# Introduction

The authentication is based on Internet-Draft: Signing HTTP Messages proposed as a part of the Web Payments work.

> 📘 **Old authentication**
>
> If you integrated Satispay using the Security Bearer or the Diffie Hellman Exchange previously used, you can continue using the old authentication solutions also on the new APIs.

Updated almost 2 years ago

# Step by step guide

Steps required to perform the authentication

1. Generate a pair of **RSA keys**
2. Obtain the **KeyId** using the dedicated API

Step 1. and 2. must be performed only once. While steps from 3. to 6. must be performed for each call.

3. Create the **Digest** of the body
4. Create the **String** to be signed
5. Create the **Signature**
6. Compose the **authentication header**

> 📘 **Libraries**
>
> Please check if any of the available libraries can be integrated in your system as this will simplify and speed up the integration process of Satispay.

Updated over 2 years ago

# 1. Generate RSA key pair

An RSA key pair includes a private and a public key used to generate and verify digital signatures.

---

You can generate a public and private RSA key pair with these commands:

Shell

```
openssl genrsa -out private.pem 4096
openssl rsa -in private.pem -outform PEM -pubout -out public.pem
```

Updated 3 months ago

# 2. Obtain the KeyId

POST https://authservices.satispay.com/g_business/v1/authentication_keys

API to retrieve the KeyId

### Save the KeyId

Activation codes generated from Satispay Dashboard (or provided manually for Sandbox Account) are disposable, then the KeyId must be saved after its creation.

### Formatting the Public key

Make sure that the public_key has the newline control character `\n` for each line of the content. See the example on the right.

### Production VS Sandbox account

Activation codes generated for the Production account only work on Production endpoints, while activation codes generated for the Sandbox account only work on Sandbox endpoints.

BODY PARAMS

**public_key** string required

RSA public key, in pkcs8 encoding, generated in the previous step

**token** string required

Activation code that can be generated from the Satispay Dashboard (or provided manually for Sandbox account)

RESPONSES

**200**
200

**400**
400

**403**
403

**404**
404

LANGUAGE

cURL PHP

DEFAULT

REQUEST Examples

```
curl --request POST \
  --url https://authservices.satispay.com/g_business/v1/authentication_keys \
  --header 'content-type: application/json' \
  --data '{
    "public_key": "-----BEGIN PUBLIC KEY-----\nMIGfMA0GCSqGSIb3DQEBAQUAA4GNADCBiQKE
    "token": "623ECX"
  }'
```

RESPONSE Examples

Choose an example:

200 - Result 400 - Result 403 - Result 404 - Result

Updated almost 2 years ago

← 1. Generate RSA key pair 3. Create the Digest →

# 3. Create the Digest

1. Hash the content of the body with `sha256` algorithm using Base64 as output
2. Concatenate `SHA-256=` and the string of step 1.

> 📘 **Empty body**
>
> If your call has no body, create the Digest using an empty string.

Body example

```
{
  "flow": "MATCH_CODE",
  "amount_unit": 100,
  "currency": "EUR"
}
```

Hash of the body

```
ZML76UQPYzw5yDTmhySnU1S8nmqGde/jhqOG5rpfVSI=
```

Digest example

```
Digest: SHA-256=ZML76UQPYzw5yDTmhySnU1S8nmqGde/jhqOG5rpfVSI=
```

## Code sample 1/4

Shell PHP Node.js

```
BODY="{\n  \"flow\": \"MATCH_CODE\",\n  \"amount_unit\": 100,\n  \"currency\": \"EUR\"\n}

DIGEST=$(echo -e "$BODY\c" | openssl dgst -sha256 -binary | base64)
```

## Digest checker

Tool that allow to check the Digest created. Output from this tool and from your code must be the same.

https://satispay-signature-test.glitch.me/digest

Updated over 1 year ago

← 2. Obtain the KeyId

4. Create the String →

# 4. Create the String

This is the String that will then be signed at the next step

For the String creation the following headers `(request-target)` `host` `digest` `date` must be used.

