ПО EDGE. Public HTTP API

Версия v1 (edge-base >= 0.29.x)

Введение

В настоящем документе приводится программный интерфейс (API) для доступа к информационным ресурсам программно-аппаратного комплекса EDGE. Назначение данного API состоит в том, чтобы предоставить сторонним системам возможность получать результаты работы комплекса:

- последовательно, в пакетном режиме (например, для репликации в центральную базу данных);
- *или*
- порциями, начиная с произвольного момента времени (для оперативного отображения данных в пользовательском интерфейсе).

Термины и определения

Комплекс — программно-аппаратный комплекс EDGE.

Сервер — в настоящем документе означает подсистему, реализующую описываемый программный интерфейс комплекса.

Клиент — в настоящем документе означает подсистему, использующую описываемый программный интерфейс.

Архив событий — долговременное хранилище событий от множества комплексов.

Пост архива событий — долговременное хранилище событий от одного комплекса.

Канал поста архива событий — долговременное хранилище событий от одного комплекса, полученных в одной зоне контроля.

Событие — запись о происшествии в зоне контроля, которая относится к определенному каналу поста архива событий.

Событие Fact — факт фиксации проезда транспортного средства через зону контроля.

Доступ к функциям API осуществляется с помощью протокола HTTP. В качестве ответа на запросы клиента могут возвращаться:

1. данные о структурных компонентах EDGE в формате JSON (в кодировке UTF-8);
2. метаданные о событиях в формате JSON (в кодировке UTF-8);
3. медиаданные в формате JPEG.

Конкретные адрес и порт для доступа к устройству зависят от настроек EDGE и сети, в которой происходит развертывание.

В запросах все компоненты чувствительны к регистру; порядок параметров (после символа “?”) значения не имеет.

Все запросы должны содержать параметры аутентификации пользователя методом HTTP Basic Authentication.

Обработка запроса *не* выполняется в случае:

1. отсутствия параметров аутентификации в запросе: сервер возвращает ответ с HTTP-кодом 404 (NOT FOUND);
2. несовпадения параметров аутентификации пользователя: сервер возвращает ответ с HTTP-кодом 401 (UNAUTHORIZED);
3. наличия некорректных параметров в запросе: сервер возвращает ответ с HTTP-кодом 404 (NOT FOUND);
4. внутренней ошибки сервера при обработке запроса: сервер возвращает ответ с HTTP-кодом 500 (INTERNAL); рекомендуется сообщить о такой ошибке разработчику.

Ответы сервера могут содержать дополнительные поля, не описанные в настоящем документе. При обработке ответов сервера клиент должен игнорировать такие недокументированные поля.

Базовые возможности

Проверка доступности сервера (get-ping)

Структура URL-запроса **get-ping** определяется форматом:

```
GET /api/v1/ping
```

Сервер, после успешной обработки запроса, возвращает ответ с HTTP-кодом 200. Тело ответа содержит application/JSON-документ с параметрами:

1. **"status" :: uint16_t.** Содержит 200.

Пример тела ответа:

```
{
 "status": 200
}
```

Завершение сеанса пользователя (get-auth)

Структура URL-запроса **get-auth** определяется форматом:

```
GET /auth
```

Сервер, после успешной обработки запроса, возвращает ответ с HTTP-кодом 401.

Запрос поддерживаемых типов тревог (get_alarm-set)

Структура URL-запроса **get_alarm-set** определяется форматом:

```
GET /api/v1/alarmtypes
```

Сервер, после успешной обработки запроса **get_alarm-set**, возвращает ответ с HTTP-кодом 200. Тело ответа содержит application/JSON-документ **get_alarm-set_rbody-200** с параметрами:

1. **"status" :: uint16_t.** Содержит 200.
2. **"result" :: char[32][].** Массив. Каждый элемент содержит поддерживаемый тип тревоги. Элементы массива не упорядочены. Доступные типы тревог:
    1. **"speedviolation".** Нарушение скоростного режима.

Пример тела ответа:

```
{
  "status": 200,
  "result":
  [
    "speedviolation"
  ]
}
```

Запрос доступных постов, каналов поста и их параметров

Запрос доступных постов

get_post-set

Структура URL-запроса **get_post-set** определяется форматом:

```
GET /api/v1/post
```

Если структура URL-запроса get-post-set соответствует требованиям, сервер, после успешной обработки запроса, возвращает ответ с HTTP-кодом 200. Тело ответа содержит application/JSON-документ **get_post-set_rbody-200** с параметрами:

1. **"status" :: uint16_t.** Содержит 200.

