Уведомления обратного вызова

Программный интерфейс платёжного шлюза позволяет вам получать уведомления обратного вызова об изменении состояния заказа, а также о ряде других событий, происходящих с заказом.

Общие сведения

Операции, по которым могут быть получены уведомления

Продавец может получать от платёжного шлюза автоматические уведомления об операциях с заказами, указанных в таблице ниже.

Уведомления об операциях обратного вызова - это не уведомления по электронной почте и не уведомления по телефону. Это уведомления, принимаемые через программный интерфейс.

Операция	**Тип платежа**
Холдирование (удержание) средств.	Только двухстадийные платежи.
Списание средств.	Одностадийные и двухстадийные платежи.
Отмена перевода средств.	
Возврат средств.	

Типы уведомлений

Уведомления могут быть двух типов (см. таблицу ниже).

Тип уведомления	**Описание**
Уведомления без контрольной суммы.	Такие уведомления содержат только сведения о заказе - потенциально продавец рискует принять уведомление, отправленное злоумышленником, за подлинное.

Тип уведомления	Описание
Уведомления с контрольной суммой.	Такие уведомления, помимо сведений о заказе, содержат аутентификационный код. Аутентификационный код представляет собой контрольную сумму сведений о заказе. Эта контрольная сумма позволяет убедиться, что callback-уведомление действительно было отправлено платёжным шлюзом. Существует два способа реализации callback-уведомлений с контрольной суммой: • с помощью симметричной криптографии - для формирования контрольной суммы на стороне шлюза и для её проверки на стороне продавца используется один и тот же (симметричный) криптографический ключ; • с помощью асимметричной криптографии - для формирования контрольной суммы на стороне платёжного шлюза используется закрытый ключ, известный только шлюзу, а для подтверждения контрольной суммы используется связанный с закрытым ключом открытый ключ, который известен продавцам и может распространяться свободно. Открытый ключ можно выгрузить из личного кабинета при наличии соответствующих полномочий. Для большей безопасности рекомендуется использовать способ, в котором контрольная сумма формируется с помощью асимметричной криптографии. Чтобы включить возможность получения уведомлений с контрольной суммой и получить закрытый ключ, обратитесь в службу технической поддержки.

Требования к SSL-сертификатам сайта продавца

Если для доступа к магазину, который работает с уведомлениями о состоянии заказов, используется HTTPS-соединение, сертификат сайта, на котором расположен этот магазин, должен соответствовать следующим требованиям (см. таблицу ниже).

Требование	Описание
Длина и тип ключа сертификата.	Ключ RSA не менее 2048 бит.
Алгоритм подписи.	Не ниже SHA-256.

Требование	Описание
Поддерживаемые центры сертификации.	Ниже представлены примеры организаций, осуществляющих регистрацию сертификатов: • Thawte Consulting cc – https://www.thawte.com/; • VeriSign – https://www.verisign.com/; • DigiCert Inc – https://www.digicert.com/; • COMODO CA Limited – https://www.comodo.com/; • GeoTrust Inc. – https://www.geotrust.com/; • GlobalSign – https://www.globalsign.com/; • Trustis Limited – http://www.trustis.com/; • UniTrust – http://www.unitrust.co.uk/; • Let's Encrypt - https://letsencrypt.org/. Также существует возможность оформления сертификатов через поставщиков в России: • RU-CENTER – http://ssl.ru/; • REG.RU – https://www.reg.ru/ssl-certificate/; • MySSL – https://myssl.ru/. Не допускается использование самоподписанных сертификатов. Сертификат должен быть подписан доверенным центром сертификации (см. выше).

Формат URL-адресов уведомлений

Уведомление без контрольной суммы

```
{merchant-
url}?mdOrder={mdOrder}&orderNumber={orderNumber}&operation={operation}&status
={status}&callbackCreationDate={callbackCreationDate}
```

Уведомление с контрольной суммой

```
{merchant-
url}?mdOrder={mdOrder}&orderNumber={orderNumber}&checksum={checksum}&operatio
n={operation}&status={status}&callbackCreationDate={callbackCreationDate}
```

Передаваемые параметры представлены в таблице ниже.

В таблице приведены базовые параметры. В личном кабинете также существует возможность указать ряд дополнительных параметров, которые будут передаваться в уведомлениях.