- For the header field name `(request-target)` generate the header field value by concatenating the lowercased HTTP verb, a space and the request path
- For the remaining headers create the string by concatenating the lowercased header field name followed with a colon `:`, a space and the header field value

### 🚧 Notes

- Leading and trailing optional whitespace in the header field value must be omitted
- Make sure that the order of the headers in the HTTP request is the same used to create the String
- Request path in the `(request-target)` must include the QUERY PARAMS
- The format of the `date` header must be `EEE, dd MMM yyyy HH:mm:ss O`
- If there are multiple instances of the same header field, all header field values associated with the header field must be concatenated, separated by a comma `,` and a space and be used in the order in which they will appear in the transmitted HTTP message

HTTP request example

```
POST /wally-services/protocol/tests/signature HTTP/1.1
Host: staging.authservices.satispay.com
Date: Mon, 18 Mar 2019 15:10:24 +0000
Content-Type: application/json
Content-Length: 123
Digest: SHA-256=ZML76UQPYzw5yDTmhySnU1S8nmqGde/jhqOG5rpfVSI=
{
  "flow": "MATCH_CODE",
  "amount_unit": 100,
```

```
  "currency": "EUR"
}
```

String example

```
(request-target): post /wally-services/protocol/tests/signature
host: staging.authservices.satispay.com
date: Mon, 18 Mar 2019 15:10:24 +0000
digest: SHA-256=ZML76UQPYzw5yDTmhySnU1S8nmqGde/jhqOG5rpfVSI=
```

## Code sample 2/4

Shell PHP Node.js

```
BODY="{\n  \"flow\": \"MATCH_CODE\",\n  \"amount_unit\": 100,\n  \"currency\": \"EUR\"\n}

DIGEST=$(echo -e "$BODY\c" | openssl dgst -sha256 -binary | base64)

STRING="(request-target): post /wally-services/protocol/tests/signature\nhost: staging.au
```

Updated 6 months ago

← 3. Create the Digest 5. Create the Signature →

# 5. Create the Signature

Signature is the signed String created at the previous step

Sign with RSA ( `rsa-sha256` ) algorithm the previously created String with your private key, using Base64 as output.

String example

```
(request-target): post /wally-services/protocol/tests/signature
host: staging.authservices.satispay.com
date: Mon, 18 Mar 2019 15:10:24 +0000
digest: SHA-256=ZML76UQPYzw5yDTmhySnU1S8nmqGde/jhqOG5rpfVSI=
```

Signature example

```
signature="C5yynRxJQG2VNdsH8yuGwgribKt1yzym8lYvTAwxFmjEf7akYgLeIGOkdZo5vE/oB7O7+kNgqHxPp9
```

> 🚧 **Do not encrypt but sign**
>
> Please note that the String must be signed, not encrypted

## Code sample 3/4

Shell PHP Node.js

```
# In this example private key is storend in "private.pem" file

BODY="{\n  \"flow\": \"MATCH_CODE\",\n  \"amount_unit\": 100,\n  \"currency\": \"EUR\"\n}

DIGEST=$(echo -e "$BODY\c" | openssl dgst -sha256 -binary | base64)

STRING="(request-target): post /wally-services/protocol/tests/signature\nhost: staging.au

SIGNATURE=$(echo -e "$STRING\c" | openssl dgst -sign private.pem -sha256 -binary | base64
```

# Signature checker

Tool that allow to check the Signature created. Output must be the same.

https://satispay-signature-test.glitch.me/signature

Updated 6 months ago

← 4. Create the String

6. Compose the Authorization header →

# 6. Compose the Authorization header

The header must be composed concatenating:

1. The previously obtained keyId
2. The RSA algorithm used to sign the string ( `rsa-sha256` )
3. The list of signed headers used when creating the String
4. The previously created signature

Example:

```
Authorization: Signature keyId="vefn...", algorithm="rsa-sha256", headers="(request-target) host date digest", signature="C5yy..."
```

HTTP request example

```
POST /wally-services/protocol/tests/signature HTTP/1.1
Host: staging.authservices.satispay.com
Date: Mon, 18 Mar 2019 15:10:24 +0000
Content-Type: application/json
Content-Length: 123
Digest: SHA-256=ZML76UQPYzw5yDTmhySnU1S8nmqGde/jhqOG5rpfVSI=
{
  "flow": "MATCH_CODE",
  "amount_unit": 100,
  "currency": "EUR"
}
```

