Протокол взаимодействия с интернет кассой

Версия 2.4.0 от 14.06.2017

Оглавление

[Изменения в документе 3](#_Toc485218906)

[1. Общее описание 4](#_Toc485218907)

[Пример запроса с подписью 4](#_Toc485218908)

[2. Запросы 6](#_Toc485218909)

[2.1 Создания чека 6](#_Toc485218910)

[2.1.1 Тело запроса 6](#_Toc485218911)

[2.1.1.1 Содержимое документа 6](#_Toc485218912)

[2.1.1.2 Позиция чека 6](#_Toc485218913)

[2.1.1.3 Параметры закрытия чека 6](#_Toc485218914)

[2.1.1.4 Оплата 7](#_Toc485218915)

[2.1.2 Тело ответа с ошибками валидации 8](#_Toc485218916)

[Пример запроса: 8](#_Toc485218917)

[2.2 Состояние чека 9](#_Toc485218918)

[2.2.1 Тело ответа 9](#_Toc485218919)

[Пример ответа: 9](#_Toc485218920)

[3. Swagger 11](#_Toc485218921)

# Изменения в документе

**Версия 0.1.0 от 13.02.2017**

Первая версия.

**Версия 0.2.0 от 17.02.2017**

Изменено содержимое чека и выдаваемые на запрос статуса документа данные.

**Версия 0.3.0 от 21.02.2017**

Изменено время выдаваемое в результате пробития чека на локальное, уточнение по цене товара, добавлен признак отправки в ОФД и данные квитанции.

**Версия 0.4.0 от 23.03.2017**

Добавлены значения параметров система налогообложения и тип оплаты, в запрос регистрации добавлен атрибут группа.

**Версия 0.5.0 от 13.04.2017**

Убраны признак отправки в ОФД и данные квитанции.

**Версия 2.0.0 от 24.05.2017**

Добавлен ИНН в запрос статуса документа. Изменена версию API на v2.

**Версия 2.1.0 от 29.05.2017**

Добавлено разъяснение по поводу группы по умолчанию.

Добавлено упоминание о swagger.

**Версия 2.2.0 от 01.06.2017**

Добавил примеры генерации подписи на php и .net core и уточнение по используемому паддингу.

**Версия 2.3.0 от 07.06.2017**

Добавлено описание 401 ошибки.

Добавлен пример генерации подписи на .net full.

**Версия 2.4.0 от 14.06.2017**

Добавлен пример генерации подписи на php.

# 1. Общее описание

Взаимодействие с интернет-кассой осуществляется посредством обмена сообщениями с web-сервисом поднимаемым устройством.

Транспорт – https с использованием клиентского и серверного сертификатов для взаимной аутентификации.

Формат сообщений – JSON.

Для защиты клиентских сообщений от изменений используется подпись SHA256-RSA с ключом не менее 2048 Кбит, паддинг Pkcs1.

Подпись передается в http-заголовке X-Signature закодированная с использованием base64 кодировки.

Интернет-касса имеет публичную часть ключа клиента и осуществляет проверку переданной подписи.

При генерации подписи используется сообщение конвертированное в байты с использованием кодировки UTF-8.

### Пример запроса с подписью

POST /api/v2/documents HTTP/1.1

X-Signature: WtL1bB9sZG9JQVYCd8lgf7tnatCBonKG3+asK3ypIfHCKpWH5IKVYGKcyw6k1jCPflLjjBJzr3/qeY4TCNj1qRmQt8SuxOdfp26gSCsFOy3crRvvzNxqxoxXSXSGY8sOUENvQSm9ZiTSm2YdKJCtQDDCJUXQMqd4CD3ygijP8UM/OPMTcQkeHMN4UV44EhWOt7TJvkOn1KZxvFV6p3FRwYtBSOLmXd/mAFFEGvSoRjFKH3fNsof4YYnGMTIoqM+TyD22AGthG9/u6WhT9lpOEMNQ7QcPEeYjAsvbzSQFyom9dvNUPVPGORhDR0FMPyn71ykiO4cHPkTzqzSnT/zhzQ==

Connection: Keep-Alive

Content-Length: 42

Content-Type: application/json; charset=utf-8

Expect: 100-continue

Host: localhost:5000

{

"DocumentNumber": "45eg9uw23KKK"

}

Ключ RSA:

<RSAKeyValue>

<Modulus>t8nC/Eth8UabQbXu8pdro3v7NqUanV8Y+g92YgT7z1xqkBLRHXZ1guml3PxrqjNX9AvOmu8R+qaKOyHfJW0PcRDLzCoIUcHNAwpDO/E5j6WAaLIv7gAjTtyr9kJB9rfJaparViJNZu3RSUYGTvVznOmXMf7LTOTMR6HP/5H1TP5n1g4+BbLmC9EhjUf2eNFqwZBqPtzybBb6jaHBRaJ0XdE3lh2OeE9/OF0BtLwiYPDKsVTxIekbNf7l/DREy+YbUOxQLceeHXrvbYLiGWecP0a7CqHGj9ZNY1oJThK3AwrSd4yHa9Wnx/GaZUNtWud1BaP9g3sVX+sRV9xtnI96dw==</Modulus>

<Exponent>AQAB</Exponent>

<P>3WSb72a1erb6jcLkyZA2Y21VNIipGz+ta1RP+iacs3xnktFsxgTYgqWyt6SWZ2rStp0u4vb/IAHyKhgJPNTUSi2u0G44MOsRxMC/FWTF8zdyrDF4BjPBM4j84nAmE/FQYv5F8ldDkakc96zEPiTk5Fka3MPeN8mMk6/OA59JdF0=</P>

<Q>1IRVid5SsDrOwJQAEKkdT436XEb0sVWe9AcU8JyaCEEMj0NPzownNbIrebPofMYdDHikopQpr2XqxZYDbb7AneoHkhEV26TfpPVbN4wBJFXih3lAP2n5hqhgqHGp5Wq2Lu7jUS376Ruw3bhwW+MiWpXv1xhMTZ8AtDfnZFFNvOM=</Q>

<DP>Fo5KiNCJCtCbpFfH4XVM5UJdXPXTbNBHBdlYMJ9AddTl5IJrt50ExgLFu4oMPMsYXryS61LI2WT5XCqIvmbcnhYbambgWLOKYuZUUYSr2kS67So5FUCunWaGhTdx2bRLQVqwm6kiXDPDnMRAViiCHXWqk/VsrXheVymhLqNK440=</DP>

<DQ>mowSWMzhfV+G8+2tjnAt7KjnpSvEzyHhEr4DsGdybQZBR/4/j4nFCfukOkFnlTXN8j/aGpF9Lx0C+uX5YFoUYcLL9qGOL8lbCu+TgnXCbtY2gybeXj+HQzI3+MeQMlLEYqU/ks3KIOAOY2+55ljrpszbOqVk+B3luSnekMm/qtk=</DQ>

<InverseQ>aP5e5F1j6s82Pm7dCpH3mRZWnfZIKqoNQIq2BO8vA9/WrdFI2C27uNhxCp2ZDMulRdBZcoeHcwJjnyDzg4I4gBZ2nSKkVdlN1REoTjLBBdlHi8XKiXzxvpItc2wjNC2AKHaJqj/dnh3bbTAQD1iUAxPmmLJYYkhfZ2i1IrTVxZE=</InverseQ>

<D>PUfM+Aq6kZSVWAetsL3EajKAxOuwQCDhVx+ovW4j+DQ8Y+WiTEyfShNV9qVD0PBltz3omch1GjpFhQn6OaRvraeIDH9HXttb3FOjr2zzYG4yrrYbPSRWoYj63ZWiIP2O7zdl0caGQHezfNcYa2N0NTG99DGc3/q6EnhlvjWQsSbiEjmxcPx8fmV1i4DoflMQ383nsixAFapgrROUAtCgMvhWn1kSeoojKd+e4eKZxa/SNYulsBJWNFkmo1CZH4YTqlPM+IwYeDUOnOUGNxGurRZ3qQdWs2N2ZQhnrvlh+zpzurD2hwAz6gQXP7mxxMR1xHtAD8XQ+w4OiJK6VWjoIQ==</D>

</RSAKeyValue>

Пример создания подписи в php с использованием https://github.com/phpseclib/phpseclib

function computeSignature($data)

{

    $rsa = new RSA();

    $rsa->setPrivateKey(($this->privateKey);

    $rsa->setPrivateKeyFormat(RSA::PRIVATE\_FORMAT\_XML);

    $rsa->setHash('sha256');

    $rsa->setMGFHash('sha256');

    $rsa->setSignatureMode(RSA::SIGNATURE\_PKCS1);

    return base64\_encode($rsa->sign($data));

}

Пример отправки данных на php:

$sign = $this->computeSignature(json\_encode($data, JSON\_UNESCAPED\_UNICODE));

$headers = [

    'Accept: application/json',

    'Content-Type: application/json',

    'X-Signature: '.$sign

];

$ch = new Curl();

$ch->init();

