PPG

Proxy Payment Gateway

Table of Contents

Chapter 1. Overview

PPG is the Proxy Payment Gateway of JIBit Corporation.
It lets the users of 3rd-party clients (merchants, banks, organizations, etc.) purchase goodies from them using PPG as a transparent payment gateway.
In the back end, we will redirect the user to a PSP payment gateway (for example, Saman IPG) to pay the purchase price.

Using the APIs of this application, a client can create and initiate purchase orders and track the state of these purchases.

You can always download the latest REST API documentation of PPG (version 3) in HTML and PDF and Swagger (Open API) format.

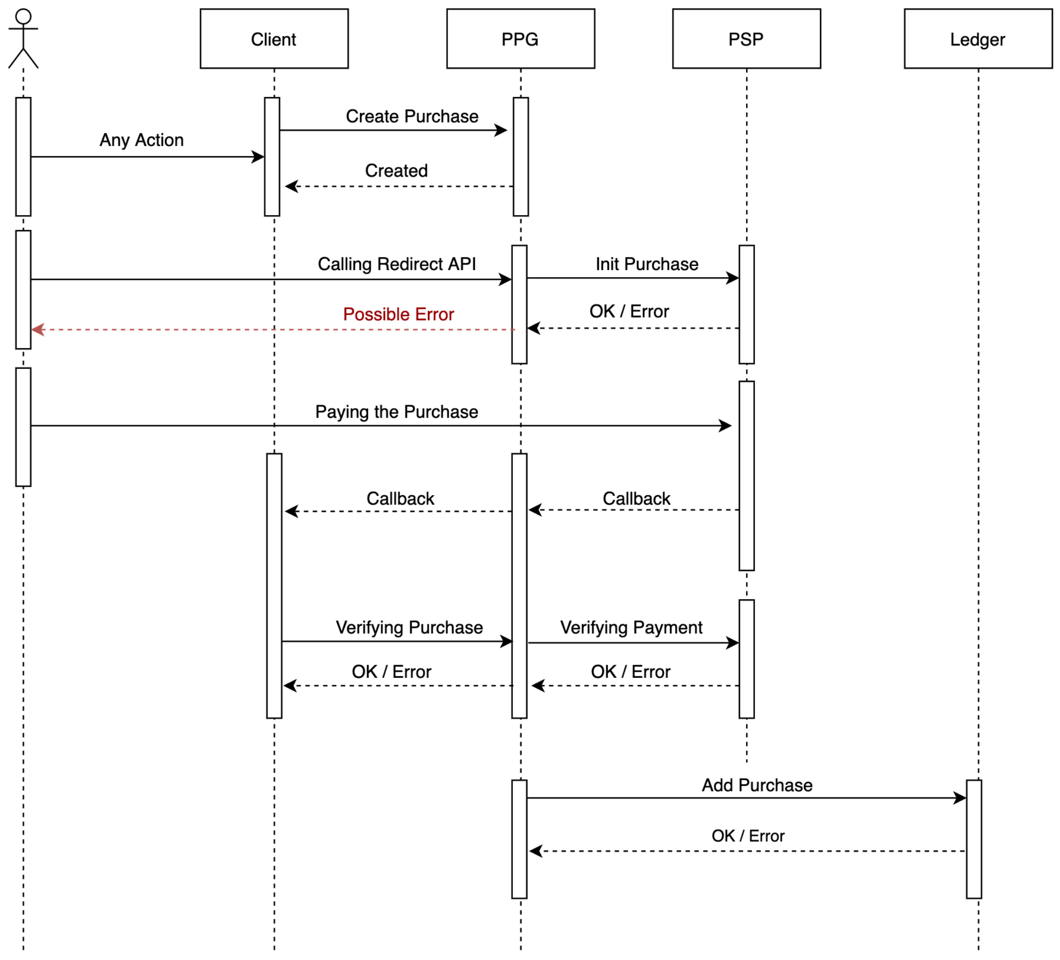
Using Swagger Online Editor, you can directly call the PPG APIs and auto-generate an HTTP client for PPG APIs in your preferred programming language.

The base URL of PPG is:

`https://napi.jibit.ir/ppg`

1.1. Purchase Flow

The sequence diagram of the purchase flow is as follows:



It boils down to the following steps:

1. An authorized client creates a purchase.
   If the purchase remains CREATED for a long time (15 minutes), we will expire and close the purchase (`EXPIRED` state).
   Note: To obtain an access token, use the Generate Token API.
   To create a purchase, use the Create Purchase API.
2. We create the requested purchase and return a few details, including the *Payment URL* (`pspSwitchingUrl`).
3. The client's user calls that *Payment URL*.
4. We select a PSP terminal for the user and initiate a payment process in the selected PSP.
5. If the initiation process succeeds, we will redirect the user to the PSP page.
   If the initiation process has failed for any reason, we may fail and close the purchase after retrying with other possible PSP terminals (`FAILED` state).
6. In case of a successful initiation process in PSP, we expect the PSP to call our *Callback* to inform us of the payment process.

If the user does not complete the payment in less than 15 minutes, we may close the purchase and change its state to `EXPIRED`.

7. If the PSP redirects the user to our callback, we would redirect the user to the client callback with the same status as the PSP's status.
   Possible statuses are `SUCCESSFUL` and `FAILED` (and `UNKNOWN` for auto-verify enabled terminals).
   See Redirecting The User to The Client Callback.
8. If the payment was successful, the client should verify the purchase completion using the Verify Purchase API (`READY_TO_VERIFY` state).
   If the client does not verify the purchase, we will expire the purchase after 15 minutes(`EXPIRED` state).
9. If the client verifies us, we will verify the PSP payment and return the verification result to the client.
   If, for whatever reason, the PSP verification result is unknown (`UNKNOWN` state), the client should use the corresponding inquiry endpoint to identify the actual payment verification state.
   Note: Use the Filter Purchases API to inquiry the purchases.
10. In case of user payment fraud, we would reverse the payment (`REVERSED` state).
    Possible frauds are changing the purchase amount or paying with a different card than the forced one.
    After reversing the purchase, its amount will return to the user's bank account in minutes or hours.
11. Finally, we would insert the purchase in our ledger, and the purchase state will be `SUCCESS`.