Authorization header example

```
Authorization: Signature keyId="4ekqhmf77q95deciis2frre12el393rteletbrg4rffqri3n58lsjsvf6
```

## Code sample 4/4

Shell PHP Node.js

```
KEY_ID="Your Key ID"
# In this example private key is stored in "private.pem" file

BODY="{\n  \"flow\": \"MATCH_CODE\",\n  \"amount_unit\": 100,\n  \"currency\": \"EUR\"\n}
```

```
DIGEST=$(echo -e "$BODY\c" | openssl dgst -sha256 -binary | base64)

STRING="(request-target): post /wally-services/protocol/tests/signature\nhost: staging.au

SIGNATURE=$(echo -e "$STRING\c" | openssl dgst -sign private.pem -sha256 -binary | base64

AUTHORIZATION_HEADER="Authorization: Signature keyId=\"$KEY_ID\", algorithm=\"rsa-sha256\
```

Updated 6 months ago

← 5. Create the Signature

Test the Authentication →

# Test the Authentication

POST https://staging.authservices.satispay.com/wally-services/protocol/tests/signature

API to test your authentication. The API works with [GET|POST|PUT|DELETE|PATCH]

> **Sandbox only**
>
> Please note that this API works on Sandbox endpoint only.

## Response details

## Role

- `authentication_key.role = PUBLIC` => authentication failed
- `authentication_key.role = ONLINE_SHOP` => authentication success
- `authentication_key.role = DEVICE` => authentication success

## Hints

- The expected behavior of not passing any signature header is to get a 200 response with only the `role` key populated as `PUBLIC`
- If you get a 403 or 401 error, either the signature string is malformed or the key-id is wrong or the public key has not been formatted correctly
- If you get `PUBLIC` as role when providing the signature header, the key-id was recognized but the signature is wrong.

## If the Test Authentication is NOT working

If your authentication is not working and you still get `PUBLIC` role, we suggest to double check both digest and signature with our tools:

- Digest checker
- Signature checker

## If the Test Authentication IS working but not the other APIs

Make sure that:

- You are adding all the mandatory headers
- The Request path in the (request-target) includes the QUERY PARAMS

HEADERS

**Host** string required
The host declared in the signature

**Date** string required
The date declared in the signature

**Digest** string required
The digest declared in the signature

**Authorization** string required
Signature of the request

RESPONSE

**200**
200

LANGUAGE

cURL

AUTHENTICATION HEADER

**Header** Authorization

CURL

REQUEST

```
curl --request POST \
  --url https://staging.authservices.satispay.com/wally-services/protocol/tests/sig
  --header 'content-type: application/json' \
  --header 'host: staging.authservices.satispay.com' \
  --header 'date: Mon, 18 Mar 2019 15:10:24 +0000' \
  --header 'digest: SHA-256=ZML76UQPYzw5yDTmhySnU1S8nmqGde/jhqOG5rpfVSI=' \
  --header 'Authorization: Signature keyId="4ekqhmf7...", algorithm="rsa-sha256", h
  --data '{
```

```
 9   "flow": "MATCH_CODE",
10   "amount_unit": 100,
11   "currency": "EUR"
12 }'
```

RESPONSE Examples

Choose an example:

200 - Authentication Success 200 - Authentication Failed

Updated 22 days ago

← 6. Compose the Authorization header

Making requests →

# Metadata

Generic field that can be used to store generic info

This generic field can contain up to 20 key-value items with a maximum length of 45 for the key and of 500 chars for the value.
Metadata can be defined when using the Create payment or the Create authorization and then it can be changed at any time using the Update a payment or the Update authorization.
The field `phone_number` can be defined to pre-fill the mobile number when using the Create payment or the Create authorization.