2. **"result" :: char[][37].** Массив. Каждый элемент содержит глобальный идентификатор *поста* (см. **Термины и определения**).

Пример **get_post-set_rbody-200**:

```
{
 "status": 200,
 "result":
 [
   "a51d1532-466f-490a-8b06-753cfef7de28"
 ]
}
```

get_post-set2

Структура URL-запроса **get_post-set2** определяется форматом:

```
GET /api/v1/post?full
```

Сервер, после успешной обработки запроса **get_post-set2**, возвращает ответ с HTTP-кодом 200. Тело ответа содержит application/JSON-документ **get_post-set2_rbody-200** с параметрами:

1. **"status" :: uint16_t.** Содержит 200.
2. **"result" :: struct[]**. Массив. Каждый элемент содержит параметры поста архива событий.
    1. **"Name" :: char[256]**. Имя.
    2. **"Guid" :: char[37]**. Глобальный идентификатор.

Пример **get_post-set2_rbody-200**:

```
{
 "status": 200,
 "result":
 [
   {
     "Name": "UA-03",
     "Guid": "a51d1532-466f-490a-8b06-753cfef7de28",
   }
 ]
}
```

Запрос параметров поста (get_post-args)

Структура URL-запроса **get_post-args** определяется форматом:

```
GET /api/v1/post/<post-guid>
```

где **<post-guid> :: char[37]** — глобальный идентификатор поста; возможны значения согласно **get_post-set_rbody-200** (см. **Запрос доступных постов** выше).

Пример URL-запроса:

```
GET /api/v1/post/a51d1532-466f-490a-8b06-753cfef7de28
```

Сервер, после успешной обработки запроса **get_post-args**, возвращает ответ с HTTP-кодом 200. Тело ответа содержит application/JSON-документ **get_post-args_rbody-200** с параметрами:

1. **"status" :: uint16_t.** Содержит 200.
2. **"result" :: struct.** Параметры поста архива событий.
    1. **"Name" :: char[256].** Имя поста.
    2. **"Guid" :: char[37].** Глобальный идентификатор поста.

Пример **get_post-args_rbody-200**:

```
{
  "status": 200,
  "result":
  {
    "Name": "UA-03",
    "Guid": "a51d1532-466f-490a-8b06-753cfef7de28"
  }
}
```

Запрос доступных каналов поста

get_post-channel-set

Структура URL-запроса **get_post-channel-set** определяется форматом:

```
GET /api/v1/post/<post-guid>/channel
```

где **<post-guid>** — глобальный идентификатор поста.

Если структура URL-запроса **get-post-channel-set** соответствует требованиям, сервер, после успешной обработки, возвращает ответ с HTTP-кодом 200. Тело ответа содержит application/JSON-документ **get_post-channel-set_rbody-200** с параметрами:

1. **"status" :: uint16_t.** Содержит 200.
2. **"result" :: int32_t[].** Массив. Каждый элемент содержит номер *канала поста архива событий* (см. **Термины и определения**). Элементы массива не упорядочены.

Пример **get_post-channel-set_rbody-200**:

```
{
 "status": 200,
 "result": [0]
}
```

get_post-channel-set2

Структура URL-запроса **get_post-channel-set2** определяется форматом:

```
GET /api/v1/post/<post-guid>/channel?full
```

где **<post-guid>** — глобальный идентификатор поста.

Если структура URL-запроса **get_post-channel-set2** соответствует требованиям, сервер, после успешной обработки, возвращает ответ с HTTP-кодом 200. Тело ответа содержит application/JSON-документ **get_post-channel-set2_rbody-200** с параметрами:

1. **"status" :: uint16_t.** Содержит 200.
2. **"result" :: struct[].** Массив. Каждый элемент содержит параметры канала поста.
    1. **"PostGuid" :: char[37].** Глобальный идентификатор поста.
    2. **"Channel" :: int32_t.** Номер канала поста.

Пример **get_post-channel-set2_rbody-200**:

```
{
  "status": 200,
  "result":
  [
    {
      "PostGuid": "a51d1532-466f-490a-8b06-753cfef7de28",
      "Channel": 0
    }
  ]
}
```

Запрос параметров канала поста (get_post-channel)

Структура URL-запроса **get_post-channel** определяется форматом:

```
GET /api/v1/post/<post-guid>/channel/<channel-no>
```

где

- **<post-guid> :: char[37]** — глобальный идентификатор поста; возможны значения согласно **get_post-set_rbody-200** (см. **Запрос доступных постов** выше);
- **<channel-no> :: int32_t** — номер канала поста; возможны значения согласно **get_post-channel-set_rbody-200** (см. **Запрос доступных каналов поста** выше).