Purchase possible states are: `IN_PROGRESS`, `READY_TO_VERIFY`,`EXPIRED`, `FAILED`, `REVERSED`, `SUCCESS`, and `UNKNOWN`.

Note: After creating the purchase, if it is not verified (completed) within 15 minutes, it will be expired.

1.2. Redirecting The User to The Client Callback

After that, PSP redirects the user to our callback (Step 7); we would redirect the user to the client callback with the `POST` method and `Content-Type: application/x-www-form-urlencoded` in the header.

The passed parameters are:

- `amount`: The amount of payment
- `wage`: Represents the wage of purchase.
- `currency`: The currency of payment.
  Currently, its only value is `IRR`.
- `purchaseId`: The purchase Id generated on our side that uniquely identifies the purchase.
- `clientReferenceNumber`: The reference number provided by the client for the purchase.
  It may have special characters and therefore is URL-encoded.
- `status`: The status of PSP payment.
  Possible values are `SUCCESSFUL`, `FAILED`, and `UNKNOWN`.

UNKNOWN status would only happen to terminals with auto-verify feature enabled.
In case of SUCCESSFUL and UNKNOWN statuses, client is responsible to call the Verify API to verify purchase.
In UNKNOWN status not resolved even after verifying the purchase, you need to inquiry purchase periodically until the purchase status changes to a final state (FAILED,
SUCCESS, REVERSED, or EXPIRED)

- payerIp: The payer IP.
- pspReferenceNumber: The PSP reference number.
It may have special characters and therefore is URL-encoded.
If the payment fails, we will not send this.
- pspRRN: The PSP RRN.
If the payment fails, we will not send this.
- payerMaskedCardNumber: The payer’s masked card number.
If the payment fails, we will not send this.
- pspName: The PSP Name like 'saman-ipg', 'sepehr-ipg' or 'ap-ipg'.
- pspHashedCardNumber: The hashed bank card number used for the transaction.
PSP provides it, and all its characters are upper case.
Example: F62A0955E51BC46D71B3647594913594
If the payment fails, we will not send this.
- failReason: The purchase failure reason in case of any failure.
Possible values are: CANCELLED_BY_USER, TRANSACTION_TIMED_OUT, PSP_PAYER_CARD_NOT_MATCHED, UNKNOWN.
If the fail reason is UNKNOWN, you can inquire the purchase later to find the exact fail reason.
If the payment succeeds, we will not send this.

Example Callback Request:

```
POST {clientCallbackPath} HTTP/1.1
Content-Type: application/x-www-form-urlencoded
Host: {clientCallbackHost}

amount=500000&wage=4000&currency=IRR&purchaseId=34765321083648336&clientReferenceNumber
=client-ref-
num&status=SUCCESSFUL&payerIp=23.155.43.100&pspReferenceNumber=GmshtyjwKSsd%2Fd54Idy8C
OJK78gDjse2BjPw%2B3dlFj&payerMaskedCardNumber=504172******7412&pspRRN=17356154785&pspN
ame=saman-ipg&pspHashedCardNumber=F62A0955E51BC46D71B3647594913594
```

The client gives us the client callback URL at the purchase creation.
callbackUrl == {clientCallbackHost} + {clientCallbackPath}

For example, if the client’s callbackUrl is https://api.client.ir/transaction/1234567/callback, then:

- {clientCallbackHost} == https://api.client.ir
- {clientCallbackPath} == /transaction/1234567/callback

1.3. JWT Access Tokens

The client must have a token to access the authorized APIs on PPG.

The access token expiration duration is 24 hours.
The refresh token has a 2x expiration duration, which means it will expire in 48 hours.

The client is responsible for generating a new pair of access/refresh tokens using the refresh token every day.

For generating the pair of access/refresh tokens on the first day, the client must use the API/secret keys.
The API/secret keys are accessible on Dashboard Panel.

Note: Please don't use the API/secret keys to generate a new pair of access/refresh tokens in subsequent days.
Please consider this for the sake of your security.

The client should set the JWT token on the header of the requests that want to access the secured APIs in this format:

```
Authorization: Bearer {client.jwt.accessToken}
```

To get a new pair of access/refresh tokens using API/secret keys, use the Generate Token API.
To get a new pair of access/refresh tokens using the current refresh token, use the Refresh Token API.

1.4. Client-Trusted IPs

The client could introduce a set of trusted IPs to access the APIs on PPG.
After that, the client must call the APIs from one of those trusted IPs.
This feature is optional, and if the client doesn't set any trusted IP, API calls from all IPs are allowed.
The client owner should add the IPs on the Dashboard Panel to register the trusted IPs for calling APIs of PPG.

If the client calls the APIs from an untrusted IP, it will get the `ip.not_trusted` error.

Note: Only IPv4 addresses are supported. example: `5.117.199.255`

1.5. Error Handling Mechanism

In any unsuccessful endpoint call (4xx and 5xx HTTP status codes), the response is JSON content in the form of:

```
{
  "fingerprint": "79d4215f-deba-4f3c-8024-839448bcfaa0",
  "errors": [
    {
      "code": "token.generation.failed",
      "message": "Token generation failed."
    }
  ]
}
```

The fingerprint helps us to track the exact problem internally.