Updated 8 months ago

# Create payment

POST https://authservices.satispay.com/g_business/v1/payments

API to create a payment

## Flows

When creating a payment you can use one of this flows:

- MATCH_CODE: to create a payment that has to be paid scanning a Dynamic Code
- MATCH_USER: to create a payment request for a specific consumer
- REFUND: to partially/completely refund a Payment that is ACCEPTED
- PRE_AUTHORIZED: to create a payment with a pre-authorized token

> 🚧 Currently payments can be refunded only through API

## Responses

## 200 OK

- **id** [string]: Unique ID of the payment. It can then be used to retrieve the Get payment details or Update the payment.
- **code_identifier** [string]: Generated code identifier which contains payment details
- **type** [string]: Type of payment ( `TO_BUSINESS` or `REFUND_TO_BUSINESS` )
- **amount_unit** [number]: Amount of the payment in cents
- **currency** [string]: Currency of the payment
- **status** [string]: Status of the payment ( `PENDING` or `ACCEPTED` )
- **expired** [boolean]: If true, the payment is expired
- **metadata** [object]: Metadata inserted within the payment request
- **sender** [object]: The sender actor of the payment
  - **id** [string]: Unique ID of the sender
  - **type** [string]: Type of the actor ( `CONSUMER` or `SHOP` )
  - **name** [string]: Short name of the actor
- **receiver** [object]: The receiver actor of the payment
  - **id** [string]: Unique ID of the receiver

    - **type** [string]: Type of the actor ( `SHOP` or `CONSUMER` )
- **insert_date** [datetime]: Timestamp of payment insertion
- **expire_date** [datetime]: Timestamp of payment expiration
- **external_code** [string]: Order ID or payment external identifier
- **redirect_url** [string]: Redirect url to the payment page

## 400 Bad Request

- **code** [integer]: Error code
  `21` → insufficient availability
  `36` → malformed flow, payment or metadata
  `131` → payment too old to be refunded (it is possible to refund payments that were made within the past 90 days)
  `27` → payment not allowed: consumer or shop are currently not able to pay/be payed (this condition could be temporary)
  `172` → pre authorized payments token is not valid
- **message** [string]: Error message

## 401 Unauthorized

- **code** [integer]: Error code
  `34` → shop not found or unauthorized
- **message** [string]: Error message

## 403 Forbidden or invalid authorization header

- **code** [integer]: Error code
  `45` → unable to fulfil request
  `70` → anti-hammering violation
- **message** [string]: Error message

BODY PARAMS

**flow** string required
The flow of the payment (MATCH_CODE, MATCH_USER, REFUND or PRE_AUTHORIZED)

**amount_unit** int32 required
Amount of the payment in cents

**currency** string required
Currency of the payment (only EUR currently supported)

**pre_authorized_payments_token** string

Pre-Authorized token id (required with the PRE_AUTHORIZED flow only)

**parent_payment_uid** string

Unique ID of the payment to refund (required with the REFUND flow only)

**consumer_uid** string

Unique ID of the consumer that has to accept the payment. To retrieve the customer uid use the Retrive customer API (required with the MATCH_USER flow only)

**external_code** string

Order ID or payment external identifier (max length allowed is 50 chars)

**callback_url** string

The url that will be called with an http GET request when the payment changes state. When url is called a Get payment details can be called to know the new Payment status. Note that {uuid} will be replaced with the Payment ID

**redirect_url** string

The url to redirect the user after the payment flow is completed

**expiration_date** string

The expiration date of the payment

**metadata** object

Generic field that can be used to store generic info. The field `phone_number` can be used to pre-fill the mobile number.