Параметр	Описание
mdOrder	Уникальный номер заказа в системе платёжного шлюза.
orderNumber	Уникальный номер (идентификатор) заказа в системе продавца.
sign_alias	Алиас (имя) приватного ключа для подписи.
checksum	Аутентификационный код, или контрольная сумма, полученная из набора параметров.
callbackCreationDate	Время создания запроса уведомления обратного вызова.
operation	Тип операции, о которой пришло уведомление: • approved - операция удержания (холдирования) суммы; • declinedByTimeout - операция отклонения заказа по истечении его времени жизни; • deposited - операция завершения; • reversed - операция отмены; • refunded - операция возврата. Также типы операций, применимых только для **SberPay**: • bindingActivated - связка была активирована; • bindingDeactivated - связка была деактивирована; • bindingCreated - данные связки были изменены.
status	Индикатор успешности операции, указанной в параметре operation: • 1 - операция прошла успешно; • 0 - операция завершилась ошибкой.

При активации/деактивации связок (operation=bindingDeactivated и operation=bindingActivated) возвращаются следующие параметры:

- clientId
- bindingId
- operation=bindingDeactivated or bindingActivated
- enabled=false or true

Пример URL-адреса уведомления без контрольной суммы

```
https://myshop.ru/callback/?mdOrder=1234567890-098776-234-522&orderNumber=098
7&operation=deposited&callbackCreationDate=Mon Jan 31 21:46:52 MSK
2022&status=0
```

Пример URL-адреса уведомления с контрольной суммой

```
https://myshop.ru/callback/?mdOrder=1234567890-098776-234-522&orderNumber=098
7&checksum=DBBE9E54D42072D8CAF32C7F660DEB82086A25C14FD813888E231A99E1220AB3&o
```

peration=deposited&callbackCreationDate=Mon Jan 31 21:46:52 MSK 2022&status=0

Кастомные заголовки callback-уведомлений

В панели администратора в настройках мерчанта можно задавать кастомные заголовки для callback-уведомлений. Если они заданы, то такие заголовки будут отражаться в самих уведомлениях соответственно, например:

'http://kzntest.ru/callback.php', headers={Authorization=token, Content-type=plain /text}, params={orderNumber=349002, mdOrder=5ffb1899-cd1e-7c1e-8750-e98500093c42, operation=deposited, status=1}

где {Authorization=token, Content-type=plain/text} является кастомным заголовком.

Алгоритм обработки уведомлений о состоянии заказов

В разделах ниже представлен алгоритм обработки уведомлений о состоянии заказов в зависимости от типа таких уведомлений.

Уведомление без контрольной суммы

1. Платёжный шлюз отправляет на сервер продавца HTTP-запрос **GET** следующего вида.

   https://myshop.ru/callback/?mdOrder=1234567890-098776-234-522&orderNumber=0987&operation=deposited&status=0

2. Сервер отправляет в платёжный шлюз HTTP-код 200 OK.

Уведомление с контрольной суммой

1. Платёжный шлюз отправляет на сервер продавца HTTP-запрос **GET** следующего вида, при этом:
   - при использовании симметричной криптографии контрольная сумма формируется с помощью ключа, общего для платёжного шлюза и продавца;
   - при использовании асимметричной криптографии контрольная сумма формируется с помощью закрытого ключа, известного только платёжному шлюзу.

     http://site.ru/path?amount=123456&orderNumber=10747&checksum=DBBE9E54D42072D8CAF32C7F660DEB82086A25C14FD813888E231A99E1220AB3&mdOrder=3ff6962a-7dcc-4283-ab50-a6d7dd3386fe&operation=deposited&status=1

     Порядок параметров в уведомлении может быть произвольным.

2. На стороне продавца из строки параметров уведомления удаляется параметр `checksum` и `sign_alias`, а значение этого параметра (контрольная сумма) сохраняется для проверки подлинности уведомления.
3. Из оставшихся параметров и их значений генерируется строка следующего вида.

   имя_параметра1;значение_параметра1;имя_параметра2;значение_параметра2;...;имя_параметраN;значение_параметраN;

   В качестве разделителя следует использовать точку с запятой («;») без пробела. Также строка должна заканчиваться на точку с запятой («;»).

   При этом на стороне продавца пары имя_параметра;значение_параметра должны быть отсортированы в прямом алфавитном порядке по имени параметров.

   Пример сгенерированной строки параметров представлен ниже.