Если структура URL-запроса **put-post-channel** соответствует требованиям, сервер после успешной обработки возвращает ответ с HTTP-кодом 200. Тело ответа содержит application/JSON-документ **get_post-channel_rbody-200** с параметрами:

1. **"status" :: uint16_t.** Содержит 200.
2. **"result" :: struct.** Параметры канала поста:
    1. **"PostGuid" :: char[37].** Глобальный идентификатор поста.
    2. **"Channel" :: int32_t.** Номер канала поста.

Пример тела ответа:

```
{
  "status": 200,
  "result":
  {
    "PostGuid": "a51d1532-466f-490a-8b06-753cfef7de28",
    "Channel": 0
  }
}
```

Запрос параметров событий Fact, связанных изображений и метаданных

Запрос параметров события Fact (get_fact-args)

Структура URL-запроса **get_fact-args** определяется форматом:

```
GET /api/v1/fact/<fact-guid>
```

где **<fact-guid> :: char[37]** — глобальный идентификатор события.

Сервер, после успешной обработки запроса **get_fact-args**, возвращает ответ с HTTP-кодом 200. Тело ответа содержит application/JSON-документ **get_fact-args_rbody-200** с параметрами:

1. **"status" :: uint16_t.** Содержит 200.
2. **"result" :: struct.** Параметры события Fact:
    1. **"Id" :: int64_t.** Локальный идентификатор события (уникален в пределах архива событий). Значение отражает хронологический порядок фиксации изображений и записи события в архив.
    2. **"Guid" :: char[37].** Глобальный идентификатор события.
    3. **"Bestts" :: uint64_t.** Время фиксации изображений события, мкс (от 01.01.1970 00:00 UTC).
    4. **"Image" :: uint64_t.** Идентификатор изображений событий. Содержит **media-id** (см. далее).
    5. **"Lane" :: uint8_t.** Полоса движения транспортного средства. Возможные значения в интервале [0; 8]. Значение 0 указывает, что полоса не определена.
    6. **"Plate" :: struct.** Параметры номерной пластины транспортного средства:
        1. **"Text" :: char[].** Текст номера.
        2. **"Country" :: uint16_t.** Код страны регистрации номера. Типичные значения согласно ISO 1366.
        3. **"Validity" :: int8_t.** Мера достоверности фиксации номера. Большим значениям соответствует большая достоверность.
    7. **"Speed" :: uint8_t | null.** Скорость движения транспортного средства, км/ч. Значение null (по умолчанию) означает отсутствие фиксации.
    8. **"SpeedLimit" :: uint8_t | null.** Установленное ограничение скоростного режима, км/ч. Значение null (по умолчанию) означает отсутствие ограничения.
    9. **"Direction" :: char[16].** Направление движения транспортного средства в зоне контроля:
        1. **"unknown".** Неизвестно
        2. **"approaching".** Приближается
        3. **"receding".** Удаляется
    10. **"AlarmTypes" :: char[][32].** Массив. Каждый элемент содержит тип тревоги, зафиксированный при проезде транспортным средством зоны контроля. Возможные значения типов тревог определяются запросом AlarmSet. По умолчанию пустой массив.
    11. **"Geodata" :: struct.** Параметры геолокации устройства в момент фиксации события:
        1. **"latitude" :: double.** Координата широты, градусов. Направление координаты определяется знаком значения:
            1. **Плюс.** На север от экватора
            2. **Минус.** На юг от экватора
        2. **"longitude" :: double.** Координата долготы, градусов. Направление координаты определяется знаком значения:
            1. **Плюс.** На восток от Гринвича
            2. **Минус.** На запад от Гринвича
        3. **"altitude" :: double | null.** Высота над уровнем моря, метров.
        4. **"speed" :: double | null.** Опциональный параметр. Собственная скорость устройства в момент фиксации события *(для мобильных комплексов)*, м/с.

