Модуль оповещения

«Тромбон IP-МО8»

Руководство по интеграции

ДВТР.425629.005Д2

Протокол Обмена API v.1

Москва 2022г.

Rev. 3 31.01.2023

Оглавление

1. Введение

1.1. Что такое API

API - это аббревиатура от Application Programming Interface, программный интерфейс приложения, интерфейс прикладного программирования. Описание способов (набор классов, процедур, функций, структур или констант), которыми одна компьютерная программа может взаимодействовать с другой программой. Набор этих правил может быть найден в этой документации к API, которая описывает как ваше приложение может взаимодействовать с «Тромбон IP - MO8».

1.2. Цель использования API протокола

«Тромбон IP - MO8» API разработан для того что бы позволить сторонним программам, таким как программы СКУД, централизованным станциям ОПС, серверам видеонаблюдения, а также другим, в зависимости от потребностей заказчика или системного интегратора, получать данные о состоянии прибора «Тромбон IP - MO8» и управлять им, например, включать режим оповещения по входу; в случае получения сигнала от видеосервера, принявшего решение о пожаре; опираясь на данные своей нейросетевой видеоаналитики; или выполнять сброс тревоги оператором, посредством нажатия им кнопки в некой централизованной программе управления зданием.

1.3. Особенности REST API протокола «Тромбон IP-MO8»

- Протокол представляет широкий набор функций для доступа к модулю и позволяет создавать гибкие пользовательские приложения с его помощью.
- API разработано с повышенным требованием к скорости ответов.

- Использование API возможно посредством программ, написанных с использованием широкого набора языков программирования, таких как Python, JavaScript, C++, Go и других.

1.4. Требования

- Все конечные точки (Endpoints) API являются защищёнными. Для доступа к ним требуется API токен. Для получения информации о создании, назначении прав и отзыве токенов, читайте в руководстве по эксплуатации прибора «Тромбон IP-MO8» ДВТР.425629.005РЭ.
- Nonce. API запросы требуют nonce (number used once), специальное число используемое один раз. Оно используется для безопасности и защищает от атак типа Reply. (Атака повторного воспроизведения - атака на систему аутентификации путём записи и последующего воспроизведения ранее посланных корректных сообщений или их частей). Прибор отклонит любые запросы с неверным nonce. Nonce каждого следующего запроса должен быть строго больше nonce предыдущего запроса, при перезапуске Прибора, nonce сбрасывается (устанавливается равным 0). Как правило nonce генерируется как unix timestamp - количество секунду с 1 января 1970 года. Nonce не должен превышать 64-битного unsigned long = 0xFFFFFFFFFFFFFFF или 18446744073709551615. Такое количество секунд с момента эпохи будет достигнуто в понедельник, 22 июля 2554 года, то есть не в обозримом будущем. Nonce может быть сгенерирован как, например, nonce = unixts * 1000.

Примечание - Проверяйте nonce на условия (nonce < 18446744073709551615).

Стоит иметь в виду, что в некоторых языках программирования есть свои ограничения на максимальные числа. Например в JavaScript константа Number.MAX_SAFE_INTEGER представляет максимальное безопасное целочисленное значение в JavaScript (2^{53} - 1).

Примечание - Учитывайте ограничения языка программирования при генерации nonce. Учитывайте также ограничения на количество запросов. Не рекомендуется осуществлять запросы чаще одного раза в секунду.

2. Главное

2.1. Описание

REST API (Representational State Transfer - «передача состояния представления»). Прибор «Тромбон IP-MO8» прослушивает http порт (8080) и получает от клиента обычный http запрос (request), возвращая клиенту ответ (response). Запрос должен содержать определённые заголовки, а в некоторых запросах тело и URL параметры. Тело запроса и ответ передаются в формате JSON.

2.2. URL

Для доступа к API используются адрес прибора + адрес API метода по протоколу http. Пример URL «http://192.168.0.20/api/v1/power». Текущий IP адрес прибора можно посмотреть на экране прибора в разделе «Настройки устройства» или через конфигуратор.