The error array contains a list of possible errors.
Each error has a code as a field, and you can use this code to understand the meaning of the error.

Sometimes you will see a message alongside the code.
This message is a human-readable reason for the error that occurred.

If you encountered `server.error` as code, tell us the fingerprint value to track the exact problem internally.

We listed the possible error codes and messages for every endpoint.

1.5.1. Error Explained: Invalid Or Missing Body

When the request body is not a valid JSON object, the `web.invalid_or_missing_body` as the error code will return.
Possible reasons are:

1. When a field value does not have the correct type.
   For example:
   - Assigning a primitive value to an object field.
     Example: `{ "additionalData": 1234, … }` => Correction: `{ "additionalData": { "count": 1234 }, … }`
   - Assigning an incorrect value for an enum field.
     Example: `{ "currency": "RIALS", … }` => Correction: `{ "currency": "IRR", … }`
   - Assigning a float number to an integer field.
     Example: `{ "amount": 1840000.6, … }` => Correction: `{ "amount": 1840000, … }`
2. When not opening or closing the brackets correctly.
   Example: `{ "amount": 120000, …` with missing closing bracket => Correction: `{ "amount": 120000, … }`
3. When using field names without " around them.
   Example: `{ amount: 120000, … }` => Correction: `{ "amount": 120000, … }`
4. When using string values without " around them.
   Example: `{ "payerMobileNumber": 09334280857, … }` => Correction: `{ "payerMobileNumber": "09334280857", … }`

1.6. Version information

Version : 3.1.143

1.7. URI scheme

Host : napi.jibit.ir
BasePath : /ppg
Schemes : HTTPS

1.8. Tags

- App API : Provides APIs to check the health status of the service.
- Client API : Provides the Client-related APIs.
- Purchase API : Provides APIs to manage purchase-related APIs.
- Refund API : Provides APIs to manage refund-related APIs.
- Settlement API : Provides APIs to manage settlement-related APIs.
- Token API : Provides APIs to manage tokens of the client. These APIs are permitted to be called without the authorization token.

1.9. Consumes

- `application/json`

1.10. Produces

- `application/json`

Chapter 2. Resources

2.1. App API

Provides APIs to check the health status of the service.

2.1.1. Check Health Status

```
GET /v3/app/health
```

Description

Checks the health status of the service.

Possible Error Codes:

- `ip.not_trusted`: When the client IP is not trusted. Read More.
- `client.not_active`: When the client is inactive.
- `security.auth_required`: The bearer JWT token is not in the request header as the `Authorization` parameter.
  Read More.
- `token.verification_failed`: When the access token is invalid or expired. Read More.
- `server.error`: Internal server error. Please provide the value of the fingerprint to help us track the exact problem internally.

Authorization: Bearer JWT token in the header. Read More.

Responses

HTTP Code	Description	Schema
200	UP if the service is OK. DOWN if not OK	enum (DOWN, UP)

Example HTTP request

```
GET /ppg/v3/app/health HTTP/1.1
Host: napi.jibit.ir
Authorization: Bearer {client.jwt.accessToken}
```

Example HTTP response

```
HTTP/1.1 200 OK
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Content-Type: application/json
Content-Length: 4

"UP"
```

Example Curl request

```
$ curl 'https://napi.jibit.ir/ppg/v3/app/health' -i -X GET \
    -H 'Authorization: Bearer {client.jwt.accessToken}'
```

2.2. Client API

Provides the Client-related APIs.

2.2.1. Get Client Balances

```
GET /v3/balances
```

Description

Retrieves the balances of the client for all its balance types.
Possible Error Codes:

- `client.not_active`: When the current client is not active.
- `ip.not_trusted`: When the client IP is not trusted. Read More.
- `security.auth_required`: The bearer JWT token is not present in request header as the `Authorization` parameter.
  Read More.
- `token.verification_failed`: The access token is invalid or expired. Read More.
- `server.error`: Internal server error. Please tell us the value of the fingerprint to track the exact problem internally.

Authorization: Bearer JWT token in header. Read More.

Responses

HTTP Code	Description	Schema
200	The balances of the client.	ClientBalances

Example HTTP request

```
GET /ppg/v3/balances HTTP/1.1
Host: napi.jibit.ir
Authorization: Bearer {client.jwt.accessToken}
```

Example HTTP response

```
HTTP/1.1 200 OK
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Content-Type: application/json
Content-Length: 416

{
  "balances" : [ {
    "balanceType" : "WLT",
    "amount" : 10000,
    "currency" : "IRR"
  }, {
    "balanceType" : "STL",
    "amount" : 10000,
    "currency" : "IRR"
  }, {
    "balanceType" : "FEE",
    "amount" : 10000,
    "currency" : "IRR"
  }, {
    "balanceType" : "SHW",
    "amount" : 10000,
    "currency" : "IRR"
  }, {
    "balanceType" : "BLK",
    "amount" : 10000,
    "currency" : "IRR"
  } ]
}
```

Example Curl request

```
$ curl 'https://napi.jibit.ir/ppg/v3/balances' -i -X GET \
    -H 'Authorization: Bearer {client.jwt.accessToken}'
```

2.3. Purchase API

Provides APIs to manage purchase-related APIs.

2.3.1. Create Purchase

```
POST /v3/purchases
```

Description

Creates a purchase request.

After creating the purchase, the client must redirect its user to the provided URL (`pspSwitchingUrl`).

Once the purchase is initialized, the user will be redirected to a PSP payment gateway. See also Redirecting The User to The Client Callback

Note 1: After the user completes the payment on the PSP page, the client should verify the purchase.

Note 2: If the purchase is not verified (completed) within 15 minutes of creation, it will expire.