Пример **get_fact-args_rbody-200**:


```
{
  "status": 200,
  "result":
  {
    "Id": 1437984917029087,
    "Guid": "3cb36c1b-5ccc-4d9c-9004-424e82530fbb",
    "Bestts": 1437984917029087,
    "Image": "1437984917029087",
    "Lane": 0,
    "Plate":
    {
      "Text": "A A0514 T P",
      "Country": 804,
      "Validity": 76,
      "Exact_rect": [9.87973e-3, 0.583429, 4.2955299999999e-2, 0.590857],
      "Type": 1
    },
    "Speed": 67,
    "SpeedLimit": 60,
    "Direction": "unknown",
    "AlarmTypes": ["speedviolation"],
    "Geodata":
    {
      "latitude": 50.43756,
      "longitude": 30.45221,
      "altitude": null,
      "speed": null
    }
  }
}
```


Запрос изображений, связанных с событием Fact

get_fact-image

Структура URL-запроса **get-fact-image** определяется форматом:

```
GET /api/v1/post/<post-guid>/channel/<channel-no>/\

media/<media-id>/<image-class>
```

где

- **<post-guid> :: char[37]** — глобальный идентификатор поста; возможны значения согласно **get_post-set_rbody-200** (см. **Запрос доступных постов, каналов поста и их параметров**);
- **<channel-no> :: int32_t** — номер канала поста; возможны значения согласно документа **get_post-channel-set_rbody-200** (см. **Запрос доступных постов, каналов поста и их параметров**);
- **<media-id> :: int64_t** — идентификатор изображения;
- **<image-class> :: char[]** — тип изображения.

Возможные значения параметра **<image-class>**:

1. **"full".** Изображение зоны контроля. Содержит полный кадр (полученный от камеры), на котором зафиксированы лучшие результаты фотовидеофиксации за время проезда транспортным средством зоны контроля.
2. **"body".** Изображение транспортного средства. Содержит фрагмент full-изображения минимального размера, который позволяет определить модель и цвет автомобиля.
3. **"plate".** Изображение номера транспортного средства. Содержит фрагмент full-изображения, который позволяет определить номер автомобиля.
4. **"fulltitled".** Титрованное full-изображение.
5. **"bodytitled".** Титрованное body-изображение.

get_fact-image2

Структура URL-запроса **get-fact-image2** определяется форматом:

```
GET /api/v1/fact/<fact-guid>/media/<image-class>
```

где **<fact-guid> :: char[37]** — глобальный идентификатор события фотовидеофиксации.

Сервер, после успешной обработки запросов **get_fact-image**, **get_fact-image2**, возвращает ответ с HTTP-кодом 200. Тело ответа содержит image/JPEG-изображение. Если изображение недоступно, сервер возвращает ответ с HTTP-кодом 404.

Запрос метаданных изображений, связанных с событием Fact

get_fact-image-meta

Структура URL-запроса **get_fact-image-meta** определяется форматом:

```
GET /api/v1/post/<post-guid>/channel/<channel-no>/\

media/<media-id>/<image-class>/meta
```

где

- **<post-guid> :: char[37]** — глобальный идентификатор поста; возможные значения согласно **get_post-set_rbody-200** (см. Запрос доступных постов, каналов поста и их параметров);
- **<channel-no> :: int32_t** — номер канала поста; возможны значения согласно **get_post-channel-set_rbody-200** (см. Запрос доступных постов, каналов поста, и их параметров);
- **<media-id> :: int64_t** — идентификатор изображения;
- **<image-class> :: char[]** — тип изображения.

Возможные значения параметра **<image-class>**:

1. **"full".** Изображение зоны контроля. Содержит полный кадр (полученный от камеры), на котором зафиксированы лучшие результаты фотовидеофиксации за время проезда транспортным средством зоны контроля.
2. **"body".** Изображение транспортного средства. Содержит фрагмент full-изображения минимального размера, который позволяет определить модель и цвет автомобиля.
3. **"plate".** Изображение номера транспортного средства. Содержит фрагмент full-изображения, который позволяет определить номер автомобиля.
4. **"fulltitled".** Титрованное full-изображение.
5. **"bodytitled".** Титрованное body-изображение.

get_fact-image-meta2

Структура URL-запроса **get_fact-image-meta2** определяется форматом:

```
GET /api/v1/fact/<fact-guid>/media/<image-class>/meta
```

где **<fact-guid> :: char[37]** — глобальный идентификатор события фотовидеофиксации.

Если структура URL-запроса **get_fact-image-meta** и **get_fact-image-meta2** соответствует требованиям, сервер, после успешной обработки, возвращает ответ с HTTP-кодом 200. Тело ответа содержит application/JSON-документ **get_fact-image-meta_rbody-200** с параметрами:

1. **"status" :: uint16_t.** Содержит 200.
2. **"result" :: struct.** Метаданные изображения события Fact.
    1. **"transform" :: struct.** Матрица трансформации. Характеризует соответствие точек на **full-** и **<image-class>-**изображении. Содержит коэффициенты a, b, c, d, x, y. В случае использования full-изображения в качестве

<image-class>-изображения параметр transform содержит матрицу единичного оператора (a=d=1, b=c=x=y=0).