2.3. Заголовки запроса

Все запросы должны содержать следующие заголовки:

KEY	VALUE
trombon-apikey	Публичный ключ токена, которым подписывается запрос
trombon-nonce	Число nonce описанное в пункте 1.4
trombon-signature	Цифровая подпись запроса
content-type	application/json

В случае отсутствия любого из указанных заголовков в ответ на запрос вернётся сообщение об ошибке.

2.4. Подпись запроса

Подпись запроса осуществляется по алгоритму HMAC SHA-1 (hash-based message authentication code) с использованием

- пути запроса - «api/v1/method»
- nonce
- Тела POST запроса, если такой имеет место в методе.

закрытым ключом. После чего результат вычисления указанной функции подставляется в заголовок trombon-signature запроса.

```
const CryptoJS = require('crypto-js') // Standard JavaScript cryptography
library
const fetch = require('node-fetch') // "Fetch" HTTP req library

const apiKey = '1whI2fsp' // const apiKey = 'paste key here'
const apiSecret = 'nFntvulZTnvXuhq8' // const apiSecret = 'paste secret here'
const apiPath = 'api/v1/fire'// Example path
const nonce = (Date.now() * 1000).toString() // Standard nonce generator.
Timestamp * 1000
const body = {'alarm' : true} // Field you may change depending on endpoint
let signature = `${apiPath}${nonce}${JSON.stringify(body)}`
// Consists of the complete url, nonce, and request body
const sig = CryptoJS.HmacSHA1(signature, apiSecret).toString()
// The authentication signature is hashed using the private key
fetch(`http://127.0.0.1:8080/${apiPath}`, {
  method: 'POST',
  body: JSON.stringify(body),
  headers: {
    'Content-Type': 'application/json',
    'trombon-nonce': nonce,
    'trombon-apikey': apiKey,
    'trombon-signature': sig
  }
})
.then(res => res.json())
.then(json => console.log(json)) //Logs the response body
.catch(err => {
    console.log(err)
 })
```

Текст 1 - Пример программы для работы с API на языке JavaScript (http метод POST)

```
const CryptoJS = require('crypto-js') // Standard JavaScript cryptography
library
const fetch = require('node-fetch') // "Fetch" HTTP req library

const apiKey = '1whI2fsp' // const apiKey = 'paste key here'
const apiSecret = 'nFntvulZTnvXuhq8' // const apiSecret = 'paste secret
here'
const apiPath = 'api/v1/power'// Example path
const nonce = (Date.now() * 1000).toString() // Standard nonce generator.
let signature = `${apiPath}${nonce}`
// Consists of the complete url and nonce
const sig = CryptoJS.HmacSHA1(signature, apiSecret).toString()
// The authentication signature is hashed using the private key
fetch(`http://127.0.0.1:8080/${apiPath}`, {
  method: 'GET',
  headers: {
    'Content-Type': 'application/json',
    'trombon-nonce': nonce,
    'trombon-apikey': apiKey,
    'trombon-signature': sig
  }
})
.then(res => res.json())
.then(json => console.log(json)) //Logs the response body
.catch(err => {
    console.log(err)
 })
```