   ```
   amount;123456;mdOrder;3ff6962a-7dcc-4283-ab50-a6d7dd3386fe;operation;deposited;orderNumber;10747;status;1;
   ```

Обратите внимание, что строка так же должна заканчиваться на точку с запятой («;»).

1. На стороне продавца высчитывается контрольная сумма, способ вычисления зависит от способа её формирования:
   - при использовании симметричной криптографии - с помощью алгоритма HMAC-SHA256 и общего с платёжным шлюзом закрытого ключа;
   - при использовании асимметричной криптографии - с помощью алгоритма хеширования SHA512withRSA, который зависит от способа создания ключевой пары, и открытого ключа, который связан с закрытым ключом, находящимся на стороне платёжного шлюза..
2. В получившейся строке контрольной суммы все буквы нижнего регистра заменяются на буквы верхнего регистра.
3. Происходит сравнение полученного значения с контрольной суммой, извлечённой ранее из параметра `checksum`.
4. Если контрольные суммы совпадают, сервер отправляет в платёжный шлюз HTTP-код `200 OK`.

Если контрольные суммы совпадают, это уведомление подлинно и было отправлено платёжным шлюзом. В противном случае вероятно, что злоумышленник пытается выдать своё уведомление за уведомление платёжного шлюза.

Если в платёжный шлюз возвращается ответ, отличный от HTTP-кода 200 OK, отправка уведомления считается неуспешной. В этом случае платёжный шлюз повторяет отправку

уведомления с интервалом 10 минут до тех пор, пока не будет достигнуто одно из следующих условий:

- платёжный шлюз получает HTTP-код 200 OK в ответ на callback-уведомление

или

- происходит четыре неуспешные попытки информирования подряд.

По достижении одного из указанных выше условий попытки отправки callback-уведомлений об операции прекращаются.

Примеры кода

Асимметричная криптография

Java

```
package ru.bpc.test;


import org.apache.commons.codec.binary.Base64;

import org.apache.commons.codec.binary.Hex;


import java.io.ByteArrayInputStream;

import java.io.InputStream;

import java.security.Signature;

import java.security.cert.CertificateFactory;

import java.security.cert.X509Certificate;

import java.util.Comparator;

import java.util.Map;

import java.util.stream.Collector;

import java.util.stream.Collectors;

import java.util.stream.Stream;
```

```
public class App99 {

    public static void main(String[] args) throws Exception {

        String callbackParamsString = "amount=35000099, sign_alias=SHA-256
with RSA,
checksum=163BD9FAE437B5DCDAAC4EB5ECEE5E533DAC7BD2C8947B0719F7A8BD17C101EBDBE
ACDB295C10BF041E903AF3FF1E6101FF7DB9BD024C6272912D86382090D5A7614E174DC034EB
BB541435C80869CEED1F1E1710B71D6EE7F52AE354505A83A1E279FBA02572DC4661C1D75ABF
5A7130B70306CAFA69DABC2F6200A698198F8, mdOrder=12b59da8-
f68f-7c8d-12b5-9da8000826ea, operation=deposited, status=1";



        Map<String, String> callbackParamsMap =
Stream.of(callbackParamsString.split(","))

                .map(String::trim)

                .map(s -> s.split("="))

                .collect(Collectors.toMap(s -> s[0].trim(), s ->
s[1].trim()));



        String checksum = callbackParamsMap.get("checksum");

        callbackParamsMap.remove("checksum");

        callbackParamsMap.remove("sign_alias");



        String signString = callbackParamsMap.entrySet().stream()

                .sorted(Map.Entry.comparingByKey(Comparator.naturalOrder()))

                .collect(Collector.of(

                        StringBuilder::new,

                        (accumulator, element) -> accumulator

                                .append(element.getKey()).append(";")

                                .append(element.getValue()).append(";"),

                        StringBuilder::append,

                        StringBuilder::toString
```

```
                ));



        String cert =
"MIICcTCCAdqgAwIBAgIGAWAnZt3aMA0GCSqGSIb3DQEBCwUAMHwxIDAeBgkqhkiG9w0BCQEWEWt
6\n" +

"bnRlc3RAeWFuZGV4LnJ1MQswCQYDVQQGEwJSVTESMBAGA1UECBMJVGF0YXJzdGFuMQ4wDAYDVQQ
H\n" +

"EwVLYXphbjEMMAoGA1UEChMDUkJTMQswCQYDVQQLEwJRQTEMMAoGA1UEAxMDUkJTMB4XDTE3MTI
w\n" +

"NTE2MDEyMFoXDTE4MTIwNTE2MDExOVowfDEgMB4GCSqGSIb3DQEJARYRa3pudGVzdEB5YW5kZXg
u\n" +

"cnUxCzAJBgNVBAYTAlJVMRIwEAYDVQQIEwlUYXRhcnN0YW4xDjAMBgNVBAcTBUthemFuMQwwCgY
D\n" +

"VQQKEwNSQlMxCzAJBgNVBAsTAlFBMQwwCgYDVQQDEwNSQlMwgZ8wDQYJKoZIhvcNAQEBBQADgY0
A\n" +

"MIGJAoGBAJNgxgtWRFe8zhF6FE1C8s1t/dnnC8qzNN+uuUOQ3hBx1CHKQTEtZFTiCbNLMNkgWtJ
/\n" +

"CRBBiFXQbyza0/Ks7FRgSD52qFYUV05zRjLLoEyzG6LAfihJwTEPddNxBNvCxqdBeVdDThG81zC
0\n" +

"DiAhMeSwvcPCtejaDDSEYcQBLLhDAgMBAAEwDQYJKoZIhvcNAQELBQADgYEAfRP54xwuGLW/Cg0
8\n" +

"ar6YqhdFNGq5TgXMBvQGQfRvL7W6oH67PcvzgvzN8XCL56dcpB7S8ek6NGYfPQ4K2zhgxhxpFED
H\n" +

"PcgU4vswnhhWbGVMoVgmTA0hEkwq86CA5ZXJkJm6f3E/J6lYoPQaKatKF24706T6iH2htG4Bkjr
e\n" +

                "gUA=";



        byte[] b = Base64.decodeBase64(cert);

        CertificateFactory certFactory =
CertificateFactory.getInstance("X.509");



        InputStream in = new ByteArrayInputStream(b);
```

```
        X509Certificate x509Cert =
(X509Certificate)certFactory.generateCertificate(in);



        Signature sig = Signature.getInstance("SHA512withRSA");

        sig.initVerify(x509Cert.getPublicKey());



        sig.update(signString.getBytes());

        boolean verifies =
sig.verify(Hex.decodeHex(checksum.toLowerCase().toCharArray()));

        System.out.println("signature verifies: " + verifies);

    }

}
```