2. **"size" :: (uint16_t, uint16_t).** Размер изображения (ширина и высота), пикселей.

Пример **get_fact-image-meta_rbody-200**:


```
{
 "status": 200,
 "result":
 {
   "transform":
   {
     "a": 22.17142857142857,
     "b": 0.0,
     "c": 0.0,
     "d": 83.33333333333333,
     "x": -7.731958762886598e-3,
     "y": -0.7291428571428571
   },
   "size":[134, 30]
 }
}
```


Для определения положения точки full-изображения (с координатами x’, y’) на <image-class>-изображении (с координатами x’’, y’’) необходимо в приведенную ниже систему подставить целевые значения x’, y’ и коэффициенты матрицы трансформации <image-class>-изображения.

```
|x"|   | a b x |   |x'|
|y"| = | c d y | * |y'|
|1 |   | 0 0 1 |   |1 |
```

где

- **x’’, y’’ —** относительные координаты точки на <image-class>-изображении; на видимой части изображения координаты принимают значения в интервале [0.0; 1.0];
- **a, b, c, d, x, y —** коэффициенты матрицы трансформации <image-class>-изображения;
- **x’, y’ —** относительные координаты точки на full-изображении; на видимой части изображения координаты принимают значения в интервале [0.0; 1.0].

Запросы для постраничного доступа к событиям Fact

get_fact-set

Структура URL-запроса **get_fact-set** (для загрузки первой страницы) определяется форматом:

```
GET /api/v1/post/<post-uuid>/channel/<channel-no>/fact\

    ?order=<order>

    &limit=<limit>
```

где

- **<post-uuid> :: char[37]** — глобальный идентификатор поста;
- **<channel-no> :: int32_t** — номер канала поста;
- **<limit> :: uint16_t** — максимальное количество событий в ответе (возможные значения в интервале [1;1000], по умолчанию — 10);
- **<order> :: char[]** — порядок загрузки событий относительно курсора чтения.

Возможные значения параметра **<order>**:

1. **"asc".** В прямом хронологическом порядке фиксации изображений и записи события в архив. По умолчанию начальное положение курсора чтения определяется событием с наименьшей отметкой времени. В теле ответа события упорядочены по возрастанию #/Id.
2. **"desc"** (по умолчанию). В обратном хронологическом порядке фиксации изображений события. По умолчанию начальное положение курсора чтения определяется событием с наибольшей отметкой времени. В теле ответа события упорядочены по убыванию #/Id.
3. **"db".** В прямом хронологическом порядке записи событий Fact в архив.
4. **"dbrev".** В обратном хронологическом порядке записи событий Fact в архив.

Клиентам рекомендуется задавать значение URL-параметра **limit**. У сервера может быть свое значение для этого ограничения, меньшее, чем запросил клиент. В этом случае сервер вернет ошибку. Соответственно, если клиент запросил лимит и сервер успешно вернул число записей, меньшее, чем было запрошено, — это значит, что были получены все имеющиеся записи, удовлетворяющие критерию поиска.

get_fact-set2

Структура URL-запроса **get_fact-set2** (для загрузки последующих страниц) определяется форматом:

```
GET /api/v1/post/<post-guid>/channel/<channel-no>/fact\

    ?startId=<start-fact-id>\

    &order=<order>\

    &limit=<limit>
```

где

- **<start-fact-id> :: int64_t** — начальное положение курсора чтения. Задается локальным идентификатором события Fact#Id. Тело ответа содержит событие, на которое указывает курсор, если это событие присутствует в архиве.

get_fact-set3

Структура URL-запроса **get_fact-set3** определяется форматом:

```
GET /api/v1/fact/post/<post-guid>/channel/<channel-no>\

    ?startGuid=<start-fact-guid>

    &order=<order>

    &limit=<limit>
```

где

- **<start-fact-guid> :: char[37]** — начальное положения курсора чтения. Задается глобальным идентификатором события Fact#Guid. Тело ответа содержит событие, на которое указывает курсор, если это событие присутствует в архиве.