Текст 2 - Пример программы для работы с API на языке JavaScript (http метод GET)

```
#include "mainwindow.h"
#include "./ui_mainwindow.h"
#include <QJsonObject>
#include <QJsonDocument>
#include <QMessageAuthenticationCode>
#include <QNetworkAccessManager>
#include <QNetworkReply>

MainWindow::MainWindow(QWidget *parent)
    : QMainWindow(parent)
    , ui(new Ui::MainWindow)
{
    ui->setupUi(this);
    connect(ui->sendButton, &QAbstractButton::clicked, this,
&MainWindow::send);
}

MainWindow::~MainWindow() { delete ui; }

void MainWindow::send()
{
    QByteArray publicKey = ui->publicKeyEdit->text().toUtf8();
    QByteArray privateKey = ui->privateKeyEdit->text().toUtf8();
    QByteArray nonce = ui->nonceEdit->text().toUtf8();
    QByteArray address = ui->serverAddressEdit->text().toUtf8();
    QByteArray apiPath = ui->apiMethodEdit->text().toUtf8();
    QByteArray urlParams = ui->urlParams->text().toUtf8();
    QString fullPath = address + apiPath;
    if(!urlParams.isEmpty()) fullPath += "?" + urlParams;

    QJsonDocument doc = QJsonDocument::fromJson(ui->requestBodyEdit-
>toPlainText().toUtf8());
    QByteArray strBody(doc.toJson(QJsonDocument::Compact));

    QByteArray signature = apiPath + nonce + strBody;
    QByteArray signatureCalculated =
QMessageAuthenticationCode::hash(signature, privateKey,
QCryptographicHash::Sha1).toHex();

    QNetworkAccessManager *manager = new QNetworkAccessManager(this);
    connect(manager, &QNetworkAccessManager::finished, [this](QNetworkReply*
reply)
    {
        QJsonDocument doc = QJsonDocument::fromJson(reply->readAll());
        this->ui->responseView->setText(doc.toJson());
    });

    QNetworkRequest req;
    req.setRawHeader("trombon-apikey", publicKey);
    req.setRawHeader("trombon-nonce", nonce);
    req.setRawHeader("trombon-signature", signatureCalculated);
    req.setRawHeader("Content-Type", "application/json");
    QUrl url = QUrl(fullPath);
    req.setUrl(url);

    if(ui->getButton->isChecked()) { manager->get(req); return; }
    if(ui->postButton->isChecked()) { manager->post(req, strBody); return; }
}
```

Текст 3 - Пример программы для работы с API на языке Qt/C++

2.5. Параметры GET запросов

Ряд GET запросов может содержать параметры URL строки или URL-параметры. Например «api/v1/journal?strings=10»

3. Описание методов

3.1. «power»

Endpoint	api/v1/power
Разрешенные методы (Allowed http methods)	GET
GET URL-params	-
POST body	-
Ключевые слова	• power - Питание • mainpower - Основное питание • backuppower - Резервное питание • ok - Ок • loss - Потеряно • crit - Критический разряд • cap - Потеря ёмкости • low - Низкое напряжение

Указанный метод возвращает состояние систем питания прибором.

```
{
    "backupPower": "crit",
    "mainPower": "ok"
}
```

Текст 4 - Пример ответа метода "power" GET

3.2. «alarms»

Endpoint	api/v1/alarms
Разрешенные методы (Allowed http methods)	GET, POST
GET URL-params	-
POST body	`[ { "input": 1, "state": true }, { "input": 6, "state": true } ]` -
Ключевые слова	• alarms - Тревожные входы • input - Тревожный вход • state - Состояние

	• off - Отключен • fire - Пожар • error - Ошибка • display error - Ошибка отображения • wrong request structure - Неверная структура POST запроса • wrong input number - Неверный номер тревожного входа • alarm - Тревога

При использовании http метода GET указанный API метод возвращает состояние тревожных входов прибора. При использовании http метода POST указанный API метод позволяет установить состояние тревоги по выбранным тревожным входам, в ответ возвращая результат аналогичный GET.


```
[
    {
        "input": 1,
        "state": "fire"
    },
    {
        "input": 2,
        "state": "ok"
    },
    {
        "input": 3,
        "state": "ok"
    },
    {
        "input": 4,
        "state": "ok"
    },
    {
        "input": 5,
        "state": "ok"
    },
    {
        "input": 6,
        "state": "fire"
    },
    {
        "input": 7,
        "state": "ok"
    },
    {
        "input": 8,
        "state": "ok"
    }
]
```


Текст 5 - Пример ответа метода «alarms» POST, GET

В случае передачи в теле POST запроса неверного формата структуры, номеров тревожных входов больше 8 или меньше 1 или state типа не boolean, будет возвращена ошибка wrong request structure с http response code 500.

```
{
    "errors": [
        {
            "error": "wrong request structure"
        }
    ]
}
```