HEADERS

**METADATA OBJECT** +

**Host** string required

The host declared in the signature

**Date** string required

The date declared in the signature

**Digest** string required

The digest declared in the signature

**Authorization** string required

Signature of the request

**Idempotency-Key** string

The idempotent token of the request

**x-satispay-deviceinfo** string
Info about the device

**x-satispay-devicetype** string
Device type: SMARTPHONE, TABLET, CASH-REGISTER, POS, PC or ECOMMERCE_PLUGIN

**x-satispay-os** string
Operative System name

**x-satispay-osv** string
Operative System version

**x-satispay-apph** string
Software house name

**x-satispay-appn** string
Software name

**x-satispay-appv** string
Software version

**x-satispay-tracking-code** string
Tracking code used by Satispay commercial partners

RESPONSES

**200**
200

**400**
400

**401**
401

**403**
403

LANGUAGE

cURL PHP

CURL - MATCH_CODE

REQUEST Examples

```
curl --request POST \
  --url https://authservices.satispay.com/g_business/v1/payments \
  --header 'Content-Type: application/json' \
  --header 'host: authservices.satispay.com' \
  --header 'date: Fri, 28 Aug 2020 09:18:17 GMT' \
  --header 'digest: SHA-256=...' \
  --header 'Authorization: Signature keyId="vefn1p...", algorithm="rsa-sha256", hea
  --data '{
    "flow": "MATCH_CODE",
    "amount_unit": 100,
    "currency": "EUR",
    "external_code": "my_order_id",
    "callback_url": "https://myServer.com/myCallbackUrl?payment_id={uuid}",
    "redirect_url": "https://myServer.com/myRedirectUrl",
    "metadata": {
      "order_id": "my_order_id",
      "user":"my_user_id",
      "payment_id":"my_payment",
      "session_id":"my_session",
      "key": "value"
    }
  }'
```

RESPONSE Examples

Choose an example:

200 - OK - MATCH CODE 200 - OK - MATCH USER 200 - OK - REFUND

200 - OK - PRE AUTHORIZED 400 - Result 401 - Result 403 - Result

Updated 6 days ago

← External code Get payment details →

# External code

Order ID or payment external identifier

The body param `external_code` is the field that can be use to either store the order_id (eCommerce) or the payment external identifier when Creating a payment.
This field is useful to perform the financial reconciliation from the Satispay Dashboard (it will be included in the payments export) and it is also displayed on the Satispay app for the end user.

Updated 9 months ago

# Get payment details

GET https://authservices.satispay.com/g_business/v1/payments/{id}

API to retrieve the detail of a specific payment

## Responses

## 200 OK

- **id** [string]: Unique ID of the payment.
- **code_identifier** [string]: Generated code identifier
- **type** [string]: Type of payment ( `TO_BUSINESS` or `REFUND_TO_BUSINESS` )
- **amount_unit** [number]: Amount of the payment in cents
- **currency** [string]: Currency of the payment
- **status** [string]: Status of the payment ( `PENDING` , `ACCEPTED` or `CANCELED` )
- **expired** [boolean]: If true, the payment is expired
- **metadata** [object]: Additional metadata of the payment
- **sender** [object]: The sender actor of the payment
  - **id** [string]: Unique ID of the sender
  - **type** [string]: Type of the actor ( `CONSUMER` )
  - **name** [string]: Short name of the actor
- **receiver** [object]: The receiver actor of the payment
  - **id** [string]: Unique ID of the receiver
  - **type** [string]: Type of the actor ( `SHOP` )
- **daily_closure** [object]: The daily closure of the payment
  - **id** [string]: ID of the daily closure
  - **date** [string]: The closure date
- **insert_date** [datetime]: Timestamp of payment insertion
- **expire_date** [datetime]: Timestamp of payment expiration
- **external_code** [string]: Order ID or payment external identifier

## 403 Forbidden or invalid authorization header

- **code** [integer]: Error code
- **message** [string]: Error message

# 404 Resource not found

- **code** [integer]: Error code
- **message** [string]: Error message

> **RETRY**
>
> If the API returns any of the possible error, like a 500, this doesn't mean that the payment might not be ACCEPTED already.
> If you get the error we suggest to execute a retry within few seconds and if still not getting a response execute an Update Payment with the CANCEL_OR_REFUND flow.