Если структура url-запроса **get-fact-set**, **get-fact-set2** или **get-fact-set3** соответствует требованиям, сервер после успешной обработки возвращает ответ с http-кодом 200. Тело ответа содержит application/JSON-документ **get_fact-set_rbody-200** с параметрами:

1. **"status" :: uint16_t.** Содержит 200.
2. **"result" :: struct[].** Массив. Каждый элемент содержит параметры одного события Fact.

Запрос отчета по событиям Fact

Предназначен для определения статистики проездов зоны контроля транспортными средствами. Максимальный размер отчетного периода — 7 суток. Структура URL-запроса определяется форматом:

```
GET /api/v1/admin/stats/eventstore/fact\

    ?max-ts=<max-ts>

    &min-ts=<min-ts>\

    &min-ts-gap=<min-ts-gap>\

    &min-total-count=<min-total-count>
```

где

- **<max-ts>** — отметка времени завершения отчетного периода, мкс от 01.01.1970 00:00 UTC. Опциональный параметр. Возможны значения в интервале [0; 2147476447000000]. По умолчанию определяется системным временем на момент обработки запроса.
- **<min-ts>** — отметка времени начала отчетного периода, мкс (от 01.01.1970 00:00 UTC). Опциональный параметр. Возможны значения в интервале [0; 2147476447000000]. По умолчанию определяется как разность времени завершения отчетного периода и макс. размера отчетного периода в мкс.
- **<min-ts-gap>** — мин. промежуток времени между последовательными событиями, с. Опциональный параметр. Возможны значения в интервале [0; 65535]. По умолчанию 10.
- **<min-total-count>** — мин. количество проездов зоны контроля. Опциональный параметр. Возможны значения в интервале [1; 65535]. По умолчанию 2.

Если структура URL-запроса соответствует требованиям, сервер, после успешной обработки, возвращает ответ с HTTP-кодом 200. Тело ответа содержит application/JSON-документ с параметрами:

1. **"status" :: uint16_t.** Содержит 200.
2. **"result" :: struct[].** Массив. Каждый элемент содержит статистику проездов транспортным средством зоны контроля. Элементы в порядке убывания значения **total-count**:
    1. **"plate-text" :: char[].** Текст номера транспортного средства.
    2. **"total-count" :: uint32_t.** Количество проездов зоны контроля.
    3. **"last-guid" :: char[37].** Глобальный идентификатор последнего события Fact.
    4. **"last-datetime" :: uint32_t.** Время регистрации изображения зоны контроля последнего события Fact, ISO8601.

5. **"last-ts" :: uint64_t.** Время регистрации изображения зоны контроля последнего события Fact, мкс (от 01.01.1970 00:00 UTC).

Пример тела ответа:


```
{
 "status": 200,
 "result":
 [
   {
     "plate-text": "В В4105 А А",
     "total-count": 14,
     "last-guid": "69e86fa7-e1ad-4e68-96b7-b910f40bdb49",
     "last-datetime": "2016-06-14T10:52:53.520817+00:00",
     "last-ts": 100600200400
   }
 ]
}
```


Прямая трансляция изображений (get_live-stream)

Структура запроса **get_live-stream** определяется форматом:

```
GET /api/v1/post/<post-guid>/channel/<channel-no>/media/live
```

где

- **<post-guid> :: char[37]** — глобальный идентификатор поста; возможны значения согласно **get_post-set_rbody-200** (см. **Запрос доступных постов, каналов поста и их параметров**).
- **<channel-no> :: int32_t** — номер канала поста; возможны значения согласно **get_post-channel-set_rbody-200** (см. **Запрос доступных постов, каналов поста и их параметров**).

Ответ содержит поток JPEG-изображений зоны контроля в виде Server-Push-документа с типом контента multipart/x-mixed-replace. Каждое изображение содержится в отдельной части Server-Push-документа. Поле **Content-Type** заголовка Server-Push-документа содержит параметр **boundary** (граница), который обозначает последовательность символов, разделяющих части сообщения.

Пример ответа:

```
HTTP/1.1 200 OK
Transfer-Encoding: chunked
```

```
Date: Mon, 28 Sep 2015 16:32:48 GMT
Server: Warp/3.0.13.1
Content-Type: multipart/x-mixed-replace;boundary=BOUNDARY
--BOUNDARY
Content-Type: image/jpeg
Content-Length: <jpeg-image-size>
<jpeg-image-data>
--BOUNDARY
Content-Type: image/jpeg
Content-Length: <jpeg-image-size>
<jpeg-image-data>
--BOUNDARY
Content-Type: image/jpeg
Content-Length: <jpeg-image-size>
<jpeg-image-data>
```