Possible Error Codes:

- `client.not_active`: When the client is inactive.
- `amount.is_required`: When the amount is empty.
- `amount.not_enough`: When the amount is invalid. The minimum amount is 5000 Rials.
- `wage.is_invalid`: When the wage is invalid. Its minimum value is 0.
- `fee_as_wage.wage_must_be_zero`: When the *Setting the Fee as Wage* feature is enabled for the client, and the `wage` parameter has non-zero value.
- `amount_plus_wage.permitted_value_exceeded`: When amount plus wage exceeds the maximum permitted value for your client. The default max amount is 2_000_000_000 IRR.
- `wage.must_be_less_than_fifteen_percent_of_purchase_amount`: when wage exceeds 15% of the purchase amount.
- `currency.is_required`: When the currency is empty.
- `callbackUrl.is_required`: When callbackUrl is empty.
- `callbackUrl.is_invalid`: When callbackUrl is invalid.
- `callbackUrl.max_length`: When callbackUrl exceeds its allowed length. Its maximum length is 1024 characters.
- `callbackUrl.domain_not_in_whitelist`: Domain of callback URL is not in whitelist.
- `clientReferenceNumber.is_required`: When clientReferenceNumber is empty.
- `clientReferenceNumber.duplicated`: When clientReferenceNumber for the client is not unique.
- `payerCardNumber.is_invalid`: When payerCardNumber is invalid.

- `payerNationalCode.is_invalid`: When payerNationalCode is invalid.
- `userIdentifier.max_length`: When userIdentifier exceeds its allowed length. Its maximum length is 50 characters.
- `payerMobileNumber.is_invalid`: When payerMobileNumber is invalid.
- `payerMobileNumber.in_blacklist`: When the payer mobile number is in the blacklist.
- `payerCardNumber_and_payerCardNumbers.just_one_of_them_is_permitted`: Just one of these parameters is permitted: `payerCardNumber`, `payerCardNumbers`
- `description.max_length`: When the description exceeds its allowed length. Its maximum length is 256 characters.
- `ip.not_trusted`: When the client IP is not trusted. Read More.
- `security.auth_required`: The bearer JWT token is not in the request header as the `Authorization` parameter. Read More.
- `token.verification_failed`: The access token is invalid or expired. Read More.
- `web.invalid_or_missing_body`: When the request body is not a valid JSON. Read More.
- `server.error`: Internal server error. Please provide the value of the fingerprint to help us track the exact problem internally.

Authorization: Bearer JWT token in the header. Read More.

Parameters

Type	Name	Description	Schema
Body	**dto** *required*	Encapsulates the information required to create a purchase.	CreatePurchaseDto

Responses

HTTP Code	Description	Schema
200	Encapsulates the details of the created purchase including the PSP switching URL.	PurchaseCreation Result

Example HTTP request

```
POST /ppg/v3/purchases HTTP/1.1
Content-Type: application/json
Content-Length: 500
Host: napi.jibit.ir
Authorization: Bearer {client.jwt.accessToken}

{
  "amount" : 499900000,
  "wage" : null,
  "currency" : "IRR",
  "callbackUrl" : "https://www.client.ir/purchases/123456789/callback",
  "clientReferenceNumber" : "required-client-ref-num",
  "userIdentifier" : "a.pourtaghi",
  "payerNationalCode" : "2234567890",
  "payerCardNumber" : "6037997122223333",
  "payerCardNumbers" : [ "6037997122223333" ],
  "payerMobileNumber" : "09123454321",
  "additionalData" : {
    "someTag" : "some-value"
  },
  "description" : "optional client description"
}
```

Example HTTP response

```
HTTP/1.1 200 OK
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Content-Type: application/json
Content-Length: 225

{
  "purchaseId" : 1345628234,
  "purchaseIdStr" : "1345628234",
  "clientReferenceNumber" : "required-client-ref-num",
  "pspSwitchingUrl" : "https://napi.jibit.ir/ppg/v3/purchases/1345628234/payments",
  "currency" : null
}
```

Example Curl request

```
$ curl 'https://napi.jibit.ir/ppg/v3/purchases' -i -X POST \
    -H 'Content-Type: application/json' \
    -H 'Authorization: Bearer {client.jwt.accessToken}' \
    -d '{
  "amount" : 499900000,
  "wage" : null,
  "currency" : "IRR",
  "callbackUrl" : "https://www.client.ir/purchases/123456789/callback",
  "clientReferenceNumber" : "required-client-ref-num",
  "userIdentifier" : "a.pourtaghi",
  "payerNationalCode" : "2234567890",
  "payerCardNumber" : "6037997122223333",
  "payerCardNumbers" : [ "6037997122223333" ],
  "payerMobileNumber" : "09123454321",
  "additionalData" : {
    "someTag" : "some-value"
  },
  "description" : "optional client description"
}'
```

2.3.2. Filter Purchases

```
GET /v3/purchases
```

Description

Filters the purchases with provided criteria and page request.
All criteria parameter values should be URL-Encoded.
Possible Error Codes:

- `page_number.max_exceeded`: The page number is exceeded its allowed value.
- `page_size.max_exceeded`: The page size is exceeded its allowed size.
- `client.not_active`: When the client is not active.
- `ip.not_trusted`: When the client IP is not trusted. Read More.
- `security.auth_required`: The bearer JWT token is not present in request header as the `Authorization` parameter.
  Read More.
- `token.verification_failed`: The access token is invalid or expired. Read More.
- `server.error`: Internal server error. Please tell us the value of fingerprint to be able to track the exact problem internally.

Authorization: Bearer JWT token in header. Read More.

