get",
    "light_state": {
         "range_begin": 9,
         "range_end": 12,
         "list": [54, 83, 255]
    }
}
```

Пример ответа:

```
{
      "response": "ok",
      "light_state": {
            "9": 1, "10": 1, "11": 1, "12": 1, "54": 0, "83": 0, "255": 0
      }
}
```

- **Яркость группы света:**

Пример запроса:

```
{
      "request_type": "get",
      "light_brightness": {
            "range_begin": 9,
            "range_end": 12,
            "list": [54, 83, 255]
      }
}
```

Пример ответа:

```
{
	"response": "ok",
	"light_brightness": {
		"9": 50, "10": 70, "11": 30, "12": 90, "54": 0, "83": 0, "255": 0
	}
}
```

Яркость, полученная этим запросом, имеет дискретность в 10%.

- **Полное состояние группы света:**

Пример запроса:

```
{
	"request_type": "get",
	"light_full_state": {
		"range_begin": 9,
		"range_end": 12,
		"list": [54, 83, 255]
	}
}
```

Пример ответа:

```
{
	"response": "ok",
	"light_full_state": {
		"9": {"state": 1, "brightness": 50, "move_allow": 0},
		"10": {"state": 1, "brightness": 70, "move_allow": 0},
		"11": {"state": 1, "brightness": 30, "move_allow": 0},
		"12": {"state": 1, "brightness": 90, "move_allow": 0},
		"54": {"state": 1, "brightness": 0, "move_allow": 0},
		"83": {"state": 0, "brightness": 0, "move_allow": 0},
		"255": {"state": 0, "brightness": 0, "move_allow": 1}
	}
}
```

Параметр «move_allow» отвечает за разрешение лампочке срабатывать от привязанного к ней датчика движения. Из-за ограниченного размера буфера не рекомендуется запрашивать более 80 групп света одним запросом, в противном случае можно получить {"response": "bad request"}. Яркость, полученная этим запросом, имеет дискретность в 10%, состояние не зависит от яркости (лампочка может быть включена на 0%).

- **Прецизионная яркость группы света:**

Пример запроса:

```
{
	"request_type": "get",
	"light_brightness_precision": {
		"range_begin": 9,
		"range_end": 12,
		"list": [54, 83, 255]
	}
}
```

Пример ответа:

```
{
	"response": "ok",
	"light_brightness": {
		"9": 45, "10": 71, "11": 34, "12": 96, "54": 0, "83": 0, "255": 3
	}
}
```

Яркость, полученная этим запросом, имеет дискретность в 1%.

- **Полное состояние группы света с прецизионной яркостью:**

Пример запроса:

```
{
	"request_type": "get",
	"light_full_state_precision": {
		"range_begin": 9,
		"range_end": 12,
		"list": [54, 83, 255]
	}
}
```

Пример ответа:

```
{
	"response": "ok",
	"light_full_state": {
		"9": {"state": 1, "brightness": 45, "move_allow": 0},
		"10": {"state": 1, "brightness": 71, "move_allow": 0},
		"11": {"state": 1, "brightness": 34, "move_allow": 0},
		"12": {"state": 1, "brightness": 96, "move_allow": 0},
		"54": {"state": 0, "brightness": 0, "move_allow": 0},
		"83": {"state": 0, "brightness": 0, "move_allow": 0},
		"255": {"state": 1, "brightness": 3, "move_allow": 1}
	}
}
```

Параметр «move_allow» отвечает за разрешение лампочке срабатывать от привязанного к ней датчика движения. Из-за ограниченного размера буфера не рекомендуется запрашивать более 80 групп света одним запросом, в противном случае можно получить {"response": "bad request"}. Яркость, полученная этим запросом, имеет дискретность в 1%, состояние зависит от яркости (если яркость 0%, то state = 0).

- **Шторы:**

Запросы информации о шторах соответствуют запросам для света, необходимо лишь заменить в запросе «light» на «shade» и «brightness» на «position».

- **Состояние системы защиты от протечек:**

Пример запроса:

```
{
	"request_type": "get",
	"leaks": {
		"range_begin": 15,
		"range_end": 20,
		"list": [5, 1, 32]
	}
}
```

Пример ответа:

```
{
      "response": "ok",
      "leaks": {
            "15": 0, "16": 0, "17": 0, "18": 0, "19": 0,
            "20": 0, "5": 0, "1": 0, "32": 0,
            "valves": {"1: 0, "2": 0},
            "cleaning": {"state": 0, "time": 112}
      }
}
```

Под номерами идут состояния датчиков протечек (0 – нет протечки, 1 – есть), «valves» - состояния кранов стояков (всегда 2 шт. 0 – открыт, 1 - закрыт) , «cleaning» - состояние режима «уборка».

- **Состояние термостата воздуха по номеру помещения:**

Пример запроса:

```
{
      "request_type": "get",
      "air_termostat": {
            "range_begin": 1,
            "range_end": 3,
            "list": [5, 6]
      }
}
```

Пример ответа:

```
{
   "response": "ok",
        "air_termostat": {
        "1": {"current": 0.0, "setpoint": 10.0, "heating": 1, "state":
        "eco","room_mode": "manual"},
        "2": {"current": 0.0, "setpoint": 28.6, "heating": 1, "state":
        "vkl","room_mode": "auto"},
        "3": {"current": 0.0, "setpoint": 25.4, "heating": 1, "state":
        "vkl","room_mode": "auto"},
        "5": {"current": 0.0, "setpoint": 22.0, "heating": 1, "state":
        "vkl","room_mode": "manual"},
        "6": {"current": 0.0, "setpoint": 22.0, "heating": 1, "state":
        "vkl","room_mode": "manual"}
   }
}
```

Current – реальная температура, «setpoint» – желаемая, «heating»: 1 = нагрев, 0 = охлаждение, «state»: «vkl» – нормальный режим, «eco» – режим экономии, «room_mode» – режим работы климата в помещении.

- **Состояние термостата пола по номеру помещения:**

Для запроса состояния термостата пола **по номеру помещения** требуется в запросе для воздуха заменить «air» на «floor».

- **Состояние кондиционера по номеру помещения:**

Пример запроса:

```
{
      "request_type": "get",
      " ac_and_room_mode": {
            "range_begin": 1,
            "range_end": 2,
            "list": [3]
      }
}
```

Пример ответа:

```
{
    "response": "ok",
    "ac_and_room_mode": {
         "1": {"room_mode": "manual", "state": 0, "heating": 1, "cooling": 0,
         "speed": "high"},
         "2": {"room_mode": "auto", "state": 0, "heating": 0, "cooling": 1,
         "speed": "low"},
         "3": {"room_mode": "manual", "state": 0, "heating": 1, "cooling": 1,
         "speed": "high"}
    }
}
```

- **Показания датчиков качества воздуха (влажности):**

Пример запроса:

```
{
      "request_type": "get",
      "air_quality": {
            "range_begin": 1,
            "range_end": 2,
            "list": [20, 25, 43]
      }
}
```

Пример ответа:

```
{
      "response": "ok",
      "air_quality ": {
            "1": 34.4, "2": 42.6, "3": 44.8, "20": 45.0, "25": 0.0, "43": 0.0
      }
}
```

- **Состояние уведомлений:**

Пример запроса:

```
{
      "request_type": "get",
      "notifications": {
            "range_begin": 39,
            "range_end": 42,
            "list": [50, 48]
      }
}
```

Пример ответа:

```
{
    "response": "ok",
    "notifications": {
      "39": {"status": 1, "time": 1551529141, "type": "message", "parameter": 0},
      "40": {"status": 1, "time": 1551529142, "type": "message", "parameter": 5},
      "41": {"status": 0, "time": 0, "type": "message", "parameter": 1},
      "42": {"status": 1, "time": 1551529154, "type": "message", "parameter": 6},
      "50": {"status": 0, "time": 0, "type": "alarm", "parameter": 9},
      "48": {"status": 0, "time": 0, "type": "alarm", "parameter": 5},
      "52": {"status": 1, "time": 1551527451, "type": "message", "parameter": 0}
    }
}
```

«status»: 0 = не активно, 1 = активно, «time» – время появления уведомления в формате UTC, «type»: сообщение или авария, «parameter» – зависит от того что это за уведомление, например для сообщения о постановке на охрану датчиков движения - это их количество, а при тревоге на датчике – его номер. Из-за ограниченного размера буфера не рекомендуется запрашивать более 80 уведомлений одним запросом, в противном случае можно получить {"response": "bad request"}.

- **Запрос только активных уведомлений:**

Пример запроса:

```
{
      "request_type": "get",
      "active_notifications": {
            "range_begin": 39,
            "range_end": 42
      }
}
```

В этом запросе параметр «list» не используется.

Пример ответа:

```
{
    "response": "ok",
    "active_notifications": {
        "39": {"time": 1551529141, "type": "message", "parameter": 0},
        "40": {"time": 1551529142, "type": "message", "parameter": 3},
        "42": {"time": 1551529154, "type": "message", "parameter": 3}
    }
}
```

В ответ придет список активных аварий в заданном диапазоне.

- **Запрос активных аварий:**

Пример запроса:

```
{
      "request_type": "get",
      "active_alarms": {
            "range_begin": 1,
            "range_end": 100
      }
}
```

В этом запросе параметр «list» не используется.

Пример ответа:

```
{
    "response": "ok",
    "active_alarms": {
         "62": {"time": 1551532750, "parameter": 60},
         "64": {"time": 1551532751, "parameter": 60},
         "65": {"time": 1551532750, "parameter": 0},
         "66": {"time": 1551532751, "parameter": 0},
         "86": {"time": 1551532751, "parameter": 255}
    }
}
```

- **Запрос активных сообщений:**

Пример запроса:

```
{
      "request_type": "get",
      "active_messages": {
            "range_begin": 1,
            "range_end": 100
      }
}
```

В этом запросе параметр «list» не используется.

Пример ответа:

```
{
    "response": "ok",
    "active_messages": {
         "39": {"time": 1551532917, "parameter": 0},
         "40": {"time": 1551532919, "parameter": 5},
         "42": {"time": 1551532929, "parameter": 30},
         "52": {"time": 1551531677, "parameter": 0},
         "61": {"time": 1551532928, "parameter": 56},
         "63": {"time": 1551532929, "parameter": 56},
         "67": {"time": 1551532929, "parameter": 0}
    }
}
```

- **Состояния вытяжек:**

Пример запроса:

```
{
      "request_type": "get",
      "extractor_fun": {
            "range_begin": 1,
            "range_end": 3,
            "list": [5, 6]
      }
}
```

Пример ответа:

```
{
    "response": "ok",
    "extractor_fun": {"1": 0, "2": 0, "3": 1, "5": 1, "6": 0}
}
```

- **Состояния нагрузок на фазе R:**

  Пример запроса:

```
{
      "request_type": "get",
      "loads_R": {
            "range_begin": 1,
            "range_end": 6,
            "list": [20, 32]
      }
}
```

  Пример ответа:

```
{
    "response": "ok",
    "loads_R": {
         "1": {"state": 1, "over": 0},
         "2": {"state": 1, "over": 0},
         "3": {"state": 1, "over": 0},
         "4": {"state": 1, "over": 0},
         "5": {"state": 1, "over": 0},
         "6": {"state": 1, "over": 0},
         "20": {"state": 1, "over": 0},
         "32": {"state": 1, "over": 1}
    }
}
```

- **Состояния нагрузок на других фазах:**

  Состояния нагрузок на других фазах запрашиваются так же с заменой «loads_R» на «loads_S», «loads_T» и «loads_RST».

- **Получение значения из байта по адресу:**

  Пример запроса:

```
{
      "request_type": "get",
      "byte": {
            "range_begin": 620,
            "range_end": 627,
            "list": [628, 629]
      }
}
```

  Пример ответа:

```
{
    "response": "ok",
    "byte": {
         "620": 3, "621": 1, "622": 0, "623": 0,
         "624": 0, "625": 24, "626": 0, "627": 0, "628": 0, "629": 2
    }
}
```

- **Получение значения двух байт по адресу:**

  Получение значения двух байт по адресу идентично получению значения байта, нужно в запросе заменить «byte» на «word».

- **Получение значения бита из байта по адресу:**

  Пример запроса:

```
{
      "request_type": "get",
      "bit": {
            "list": {"628": 0, "629": 7}
      }
}
```

  В этом запросе параметр «range» не используется и «list» должен быть объектом. «628» – адрес байта, «0» – номер бита.

  Пример ответа:

```
{
    "response": "ok",
         "bit": {
              "628": {"0": 1}, 629": {"7": 0}
    }
}
```

Изменение текущего состояния элементов системы

Шаблон запроса на изменение

```
{
      "request_type": "set",
      "<тип изменяемого элемента>": {
          "<индекс первого элемента>": "<значение1>",
          "<индекс второго элемента>": "<значение2>"
     }
}
```

Формат значений (`<значение1>, <значение2>, …`) **различается для разных типов** и описывается в подробном списке запросов.

Ответ на запрос выглядит следующим образом:

В случае неверного запроса:

```
{
      "response": "bad request"
}
```

Неверным запросом считается так же индекс элемента превышающий их количество в системе (различается для разных типов элементов).

В случае успеха:

```
{
     "response": "ok",
     "<тип измененного элемента>"
}
```

- **Состояние группы света:**

  Пример запроса:

```
{
    "request_type": "set",
    "light_state": {"1": 0, "2":1}
}
```

- **Яркость группы света:**

  Пример запроса:

```
{
    "request_type": "set",
    "light_brightness": {"1": 10, "2":30}
}
```

  Яркость, установленная этим запросом, должна иметь дискретность в 10%.

- **Состояние группы света и её яркость в одном запросе:**

  Пример запроса:

```
{
    "request_type": "set",
    "light_state_and_brightness": {
         "1": {"state": 1},
         "2": {"brightness": 90},
         "3": {"state": 1, "brightness": 40}
    }
}
```

  Яркость, установленная этим запросом, должна иметь дискретность в 10%, состояние не зависит от яркости (лампочка может быть включена на 0%).

- **Прецизионная яркость группы света:**

  Пример запроса:

```
{
    "request_type": "set",
    "light_brightness_precision": {"1": 11, "2":34}
}
```

  Яркость, установленная этим запросом, должна иметь дискретность в 1%, состояние и яркость зависят друг от друга (при установке яркости в 0%, лампочка будет выключена, при выключении – яркость будет сброшена на 0%). Прецизионную яркость, дополнительно к стандартной (с дискретность в 10%), имеют только первые 140 ламп! Для остальных будет использована стандартная.

- **Разрешение группе света работать по датчику движения:**

  Пример запроса:

```
{
    "request_type": "set",
    "light_move_allow": {"1": 0, "2": 1}
}
```

- **Шторы:**

  Для изменения состояния штор нужно в запросах для света изменить «light» на «shade» и «brightness» на «position»

- **Управление кранами стояков:** (возможность зависит от настроек системы протечек)
  Пример запроса:
  ```
  {
      "request_type": "set",
      "valve": {"1": 0, "2": 1}
  }
  ```

- **Управление режимом «уборка»:**
  ```
  {
      "request_type": "set",
      "cleaning": {
           "1": {"state": 1, "time": 20}
      }
  }
  ```
  Параметры «state» и «time» можно указывать по-отдельности.

- **Установка температуры для термостата воздуха:**
  Пример запроса:
  ```
  {
      "request_type": "set",
      "air_termostat_temp": {"1": 15.6, "2": 29.4}
  }
  ```
  Если термостат находился в режиме «eco», то изменение уставки приведет к включению режима «vkl» и возврату к старой уставке для этого режима.

- **Изменение режима «vkl» / «eco» для термостата воздуха:**
  Пример запроса:
  ```
  {
      "request_type": "set",
      "air_termostat_state": {"1": 0, "2": 1}
  }
  ```

- **Управление турмостатом пола:**
  Для управления термостатом пола требуется в запросах для воздуха изменить «air» на «floor»

- **Управление кондиционером:**
  Пример запроса:
  ```
  {
      "request_type": "set",
      "ac_state": {
           "1": {"state": 1, "heating": 1, "cooling": 0, "speed": "low"},
           "2": {"state": 1, "heating": 0, "cooling": 1, "speed": "high"}
      }
  }
  ```
  «state» - вкл/выкл кондиционер, «heating/cooling» - на нагрев/ожлаждение, «speed» (high/low) – скорость потока воздуха.

- **Изменение режима «auto» / «manual» в комнате:**
  Пример запроса:
  ```
  {
      "request_type": "set",
      "room_mode": {"1": "auto"}

  }
  ```

- **Управление вытяжками:**
  Пример запроса:
  ```
  {
      "request_type": "set",
      "extractor_fun": {"1": 1, "2": 1, "5": 1, "8": 1}
  }
  ```

- **Управление нагрузками:**
  Пример запроса:
  ```
  {
      "request_type": "set",
      "loads_R": {"1": 1, "2": 1, "5": 0, "8": 0}
  }
  ```
  Для управления нагрузками разных фаз необходимо заменять «loads_R» на «loads_S», «loads_T», «loads_RST».

- **Запуск сцен:**
  Пример запроса:
  ```
  {
      "request_type": "set",
      "scenes": {"light": 1, "climate": 100, "ingeneer": 3, "multiscene": 1}
  }
  ```
  В запросе можно комбинировать команды на запуск глобальных сцен для различных систем. Для света и климата существуют 4 сцены под номерами 1 – 4 и специальные сцены «включить всё» и «выключить всё» под номерами 100 и 101.

- **Запуск минисцен освещения:**
  Пример запроса:
  ```
  {
      "request_type": "set",
      "light_scene": {"3": 1, "4": 101, "5": 100}
  }
  ```

Минисцены освещения запускаются для комнат «"номер комнаты": "номер сцены"». Номер комнаты можно узнать по номеру группы света, находящейся в этом помещении. Он указан в сводной таблице освещения в инженерном интерфейсе EasyHome в столбце «ID Пом.».

Максимум 50 минисцен за 1 запрос. 1 минисцена выполняется раз в 100мс, поэтому следующий запрос запуска минисцен следует производить через 5 секунд.

- **Управление замками:**
  Пример запроса:
  ```
  {
      "request_type": "set",
      "latch": {"1": 1, "2": 1, "5": 1, "8": 1}
  }
  ```

- **Изменение значения байта по адресу:**
  Пример запроса:
  ```
  {
      "request_type": "set",
      "byte": {"1940": 42, "1941": 74}
  }
  ```

- **Изменение значения двух байт по адресу:**

  Пример запроса:

```
{
    "request_type": "set",
    "word": {"1940": 26954}
}
```

- **Изменение значения бита в байте по адресу:**

  Пример запроса:

```
{
    "request_type": "set",
    "bit": {"620": {"0": 1}, "620": {"1": 1}}
}
```

Подписка на изменения элементов

В системе предусмотрена возможность получать сообщения об изменении состояния различных элементов. Для этого необходимо подписаться на изменения, сообщив контроллеру IP адрес и порт на который он должен отправлять сообщения.

Запрос подписки

```
{
     "request_type": "set",
     "subscribe": {
          "host": "192.168.1.62",
          "port": 15284,
          "uri": "easyhome/events.php?from=klen&key=abcdefgh"
     }
}
```

Параметры «host», «port» и «uri» характеризуют устройство на которое сервер должен присылать изменения. Максимальная длина «uri» - 99 символов.

Вид http заголовка с сообщением по указанной подписке:

```
POST /easyhome/events.php?from=klen&key=abcdefgh HTTP/1.1
Host: 192.168.1.62:15284
Connection: Close
Content-Type: application/json
CONTENT-LENGTH: 39

{"changes": {"light_state": {"23": 1}}}
```

Типы элементов, изменения в которых контроллер должен присылать настраиваются через инженерный интерфейс в разделе «расширения EasyHome».

Если в настройках установлен параметр «забывать подписки через 20 минут», то через 20 минут подписка будет удалена. Чтобы этого не происходило подписку необходимо обновлять не реже 1 раза в 20 минут.

Поддерживается одновременная подписка до трех устройств. Каждое следующее будет заменять собой наиболее старую подписку.

Общий вид сообщения

```
{
     "changes": {
          "<тип измененного элемента>":{
               "<индекс1>": <значение1>,
               "<индекс2>": <значение2>,
               ...
          }
     }
}
```

Формат значений (`<значение1>, <значение2>, …`) **различается для разных типов** и описывается в подробном списке сообщений.

Список возможных сообщений

- **Выключатели:**

```
{
    "changes": {
        "switch": {"1": 1}
    }
}
```

- **Датчики движения:**

```
{
    "changes": {
        "pir_sensor": {"1": 1}
    }
}
```

- **Состояние групп света:**

```
{
    "changes": {
        "light_state": {"21": 1}
    }
}
```

- **Яркость групп света:**

```
{
    "changes": {
        "light_brightness": {"3": 90}
    }
}
```

- **Разрешение работы группы света по датчику движения:**

```
{
    "changes": {
        "light_move_allow": {"3": 1}
    }
}
```

- **Нагрузки фазы R:**

```
{
    "changes": {
        "loads_R": {
            "7": {"state": 0, "over": 1}
        }
    }
}
```

- **Нагрузки на других фазах:**

  Нагрузки на других фазах соответственно помечаются как «loads_S», «loads_T», «loads_RST»

- **Вытяжки:**

```
{
    "changes": {
        "extractor_fun": {"4": 1}
    }
}
```

- **Датчики протечки:**

```
{
    "changes": {
        "leaks": {"5": 1}
    }
}
```

- **Краны стояков:**

```
{
    "changes": {
        "valves": {"1": 1}
    }
}
```

- **Состояние режима уборки:**

```
{
    "changes": {
        "cleaning": {"state": 1, "time": 111}
    }
}
```

- **Аварии:**

```
{
    "changes": {
        "alarms": {
            "68": {"state": 0, "time": 1552838803, "parameter": 5}
        }
    }
}
```

- **Сообщения:**

```
{
    "changes": {
        "messages": {
            "40": {"state": 1, "time": 1552837405, "parameter": 5},
            "42": {"state": 1, "time": 1552837410, "parameter": 36},
            "61": {"state": 1, "time": 1552837402, "parameter": 5},
            "63": {"state": 1, "time": 1552837402, "parameter": 5}
        }
    }
}
```