Текст 6 - Ошибка метода «alarms» передана неверная структура в теле POST запроса

3.3. «offinputs»

Endpoint	api/v1/offinputs
Разрешенные методы (Allowed http methods)	GET, POST
GET URL-params	-

POST body	<pre>[
	  {
	  "input": 1,
	  "state": true
	  },
	  {
	  "input": "emercom",
	  "state": true
	  }
	]</pre>

Ключевые слова	• offinputs - Отключение тревожных входов • input - Тревожный вход • emercom - Вход МЧС • state - Состояние • off - Отключен • on - Включен • display error - Ошибка отображения • wrong request structure - Неверная структура POST запроса • wrong input number - Неверный номер тревожного входа

При использовании http метода GET указанный API метод возвращает состояние отключения тревожных входов прибора. При использовании http метода POST указанный API метод позволяет установить состояние отключения по выбранным тревожным входам, в ответ возвращая результат аналогичный GET.

В случае передачи в теле POST запроса неверного формата структуры, номеров тревожных входов больше 8 или меньше 1 или state типа не boolean, будет возвращена ошибка wrong request structure с http response code 500.


```
[
    {
        "input": 1,
        "state": "off"
    },
    {
        "input": 2,
        "state": "on"
    },
    {
        "input": 3,
        "state": "on"
    },
    {
        "input": 4,
        "state": "on"
    },
    {
        "input": 5,
        "state": "on"
    },
    {
        "input": 6,
        "state": "on"
    },
    {
        "input": 7,
        "state": "on"
    },
    {
        "input": 8,
        "state": "on"
    },
    {
        "input": "emercom",
        "state": "off"
    }
]
```


Текст 7 - Пример ответа метода «offinputs» POST, GET

3.4. «controls»

Endpoint	api/v1/controls
Разрешенные методы (Allowed http methods)	GET
GET URL-params	-
POST body	-

Ключевые слова	• controls - Управление • locked - Заблокировано • unlocked - Разблокировано

Указанный API метод возвращает состояние системы блокировки управления.

```
{
    "controls": "unlocked"
}
```

Текст 8 - Пример ответа метода «controls» GET

3.5. «emercom»

Endpoint	api/v1/emercom
Разрешенные методы (Allowed http methods)	GET
GET URL-params	-
POST body	-
Ключевые слова	• emercom - МЧС • emercomSignal - Сигнал МЧС • yes - Да • no - Нет • emercomRelay - Реле МЧС • open - Открыто • closed - Закрыто

Указанный API метод возвращает состояние наличия сигнала МЧС и состояние выходного реле МЧС.

```
{
    "emercomRelay": "open",
    "emercomSignal": "no"
}
```

Текст 9 - Пример ответа метода «emercom» GET

3.6. «relays»

Endpoint	api/v1/relays
Разрешенные методы (Allowed http methods)	GET
GET URL-params	-
POST body	-
Ключевые слова	• relays - Реле • errorRelay - Реле неисправность • startRelay - Реле запуск • emercomRelay - Реле МЧС • closed - Закрыто • open - Открыто

Указанный API метод возвращает состояние выходных реле прибора.

```
{
    "emercomRelay": "open",
    "errorRelay": "closed",
    "startRelay": "open"
}
```

Текст 10 - Пример ответа метода «realys» GET

3.7. «journal»

Endpoint	api/v1/journal
Разрешенные методы (Allowed http methods)	GET
GET URL-params	• strings=(int) - количество строк журнала с конца
POST body	-
Ключевые слова	• journal - Журнал • strings - Строки • startRelay - Реле запуск • emercomRelay - Реле МЧС • closed - Закрыто • open - Открыто