Parameters

Type	Name	Description	Schema
Query	**clientReferenceNumber** *optional*	Finds a purchase by its client reference number. The `null` means no filtering at all. Example: `T1233`	string
Query	**from** *optional*	Filters purchases from their creation date inclusively. The `null` means no filtering at all. Example: `2021-04-03T13:21:25Z`	string (date-time)
Query	**page** *optional*	Represents the page number; One indexed. The `null` means the first page. The max page number is 20. Example: `1`	integer (int32)
Query	**pspReferenceNumber** *optional*	Filters purchases by their PSP reference number. The `null` means no filtering at all. Example: `"sfGD93tgGQ153Jr2Qm3"`	string
Query	**pspRrn** *optional*	Filters purchases by their PSP RRN. The `null` means no filtering at all. Example: `"385739230986"`	string
Query	**pspTraceNumber** *optional*	Filters purchases by their PSP trace number. The `null` means no filtering at all. Example: `"43083947502"`	string
Query	**purchaseId** *optional*	Finds a purchase by its identifier. Can be used to find an exact purchase. The `null` means no filtering at all. Example: `1234455`	integer (int64)
Query	**size** *optional*	Represents the page size. The `null` means to use default size. The default page size is 25. The max page size is 250. Example: `10`	integer (int32)
Query	**status** *optional*	Filters purchases based on their state. The `null` means no filtering at all. Example: `SUCCESS`	enum (EXPIRED, FAILED, IN_PROGRESS, MANUALLY_SUCCESS, READY_TO_VERIFY, REVERSED, SUCCESS, UNKNOWN)
Query	**to** *optional*	Filters purchases to their creation date exclusively. The `null` means no filtering at all. Example: `2021-05-03T13:21:25Z`	string (date-time)

Type	Name	Description	Schema
Query	**userIdentifier** *optional*	Filters purchases based on their user identifier. The `null` means no filtering at all. Example: `103243`	string

Responses

HTTP Code	Description	Schema
200	The list of paginated purchases.	Paginated«DetailedPurchaseDto»

Example HTTP request

```
GET /ppg/v3/purchases?status=FAILED&userIdentifier=09334565674&page=1&size=20 HTTP/1.1
Host: napi.jibit.ir
Authorization: Bearer {client.jwt.accessToken}
```

Example HTTP response

```
HTTP/1.1 200 OK
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Content-Type: application/json
Content-Length: 1912

{
  "pageNumber" : 1,
  "size" : 20,
  "numberOfElements" : 1,
  "hasNext" : false,
  "hasPrevious" : false,
  "elements" : [ {
    "purchaseId" : 1200,
    "purchaseIdStr" : "1200",
    "amount" : 100000,
    "wage" : 5000,
    "fee" : 4555,
    "feePaymentType" : "POST_PAID",
    "shaparakFee" : 1200,
    "resellerCode" : "test-reseller",
    "affiliateFee" : 50000,
    "netAmount" : 103800,
    "currency" : "IRR",
    "callbackUrl" : "https://www.client.ir/purchases/123456789/callback",
    "state" : "READY_TO_VERIFY",
    "clientReferenceNumber" : "client-ref-num",
    "pspName" : "saman-ipg",
    "pspRrn" : "84a29cbb205c4aa18dc71b8e3cb99639",
    "pspReferenceNumber" : "GmshtyjwKSsoD4uz5cpmzY5aBaWxx21mxAYRo92Gx0",
    "pspTraceNumber" : "psp-trace-num",
    "expirationDate" : "2024-11-13T02:40:18.144699925Z",
    "userIdentifier" : "a.pourtaghi",
    "payerMobileNumber" : "09132517905",
    "payerCardNumber" : "434567-3452",
    "payerNationalCode" : "4060145796",
    "description" : "client custom description",
    "additionalData" : {
      "someTag" : "some-value"
    },
    "pspMaskedCardNumber" : "434567****3452",
    "pspHashedCardNumber" : "538B3C64A94DEE355DA7DB7BF8A7D50F",
    "pspCardOwner" : "Ali Khaksari",
    "pspFailReason" : "CARD_IS_LIMITED",
    "pspFailReasons" : [ {
      "code" : 62,
      "description" : "□□□□ □□□□□ □□□□□ □□□ □□□",
      "pspError" : "CARD_IS_LIMITED"
    } ],
    "initPayerIp" : "43.23.6.28",
    "redirectPayerIp" : "45.73.6.28",
    "pspSettled" : false,
    "refunded" : null,
    "refundInquiryResult" : null,
    "refundableAmount" : 105000,
    "createdAt" : "2024-11-13T02:31:58.144595452Z",
    "billingDate" : "2024-11-13T02:35:18.144701172Z",
    "verifiedAt" : "2024-11-13T02:34:48.144697438Z",
    "pspSettledAt" : "2024-11-13T02:41:58.144698858Z",
    "settlementId" : null,
    "hasContradiction" : true
  } ]
}
```

Example Curl request

```
$ curl
'https://napi.jibit.ir/ppg/v3/purchases?status=FAILED&userIdentifier=09334565674&page=
1&size=20' -i -X GET \
    -H 'Authorization: Bearer {client.jwt.accessToken}'
```

2.3.3. Filter Purchase Histories

```
GET /v3/purchases/histories
```

Description

Filters purchases log history with provided criteria and page request.
For faster responses, please select smaller date ranges, for example, one-hour range
and also use `nextPageId` instead of `page` in the request.
All criteria parameter values should be URL-Encoded.

This feature is not enabled by default and should be enabled by admin manually.

Possible Error Codes:

- `from.is_required`: The from date is not provided as query param value.
- `to.is_required`: The to date is not provided as query param value.
- `from_and_to.difference_should_be_less_than_6_hours`: the difference between provided from value and to value is more than 6 hours.
- `page_number.max_exceeded`: The page number is exceeded its allowed value.
- `page_size.max_exceeded`: The page size is exceeded its allowed size.
- `client.not_active`: When the client is not active.
- `ip.not_trusted`: When the client IP is not trusted. Xref:client_trusted_ips[Read More.]
- `security.auth_required`: The bearer JWT token is not present in request header as the `Authorization` parameter.
  Xref:jwt_access_tokens[Read More.]
- `token.verification_failed`: The access token is invalid or expired. Xref:jwt_access_tokens[Read More.]
- `server.error`: Internal server error. Please tell us the value of fingerprint to be able to track the exact problem internally.