$ch->setOptions($x=[

    CURLOPT\_HEADER => false,

    CURLOPT\_SSL\_VERIFYPEER => false,

    CURLOPT\_SSLCERT => $this->client\_crt(),

    CURLOPT\_SSLKEY => $this->client\_key(),

    CURLOPT\_SSLCERTPASSWD => '1234',

    CURLOPT\_HTTPHEADER => $headers,

]);

$result = $ch->post(

    'https://apip.orangedata.ru:2443/api/v2/documents/',

    json\_encode($data, JSON\_UNESCAPED\_UNICODE));

Пример создания подписи на C# с использованием стандартной библиотеки классов .net core:

private string ComputeSignature(string document)

{

    var data = Encoding.UTF8.GetBytes(document);

    using (var rsa = RSA.Create())

    {

        rsa.FromXmlString(privateKey);

        return Convert.ToBase64String(rsa.SignData(data, HashAlgorithmName.SHA256, RSASignaturePadding.Pkcs1));

    }

}

Пример создания подписи на C# с использованием стандартной библиотеки классов .net framework full:

private string ComputeSignature(string document)

{

    var data = Encoding.UTF8.GetBytes(document);

    using (var rsa = new RSACryptoServiceProvider())

    {

        rsa.FromXmlString(privateKey);

        return Convert.ToBase64String(rsa.SignData(data, "SHA256"));

    }

}

# 2. Запросы

## 2.1 Создания чека

Создание чека – асинхронный запрос, после его выполнения чек оказывается в очереди на обработку. Помимо данных чека клиент передает уникальный идентификатор документа использую который в последствии, он сможет запросить состояние чека(п.2.2). Плюс, данный идентификатор обеспечивает идемпотентность запросов. Данный идентификатор должен быть уникальным в пределах организации.

ИНН служит для определения устройства с помощью которого может быть проведен чек.

Group в запросе может быть null или не указан, в таком случае чек отправится в группу по умолчанию.

Группой по умолчанию считается группа с названием Main.

Запрос: **POST** **/api/v2/documents/**

### 2.1.1 Тело запроса

|  |  |  |
| --- | --- | --- |
| Id | Идентификатор документа | Строка от 1 до 32 символов |
| INN | ИНН организации, для которой пробивается чек | Строка 12 символов |
| Group | Группа устройств, с помощью которых будет пробит чек | Строка от 1 до 32 символов или null |
| Content | Содержимое документа | Структура п.2.1.1.1 |

### 2.1.1.1 Содержимое документа

|  |  |  |
| --- | --- | --- |
| Type | Тип документа:   1. Приход 2. Возврат прихода 3. Расход 4. Возврат расхода | Число |
| Positions | Список позиций в чеке | Массив структур п.2.1.1.2 |
| CheckClose | Параметры закрытия чека | Структура п.2.1.1.3 |
| CustomerContact | Телефон или e-mail покупателя | Строка до 64 символов, формат +{Ц} или {С}@{C} |

### 2.1.1.2 Позиция чека

|  |  |  |
| --- | --- | --- |
| Quantity | Количество товара | Десятичное число с точностью до 3 символов после точки |
| Price | Цена товара с учетом всех скидок и наценок | Десятичное число с точностью до 2 символов после точки |
| Tax | Ставка НДС:  1 – ставка НДС 18%  2 – ставка НДС 10%  3 – ставка НДС расч. 18/118  4 – ставка НДС расч. 10/110  5 – ставка НДС 0%  6 – НДС не облагается | Число от 1 до 6 |
| Text | Текст позиции | Строка до 128 символов |

### 2.1.1.3 Параметры закрытия чека

|  |  |  |
| --- | --- | --- |
| Payments | Оплаты | Массив структур п.2.1.1.4 |
| TaxationSystem | Система налогообложения:  0 – Общая, ОСН  1 – Упрощенная доход, УСН доход  2 – Упрощенная доход минус расход, УСН доход - расход  3 – Единый налог на вмененный доход, ЕНВД  4 – Единый сельскохозяйственный налог, ЕСН  5 – Патентная система налогообложения, Патент | Число от 0 до 5 |

### 2.1.1.4 Оплата

|  |  |  |
| --- | --- | --- |
| Type | Тип оплаты:  1 – Наличными  2 – Картой Мир  3 – Картой Visa  4 – Картой MasterCard  5 – Расширенная оплата 1  6 – Расширенная оплата 2  7 – Расширенная оплата 3  8 – Расширенная оплата 4  9 – Расширенная оплата 5  10 – Расширенная оплата 6  11 – Расширенная оплата 7  12 – Расширенная оплата 8  13 – Расширенная оплата 9  14 – Предвариательная оплата(Аванс)  15 – Последующая оплата(Кредит)  16 – Иная форма оплаты | Число от 1 до 16 |
| Amount | Сумма оплаты | Десятичное число с точностью до 2 символов после точки |