PATH PARAMS

**id** string required
The id of the payment to retrieve

HEADERS

**Host** string required
The host declared in the signature

**Date** string required
The date declared in the signature

**Digest** string required
The digest declared in the signature

**Authorization** string required
Signature of the request

**x-satispay-response-wait-time** string
Seconds that the call will be hanging, waiting for a payment status change.
Maximum value is 60 seconds.
0

RESPONSES

**200**
200

**403**
403

**404**
404

LANGUAGE

cURL PHP

DEFAULT

REQUEST

```
curl --request GET \
  --url 'https://authservices.satispay.com/g_business/v1/payments/2936affa-ab4c-4da
  --header 'host: authservices.satispay.com' \
  --header 'date: Fri, 28 Aug 2020 09:18:17 GMT' \
  --header 'digest: SHA-256=...' \
  --header 'Authorization: Signature keyId="vefn1p...", algorithm="rsa-sha256", hea
```

RESPONSE Examples

Choose an example:

200 - Result 403 - Result 404 - Not Found

Updated 12 months ago

← Create payment Get shop-payments list →

# Get shop-payments list

GET https://authservices.satispay.com/g_business/v1/payments

API to retrieve the list of payments for a specific shop. The shop is automatically filtered based on the KeyID used in the authorisation header.

## Payment pagination

Payments returned by this API are paginated with a default limit of 20 items. The `has_more` boolean field tells you if there are more payments than the limit used and you should use the `starting_after` parameters filled with the id of the last payment if you want to retrieve the next page of the list.

If you want to list all Payments for specific date you should:
1 - call the Get payment list using the parameter `starting_after_timestamp` and using the timestamp in milliseconds of the date you want
2 - save the last payment `id` returned at step 1
3 - call the Get payment list using the parameter `starting_after` and using the payment `id` you saved at the step 2
4 - save the last payment `id` returned at step 3
5 - repeat step 4 until you reach the last payment of the date you need

> 📘 `starting_after` and `starting_after_timestamp` parameters are cursors within the list of payments that is ordered by creation date from the newest and this means that payments returned with this filters are sequentially after, and not temporarily after, the id or the timestamp used.

## Responses

## 200 Ok

- **has_more** [boolean]: Are there more items in the list?
- **data** [array of objects]: The matching payments
  - **id** [string]: Unique ID of the payment
  - **type** [string]: Type of payment ( `TO_BUSINESS` or `REFUND_TO_BUSINESS` )
  - **amount_unit** [number]: Amount of the payment in cents

- **currency** [string]: Currency of the payment
- **status** [string]: Status of the payment ( `PENDING` , `ACCEPTED` or `CANCELED` )
- **status_ownership** [boolean]: If true, the device making the request is responsible for the final status reached by the payment
- **expired** [boolean]: If true, the payment is expired
- **metadata** [object]: Additional metadata of the payment
- **sender** [object]: The sender actor of the payment
  - **id** [string]: Unique ID of the sender
  - **type** [string]: Type of the actor ( `CONSUMER` )
  - **name** [string]: Short name of the actor
- **receiver** [object]: The receiver actor of the payment
  - **id** [string]: Unique ID of the receiver
  - **type** [string]: Type of the actor ( `SHOP` )
- **status_owner** [object]: The actor responsible of the payment final status
  - **id** [string]: Unique ID of the actor
  - **type**[string]: Type of the actor ( `DEVICE` )
- **daily_closure** [object]: The daily closure of the payment
  - **id** [string]: ID of the daily closure
  - **date** [string]: The closure date
- **insert_date** [datetime]: Timestamp of payment insertion
- **expire_date** [datetime]: Timestamp of payment expiration
- **external_code** [string]: Order ID or payment external identifier

## 400 Bad request

- **code** [integer]: Error code
  `36` → invalid query parament status
- **message** [string]: Error message

## 401 Unauthorized

- **code** [integer]: Error code
  `34` → shop not found or unauthorized
- **message** [string]: Error message

## 403 Forbidden or invalid authorization header

- **code** [integer]: Error code
  `45` → unable to fulfil request
- **message** [string]: Error message

QUERY PARAMS

**status** string
Filter by the payment status ACCEPTED, PENDING or CANCELED
ACCEPTED

**limit** int32
A limit on the number of objects to be returned, between 1 and 100
20

**starting_after** string
Is the id that defines your place in the list when you make a payment list request

**starting_after_timestamp** string
Is the timestamp (in milliseconds) that defines your place in the list when you make a payment list request