Authorization: Bearer JWT token in header. Xref:jwt_access_tokens[Read More.]

Parameters

Type	Name	Description	Schema
Query	**from** *required*	The start of the date range (inclusive) for querying purchase history. Example: `2024-10-15T13:00:00Z`	string (date-time)
Query	**nextPageId** *optional*	The identifier for fetching the next page of results. Using `nextPageId` allows faster pagination than a page number parameter. If provided, the `page` parameter will be ignored.	integer (int64)

Type	Name	Description	Schema
Query	**page** *optional*	The page number to fetch (1-indexed). If null, it means the first page. Note: The max page number is 10. Use `nextPageId` for faster results. If `nextPageId` is provided, this field is ignored.	integer (int32)
Query	**size** *optional*	The number of results per page. If null, the default size will be used. The default page size is 10,000, and the maximum allowed size is 20,000. Example: `1500`	integer (int32)
Query	**to** *required*	The end of the date range (exclusive) for querying purchase history. Example: `2024-10-15T14:00:00Z`	string (date-time)

Responses

HTTP Code	Description	Schema
200	The list of paginated purchase histories. Note that in some cases, the returned `totalCount` value may be null that means calculating this value was costly, and for faster response was not calculated. The `hasNext` field still works correctly and has no dependency on `totalCount` field, and the client can rely on that field for fetching next page.	IdPaginated«DetailedPurchaseHistoryDto»

2.3.4. Reverse Purchase

```
POST /v3/purchases/reverse
```

Description

Reverse a purchase.

This feature is not enabled by default and should be enabled by admin manually.

Possible Error Codes:

- `reverse.is_not_enabled`: When this feature is not enabled for the client.
- `purchase.not_found`: When the purchase is not found.
- `reverse.not_supported`: When the refund is not supported for this type of purchase.
- `purchase.invalid_state`: When the purchase is not in valid (FINISHED) state.
- `clientReferenceNumber_or_purchaseId.are_required`: When clientReferenceNumber and

purchaseId are not provided.

- purchase.not_reversible: When the purchase is not reversible because of internal verification procedures.
- purchase.refunded: When the refund API has called for this purchase; so you can't reverse it.
- purchase.already_reversed: When the purchase is already reversed.
- ip.not_trusted: When the client IP is not trusted. Read More.
- security.auth_required: The bearer JWT token is not present in request header as the Authorization parameter.
  Read More.
- token.verification_failed: The access token is invalid or expired. Read More.
- web.invalid_or_missing_body: When the request body is not a valid JSON. Read More.
- server.error: Internal server error. Please tell us the value of fingerprint to be able to track the exact problem internally.

Authorization: Bearer JWT token in header. Read More.

Parameters

Type	Name	Description	Schema
Body	**dto** *required*	Encapsulated the information required to reverse a purchase.	ReversePurchaseDto

Responses

HTTP Code	Description	Schema
200	Encapsulates the purchase reverse result. It indicates whether the purchase reverse was successful, failed, already reversed or unknown.	ReverseResultDto

2.3.5. Verify Purchase

```
POST /v3/purchases/{purchaseId}/verify
```

Description

Verifies a purchase by the client.
The client should call this API after we redirect the user to the client's callback URL.

Only purchases with the state READY_TO_VERIFY can be verified.

If the client does not call this API, the PSP will automatically reverse the payment after a certain period of time, and the purchase state will be set to EXPIRED.

Note 1: This API does not have a request body and can be called using both `GET` and `POST` methods.

Note 2: For using Auto-verify feature consult our admins. Note that using this feature does not guaranty that purchase will be verified 100%,
but it can decrease the number of `EXPIRED` purchases due to not verified by client significantly.
So a client still must call the verification API with this feature enabled, but in most cases the API will return `payment.already_verified` error.

Possible Error Codes:

- `ip.not_trusted`: When the client IP is not trusted. Read More.
- `client.not_active`: When the client is inactive.
- `purchase.not_found`: When the purchase does not found.
- `payment.already_verified`: When the purchase is already verified.
- `purchase.invalid_state`: When the purchase's state is invalid.
- `purchase.already_reversed`: When the purchase is already reversed.
- `security.auth_required`: The bearer JWT token is not in the request header as the `Authorization` parameter.
  Read More.
- `token.verification_failed`: When the access token is invalid or expired. Read More.
- `server.error`: Internal server error. Please provide the value of the fingerprint to help us track the exact problem internally.

Authorization: Bearer JWT token in the header. Read More.

Parameters

Type	Name	Description	Schema
Path	**purchaseId** *required*	Represents the purchase Identifier.	integer (int64)

Responses

HTTP Code	Description	Schema
200	Encapsulates the purchase verification result. It indicates whether the purchase verification was successful, failed, already verified, reversed or unknown.	VerificationResultDto

2.4. Refund API

Provides APIs to manage refund-related APIs.

2.4.1. Refund Purchase

```
POST /v3/purchases/refund
```

Description

Refunds a purchase.