Симметричная криптография

Java

```
import org.apache.commons.codec.binary.Hex;

import javax.crypto.Mac;
import javax.crypto.spec.SecretKeySpec;
import java.nio.charset.StandardCharsets;
import java.security.InvalidKeyException;
import java.security.NoSuchAlgorithmException;

public class Example {

    public static String generateHMacSHA256(final String key, final String
data) throws InvalidKeyException, NoSuchAlgorithmException {

        final Mac hMacSHA256 = Mac.getInstance("HmacSHA256");
        byte[] hmacKeyBytes = key.getBytes(StandardCharsets.UTF_8);

        final SecretKeySpec secretKey = new SecretKeySpec(hmacKeyBytes,
"HmacSHA256");
        hMacSHA256.init(secretKey);

        byte[] dataBytes = data.getBytes(StandardCharsets.UTF_8);
```

```
        byte[] res = hMacSHA256.doFinal(dataBytes);

        return new String(Hex.encodeHex(res));
    }

    public static void main(String[] args) throws NoSuchAlgorithmException,
InvalidKeyException {
        String secretToken = "123";
        String message = "amount;1500;mdOrder;ed6f3abf-cea0-427e-
afdf-0ba43ead124f;operation;deposited;orderNumber;89312;status;1;";

        String signature = Expample.generateHMacSHA256(secretToken,
message).toUpperCase();
        System.out.println(signature);
    }
```

Симметричная криптография

PHP

```
<?php

$data = 'amount;123456;mdOrder;3ff6962a-7dcc-4283-ab50-
a6d7dd3386fe;operation;deposited;orderNumber;10747;status;1;';
$key = 'yourSecretToken';
$hmac = hash_hmac ( 'sha256' , $data , $key);

echo "[$hmac]\n";
?>
```

1. Пропишите строку в переменную `data`.
2. В переменную `key` пропишите закрытый ключ.
3. Функция `hash_hmac ( 'sha256', $data, $key)` вычисляет контрольную сумму от переданной строки, с помощью закрытого ключа по алгоритму SHA-256.
4. Сохраните результат работы функции в переменной `hmac`.
5. Выведите результат работы функции функцией `echo`.
6. Сравните это значение с тем, что передано в уведомлении о состоянии заказа.