HEADERS

**Host** string required
The host declared in the signature

**Date** string required
The date declared in the signature

**Digest** string required
The digest declared in the signature

**Authorization** string required
Signature of the request

**x-satispay-deviceinfo** string
Info about the device

**x-satispay-os** string
Operative System name

**x-satispay-devicetype** string
Device type: SMARTPHONE, TABLET, CASH-REGISTER, POS, PC or ECOMMERCE_PLUGIN

**x-satispay-osv** string
Operative System version

**x-satispay-apph** string
Software house name

**x-satispay-appn** string
Software name

**x-satispay-appv** string

Software version

**x-satispay-tracking-code** string

Tracking code used by Satispay commercial partners

RESPONSES

**200**
200

**400**
400

**401**
401

**403**
403

LANGUAGE

cURL PHP

DEFAULT

REQUEST

```
curl --request GET \
  --url 'https://authservices.satispay.com/g_business/v1/payments?status=ACCEPTED'
  --header 'Authorization: Signature keyId="vefn1p...", algorithm="rsa-sha256", hea
  --header 'host: authservices.satispay.com' \
  --header 'date: Fri, 28 Aug 2020 09:18:17 GMT' \
  --header 'digest: SHA-256=...' \
  --header 'x-satispay-devicetype: CASH-REGISTER' \
  --header 'x-satispay-deviceinfo: The Box III' \
  --header 'x-satispay-os: Hooli OS' \
  --header 'x-satispay-osv: SP1' \
  --header 'x-satispay-apph: Hooli' \
  --header 'x-satispay-appn: Xyz' \
  --header 'x-satispay-appv: v1' \
  --header 'x-satispay-tracking-code: MYCODE'
```

RESPONSE Examples

Choose an example:

200 - Result 400 - Result 401 - Result 403 - Result

Updated over 1 year ago

← Get payment details

Update payment →

# Retrieve daily closure

GET https://authservices.satispay.com/g_business/v1/daily_closure/{daily_closure_date}

API to retrieve shop daily closure

## Responses

## 200 OK

- **shop_daily_closure** [object]: The daily closure of the shop
  - **id** [string]: Unique ID of the daily closure
  - **type** [string]: Type of the daily closure ( `SHOP` )
  - **customer_uid** [string]: Unique ID of the shop
  - **amount_unit** [number]: The daily closure amount of the whole shop
  - **currency** [string]: The currency of the daily closure
- **pdf** [object]: The receipt of the daily closure
  - **url** [string]: The pre-signed url to the daily closure pdf
  - **expiration** [date]: The expiration date of the pre-signed url
  - **expire_in_sec** [integer]: The time to live of the pre-signed url in seconds
  - **bucket** [string]: The bucket in which the pdf is stored
  - **key** [string]: The key of the pdf
  - **http_method** [string]: The http method that can be invoked on pre-signed url

## 400 Bad daily closure date format

- **code** [integer]: Error code
- **message** [string]: Error message

## 403 Forbidden or invalid authorization header

- **code** [integer]: Error code
- **message** [string]: Error message

> 📘 Notes
>
> - Default offset is at 00:00 (midnight), unless a custom one has been requested

- If daily closure is called before the end of the day, it will return the payments till that moment
- Refunds impact the daily closure of the day when they have been executed

PATH PARAMS

**daily_closure_date** string required
The day on which retrieve the daily closure (format yyyyMMdd)

QUERY PARAMS

**generate_pdf** boolean
Generate the pdf with the daily closure amounts

HEADERS

**Host** string required
The host declared in the signature

**Date** string required
The date declared in the signature

**Digest** string required
The digest declared in the signature

**Authorization** string required
Signature of the request

RESPONSES

**200**
200

**400**
400

**403**
403

LANGUAGE

cURL

DEFAULT

REQUEST

```
curl --request GET \
  --url 'https://authservices.satispay.com/g_business/v1/daily_closure/20190617?ger
  --header 'host: authservices.satispay.com' \
  --header 'date: Fri, 28 Aug 2020 09:18:17 GMT' \
  --header 'digest: SHA-256=...' \
  --header 'Authorization: Signature keyId="vefn1p...", algorithm="rsa-sha256", hea
```

RESPONSE Examples

Choose an example:

200 - Result 400 - Result 403 - Result

Updated almost 2 years ago

← Update payment Create authorization →