Possible Error Codes:

- `clientReferenceNumber_or_purchaseId.are_required`: When clientReferenceNumber and purchaseId are not provided.
- `purchase.not_found`: When the purchase is not found.
- `purchase.invalid_state`: When the purchase is not in valid (SUCCESS) state.
- `refund.not_supported`: When the refund is not supported for this type of purchase.
- `batchID.invalid_length`
- `submissionMode.not_valid`
- `transfers.invalid_length`
- `transfers.0.transferID.invalid_length`
- `transfers.0.destination.not_valid`
- `transfers.0.source_bank.not_supported`
- `transfers.0.destinationFirstName.invalid_length`
- `transfers.0.destinationLastName.invalid_length`
- `transfers.0.amount.not_enough_for_ach`
- `transfers.0.amount.exceeded_maximum_for_ach`
- `transfers.0.amount.not_enough_for_normal`
- `transfers.0.amount.exceeded_maximum_for_normal`
- `transfers.0.currency.not_valid`
- `transfers.0.description.invalid_length`
- `transfers.0.metadata.invalid_length`
- `transfers.0.notifyURL.invalid_length`
- `transfers.0.notifyURL.not_valid`
- `transfers.0.paymentID.invalid_length`
- `transfers.0.paymentID.not_valid`
- `payment_id.not_enabled`
- `balances.not_enough`
- `daily_limitation.reached`

- `transfer.already_exists`
- `ip.not_trusted`: When the client IP is not trusted. Read More.
- `security.auth_required`: The bearer JWT token is not present in request header as the `Authorization` parameter.
  Read More.
- `token.verification_failed`: The access token is invalid or expired. Read More.
- `web.invalid_or_missing_body`: When the request body is not a valid JSON. Read More.
- `server.error`: Internal server error. Please tell us the value of fingerprint to be able to track the exact problem internally.

Authorization: Bearer JWT token in header. Read More.

Parameters

Type	Name	Description	Schema
Body	**dto** *required*	Encapsulated the information required to refund a purchase.	RefundPurchaseDto

Responses

HTTP Code	Description	Schema
200	Encapsulates the response information of purchase refund from transferor.	TransferorResult

Example HTTP request

```
POST /ppg/v3/purchases/refund HTTP/1.1
Content-Type: application/json
Content-Length: 120
Host: napi.jibit.ir
Authorization: Bearer {client.jwt.accessToken}

{
  "clientReferenceNumber" : "client-ref-num-222",
  "purchaseId" : 1234,
  "amount" : 490000,
  "cancellable" : true
}
```

Example HTTP response

```
HTTP/1.1 200 OK
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Content-Type: application/json
Content-Length: 120

{
  "refundId" : 1234,
  "partialRefundIndex" : 1,
  "batchId" : "REFUND-BATCH-1234",
  "transferId" : "REFUND-1234-1"
}
```

Example Curl request

```
$ curl 'https://napi.jibit.ir/ppg/v3/purchases/refund' -i -X POST \
    -H 'Content-Type: application/json' \
    -H 'Authorization: Bearer {client.jwt.accessToken}' \
    -d '{
  "clientReferenceNumber" : "client-ref-num-222",
  "purchaseId" : 1234,
  "amount" : 490000,
  "cancellable" : true
}'
```

2.4.2. Inquiry Refund

```
GET /v3/purchases/refunds/{refundId}
```

Description

Inquiries a Refund.
refundId is purchaseId.

Possible Error Codes:

- `transfer.not_found`: When the refundId is not found.
- `ip.not_trusted`: When the client IP is not trusted. Read More.
- `security.auth_required`: The bearer JWT token is not present in request header as the `Authorization` parameter.
  Read More.
- `token.verification_failed`: The access token is invalid or expired. Read More.
- `web.invalid_or_missing_body`: When the request body is not a valid JSON. Read More.

- `server.error`: Internal server error. Please tell us the value of fingerprint to be able to track the exact problem internally.

Authorization: Bearer JWT token in header. Read More.

Parameters

Type	Name	Description	Schema
Path	**refundId** *required*	refundId	integer (int64)

Responses

HTTP Code	Description	Schema
200	Encapsulates the details of transferor refund inquiry.	TransferorInquiry Result

2.4.3. Cancel Refund

```
POST /v3/purchases/refunds/{refundId}/cancel
```

Description

Cancels a cancelling refund.

Possible Error Codes:

- `transfer.not_found`: When the refundId is not found.
- `cancellation.failed`: When the cancellation process failed.
- `cancellation.not_applicable`: When cancel request could not be applied.
- `ip.not_trusted`: When the client IP is not trusted. Read More.
- `security.auth_required`: The bearer JWT token is not present in request header as the `Authorization` parameter.
  Read More.
- `token.verification_failed`: The access token is invalid or expired. Read More.
- `web.invalid_or_missing_body`: When the request body is not a valid JSON. Read More.
- `server.error`: Internal server error. Please tell us the value of fingerprint to be able to track the exact problem internally.

Authorization: Bearer JWT token in header. Read More.

Parameters

Type	Name	Description	Schema
Path	**refundId** *required*	is equal to `purchaseId` (purchase identifier)	integer (int64)
Body	**dto** *optional*	Encapsulated the information required to do an action on a (partial) refund.	RefundActionDto

Responses

HTTP Code	Description	Schema
200	OK	No Content

Example HTTP request

```
POST /ppg/v3/purchases/refunds/1234/cancel HTTP/1.1
Content-Type: application/json
Content-Length: 59
Host: napi.jibit.ir
Authorization: Bearer {client.jwt.accessToken}

{
  "transferId" : "trf-id-1",
  "partialRefundIndex" : 1
}
```

Example HTTP response

```
HTTP/1.1 200 OK
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
```

Example Curl request

```
$ curl 'https://napi.jibit.ir/ppg/v3/purchases/refunds/1234/cancel' -i -X POST \
    -H 'Content-Type: application/json' \
    -H 'Authorization: Bearer {client.jwt.accessToken}' \
    -d '{
  "transferId" : "trf-id-1",
  "partialRefundIndex" : 1
}'
```

2.4.4. Ignore Cancelling Refund

```
POST /v3/purchases/refunds/{refundId}/ignore-cancellable
```

Description

Ignores a cancelling refund.