Указанный API метод возвращает список записей журнала событий. При передаче URL параметра «strings» с целочисленным значением N, будут возвращены только N последних строк.

```
[
    "19.08.2021 13:48:42 Порт обмена данными с контроллером успешно открыт",
    "19.08.2021 13:48:42 [WARN]Неисправность - критический разряд или
неисправность батареи",
    "19.08.2021 13:48:45 [FIRE]Удаленная активация тревоги по входам: 1, 6",
    "19.08.2021 13:48:45 Трансляция тревожного сообщения в зоны: N3-10IP",
    "19.08.2021 13:48:46 Трансляция тревожного сообщения в зоны: N3-10IP",
    "19.08.2021 13:48:53 Нажата клавиша сброс",
    "19.08.2021 13:49:17 [WARN]Ошибка связи; N3-10IP офлайн",
    "19.08.2021 13:51:41 Отключены тревожные входы: 1, 6",
    "19.08.2021 13:53:02 Отключен вход МЧС",
    "19.08.2021 13:53:02 Отключены тревожные входы: 1"
]
```

Текст 11 - Пример ответа метода «journal» GET с URL параметром strings=10

3.8. «malfunctions»

Endpoint	api/v1/malfunctions
Разрешенные методы (Allowed http methods)	GET
GET URL-params	-
POST body	-
Ключевые слова	• malfunctions - Неисправности

Указанный API метод возвращает список неисправностей прибора и системы.

```
[
    "Критический разряд или неисправность батареи",
    "Ошибка связи; N3-10IP офлайн"
]
```

Текст 12 - Пример ответа метода «malfunctions» GET

3.9. «summary»

Endpoint	api/v1/summary
Разрешенные методы (Allowed http methods)	GET
GET URL-params	-
POST body	-
Ключевые слова	Используются ключевые слова всех остальных методов

Указанный API метод возвращает сводные данные по прибору.

3.10. «fire»

Endpoint	api/v1/fire
Разрешенные методы (Allowed http methods)	POST
GET URL-params	-
POST body	-
Ключевые слова	• fire - Пожар

Указанный API метод позволяет установить состояние тревоги по всем тревожным входам, в ответ возвращая результат аналогичный методу «alarms» GET. Имитирует нажатие клавиши «Пожар».


```
[
    {
        "input": 1,
        "state": "off"
    },
    {
        "input": 2,
        "state": "fire"
    },
    {
        "input": 3,
        "state": "fire"
    },
    {
        "input": 4,
        "state": "fire"
    },
    {
        "input": 5,
        "state": "fire"
    },
    {
        "input": 6,
        "state": "fire"
    },
    {
        "input": 7,
        "state": "fire"
    },
    {
        "input": 8,
        "state": "fire"
    }
]
```


Текст 13 - Пример ответа метода «fire» POST

3.11. «reset»

Endpoint	api/v1/reset
Разрешенные методы (Allowed http methods)	POST
GET URL-params	-
POST body	-
Ключевые слова	• reset - Сброс

Указанный API метод позволяет сбросить состояние тревоги по всем тревожным входам, в ответ возвращая результат аналогичный методу «alarms» GET. Имитирует нажатие клавиши «Сброс».

4. Описание возможных ошибок

В процессе выполнения запросов могут возникать различные ошибки. Если такое происходит, прибор вернёт структуру с типом ошибки, текстовым описанием ошибки и http кодом возврата соответствующему возникшей ошибке.

errorType	Http response code	Причины возникновения
authorization error	401	• Не переданы необходимы заголовки http запроса • Передан неверный nonce • Переданные данные авторизации не соответствую ни одному выпущенному API ключу • Неверная HMAC сигнатура запроса
forbidden	403	• Запрашиваемый API метод не доступен для этого токена
not implemented	405	• Выбранный http метод не реализован для запрашиваемого API метода
internal server error	500	• Внутренняя ошибка сервера (возникает при передаче неверных данных в теле POST запроса)
method not found	404	• Запрашиваемый API метод не существует

```
{
    "error": {
        "error message": "Nonce number must strictly inrease on each request",
        "error type": "authorization error"
    }
}
```

Текст 14 - Пример ответа с сообщением об ошибке