Асимметричная криптография

PHP

```
<?php
```

```
// data from response
$data = 'amount;35000099;mdOrder;12b59da8-
f68f-7c8d-12b5-9da8000826ea;operation;deposited;status;1;';
$checksum =
'9524FD765FB1BABFB1F42E4BC6EF5A4B07BAA3F9C809098ACBB462618A9327539F975FEDB4C
F6EC1556FF88BA74774342AF4F5B51BA63903BE9647C670EBD962467282955BD1D57B16935C9
56864526810870CD32967845EBABE1C6565C03F94FF66907CEDB54669A1C74AC1AD6E39B67FA
7EF6D305A007A474F03B80FD6C965656BEAA74E09BB1189F4B32E622C903DC52843C454B7ACF
76D6F76324C27767DE2FF6E7217716C19C530CA7551DB58268CC815638C30F3BCA3270E1FD44
F63C14974B108E65C20638ECE2F2D752F32742FFC5077415102706FA5235D310D4948A780B08
D1B75C8983F22F211DFCBF14435F262ADDA6A97BFEB6D332C3D51010B';

// your public key (e.g. SHA-512 with RSA)
// if you have a CERT, please see openssl_get_publickey()
$publicKey = <<<EOD
-----BEGIN PUBLIC KEY-----
MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAwtuGKbQ4WmfdV1gjWWys
5jyHKTWXnxX3zVa5/Cx5aKwJpOsjrXnHh6l8bOPQ6Sgj3iSeKJ9plZ3i7rPjkfmw
qUOJ1eLU5NvGkVjOgyi11aUKgEKwS5Iq5HZvXmPLzu+U22EUCTQwjBqnE/Wf0hnI
wYABDgc0fJeJJAHYHMBcJXTuxF8DmDf4DpbLrQ2bpGaCPKcX+04POS4zVLVCHF6N
6gYtM7U2QXYcTMTGsAvmIqSj1vddGwvNGeeUVoPbo6enMBbvZgjN5p6j3ItTziMb
Vba3m/u7bU1dOG2/79UpGAGR10qEFHiOqS6WpO7CuIR2tL9EznXRc7D9JZKwGfoY
/QIDAQAB
-----END PUBLIC KEY-----
EOD;

$binarySignature = hex2bin(strtolower($checksum));
$isVerify = openssl_verify($data, $binarySignature, $publicKey,
OPENSSL_ALGO_SHA512);
if ($isVerify == 1) {
    echo "signature ok\n";
} elseif ($isVerify == 0) {
    echo "bad (there's something wrong)\n";
} else {
    echo "error checking signature\n";
}
?>
```

Node.js

```
const fs = require("fs");
const crypto = require("crypto");
const CRT = fs.readFileSync("config/test-cert.cer");
const string =  "amount;100;mdOrder;efc638e8-f869-7e2a-a00f-
b84c5e1f6325;operation;deposited;orderNumber;15531113133;status;1;";
const checksum =
"42052D9CD86E88334D6E85F3295AD41AB867ED30FC38CDB59C4686F6E2F39CAEFA3B60A4FCD
A7A38D30A4AF6565DCB4EE54386E94154D9C7D58C1F0D6C623185BD9EBA370D2BFBCD87C7B60
```

```
F90716AD80669A84341FC64FEE6B1DB98911A30317403D4E0157D814CDD989428CB59AF88366
DBE9B5D352015BE2247D97BEA6B717DD0897B84CEB970AEF36B9ABB65423538E6878D7A67795
8DDB0FD5294CBC6902F4956576FDC1729C16E8F0C5A49648211E3E40F9D4FD4C17A226C355BF
FCEB6CF63CF15D353FC4C96963CA80F63945FEC68742C01F1354891FBE514955008CBF0A60F9
207863602DFB97084DCA4DEA8EB749C53B0FD91826FEB41DB9ED57480";
const verifier = crypto.createVerify("RSA-SHA512");
verifier.update(string);
const isVerify = verifier.verify(CRT, checksum, "hex");
console.log(isVerify);
```

Была ли статья полезна?

(7) (10)