Ответ: api может возвращать следующие статус-коды

* 201 Created – чек создан и добавлен в очередь на обработку, пустое тело ответа
* 401 Unauthorized – клиентский сертификат не прошел проверку
* 409 Conflict– чек с данным идентификатором уже был создан в системе, пустое тело ответа
* 400 Bad Request – переданные данные содержат ошибки, тело ответа п.2.1.2

### 2.1.2 Тело ответа с ошибками валидации

|  |  |  |
| --- | --- | --- |
| Errors | Массив ошибок валидации запроса | Массив строк |

### Пример запроса:

{

"Id": "12345678990",

"INN": "123456789012",

"Content": {

"Type": 1,

"Positions": [

{

"Quantity": 1.000,

"Price": 123.45,

"Tax": 6,

"Text": "Булка"

},

{

"Quantity": 2.000,

"Price": 4.45,

"Tax": 4,

"Text": "Спички"

}

],

"CheckClose": {

"Payments": [

{

"Type": 1,

"Amount": 123.45

},

{

"Type": 2,

"Amount": 8.90000

}

],

"TaxationSystem": 1

},

"CustomerContact": "foo@example.com"

}

}

## 2.2 Состояние чека

Запрос: **GET** **/api/v2/documents/****{inn}/status****/{document\_id}**

**{inn}** – ИНН организации, для которой пробивается чек

**{document\_id}** –идентификатор документа, который был указан при его создании

В данном запросе подпись SHA256-RSA не используется.

Ответ: api может возвращать следующие статус-коды

* 202 Accepted – чек создан и добавлен в очередь на обработку, но еще не обработан, пустое тело ответа
* 404 Not Found – чек с указанным идентификатором не найден в системе, пустое тело ответа
* 401 Unauthorized – клиентский сертификат не прошел проверку
* 200 OK – чек обработан, тело ответа п.2.2.1

### 2.2.1 Тело ответа

|  |  |  |
| --- | --- | --- |
| Id | Идентификатор документа | Строка от 1 до 32 символов |
| DeviceSN | Заводской номер устройства пробившего чек | Строка до 20 символов |
| DeviceRN | Регистрационный номер устройства пробившего чек | Строка до 20 символов |
| FSNumber | Номер фискального накопителя | Строка 16 символов |
| OFDName | Наименование ОФД | Строка до 256 символов |
| OFDWebsite | Web-сайт ОФД | Строка до 58? символов |
| OFDINN | ИНН ОФД | Строка 12 символов |
| FNSWebsite | Web-сайт ФНС | Строка до 256 символов |
| CompanyINN | ИНН пользователя | Строка 12 символов |
| CompanyName | Наименование пользователя | Строка до 256 символов |
| DocumentNumber | Номер ФД | Число |
| ShiftNumber | Номер смены | Число |
| DocumentIndex | Номер чека за смену | Число |
| ProcessedAt | Время регистрации фискального документа в ФН | Время в виде строки в формате ISO8601 |
| Content | Содержимое документа | Структура п.2.1.1.1 |
| Change | Сдача | Десятичное число с точностью до 2 символов после точки |
| FP | Фискальный признак | Строка 10 символов |

### Пример ответа:

{

"Id": "12345678990",

"DeviceSN": "0000000000001358",

"DeviceRN": "0000000400054952",

"FSNumber": "9999078900001341",

"OFDName": "ООО \"Ярус\"(\"ОФД-Я\")",

"OFDWebsite": "www.ofd-ya.ru",

"OFDINN": "7728699517",

"FNSWebsite": "www.nalog.ru",

"CompanyINN": "123456789012",

"CompanyName": "ЗАО ТОРГОВЫЙ ОБЪЕКТ №1",

"DocumentNumber": 117,

"ShiftNumber": 20,

"DocumentIndex": 5,

"ProcessedAt": "2017-02-14T14:16:00",

"Content": {

"Type": 1,

"Positions": [

{

"Quantity": 1.000,

"Price": 123.45,

"Tax": 6,

"Text": "Булка"

},

{

"Quantity": 2.000,

"Price": 4.45,

"Tax": 4,

"Text": "Спички"

}

],

"CheckClose": {

"Payments": [

{

"Type": 1,

"Amount": 123.45

},

{

"Type": 2,

"Amount": 8.90000

}

],

"TaxationSystem": 1

},

"CustomerContact": "+79123456789"

},

"Change": 974.01,

"FP": "2364009522",

}

# 3. Swagger

Описание api доступно в виде OpenAPI Specification(OAS).

По url **/swagger** доступен swagger-ui

По url **/swagger/v2/swagger.json** – спецификация.