Possible Error Codes:

- `transfer.not_found`: When the refundId is not found.
- `ip.not_trusted`: When the client IP is not trusted. Read More.
- `security.auth_required`: The bearer JWT token is not present in request header as the `Authorization` parameter.
  Read More.
- `token.verification_failed`: The access token is invalid or expired. Read More.
- `web.invalid_or_missing_body`: When the request body is not a valid JSON. Read More.
- `server.error`: Internal server error. Please tell us the value of fingerprint to be able to track the exact problem internally.

Authorization: Bearer JWT token in header. Read More.

Parameters

Type	Name	Description	Schema
Path	**refundId** *required*	is equal to `purchaseId` (purchase identifier)	integer (int64)
Body	**dto** *optional*	Encapsulated the information required to do an action on a (partial) refund.	RefundActionDto

Responses

HTTP Code	Description	Schema
200	OK	No Content

Example HTTP request

```
POST /ppg/v3/purchases/refunds/1234/ignore-cancellable HTTP/1.1
Content-Type: application/json
Content-Length: 55
Host: napi.jibit.ir
Authorization: Bearer {client.jwt.accessToken}

{
  "transferId" : "tf-2",
  "partialRefundIndex" : 1
}
```

Example HTTP response

```
HTTP/1.1 200 OK
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
```

Example Curl request

```
$ curl 'https://napi.jibit.ir/ppg/v3/purchases/refunds/1234/ignore-cancellable' -i -X
POST \
    -H 'Content-Type: application/json' \
    -H 'Authorization: Bearer {client.jwt.accessToken}' \
    -d '{
  "transferId" : "tf-2",
  "partialRefundIndex" : 1
}'
```

2.4.5. Retry Refund

```
POST /v3/purchases/refunds/{refundId}/retry
```

Description

Retries a failed refund.

Possible Error Codes:

- `transfer.not_found`: When the refundId is not found.
- `retry.failed`: When the retry process failed.
- `retry.not_applicable`: When retry request could not be applied.
- `ip.not_trusted`: When the client IP is not trusted. Read More.
- `security.auth_required`: The bearer JWT token is not present in request header as the

  `Authorization` parameter.
  Read More.

- `token.verification_failed`: The access token is invalid or expired. Read More.
- `web.invalid_or_missing_body`: When the request body is not a valid JSON. Read More.
- `server.error`: Internal server error. Please tell us the value of fingerprint to be able to track the exact problem internally.

Authorization: Bearer JWT token in header. Read More.

Parameters

Type	Name	Description	Schema
Path	**refundId** *required*	is equal to `purchaseId` (purchase identifier)	integer (int64)
Body	**dto** *optional*	Encapsulated the information required to do an action on a (partial) refund.	RefundActionDto

Responses

HTTP Code	Description	Schema
200	OK	No Content

Example HTTP request

```
POST /ppg/v3/purchases/refunds/1234/retry HTTP/1.1
Content-Type: application/json
Content-Length: 55
Host: napi.jibit.ir
Authorization: Bearer {client.jwt.accessToken}

{
  "transferId" : "tf-3",
  "partialRefundIndex" : 1
}
```

Example HTTP response

```
HTTP/1.1 200 OK
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
```

Example Curl request

```
$ curl 'https://napi.jibit.ir/ppg/v3/purchases/refunds/1234/retry' -i -X POST \
    -H 'Content-Type: application/json' \
    -H 'Authorization: Bearer {client.jwt.accessToken}' \
    -d '{
  "transferId" : "tf-3",
  "partialRefundIndex" : 1
}'
```

2.4.6. Verify Refund

```
POST /v3/purchases/refunds/{refundId}/verify
```

Description

Verifies a refund. If the verifying refund option is enabled, the client should verify its refund to start actual refunding.

Possible Error Codes:

- `transfer.not_found`: When the refundId is not found.
- `purchase.not_found`: When the purchase is not found.
- `verify.failed`: There is an internal issue while verifying. Please try again.
- `verify.not_applicable`: The refund is already verified, and it can't be verified again.
- `ip.not_trusted`: When the client IP is not trusted. Read More.
- `security.auth_required`: The bearer JWT token is not present in request header as the `Authorization` parameter.
  Read More.
- `token.verification_failed`: The access token is invalid or expired. Read More.
- `server.error`: Internal server error. Please tell us the value of fingerprint to be able to track the exact problem internally.

Authorization: Bearer JWT token in header. Read More.

Parameters

Type	Name	Description	Schema
Path	**refundId** *required*	is equal to `purchaseId` (purchase identifier)	integer (int64)
Body	**dto** *optional*	Encapsulated the information required to do an action on a (partial) refund.	RefundActionDto

Responses

HTTP Code	Description	Schema
204	No Content	No Content

Example HTTP request

```
POST /ppg/v3/purchases/refunds/1234/verify HTTP/1.1
Content-Type: application/json
Content-Length: 57
Host: napi.jibit.ir
Authorization: Bearer {client.jwt.accessToken}

{
  "transferId" : "trf-id",
  "partialRefundIndex" : 1
}
```

Example HTTP response

```
HTTP/1.1 204 No Content
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
```

Example Curl request

```
$ curl 'https://napi.jibit.ir/ppg/v3/purchases/refunds/1234/verify' -i -X POST \
    -H 'Content-Type: application/json' \
    -H 'Authorization: Bearer {client.jwt.accessToken}' \
    -d '{
  "transferId" : "trf-id",
  "partialRefundIndex" : 1
}'
```

2.5. Settlement API

Provides APIs to manage settlement-related APIs.

2.5.1. Filter Settlements

```
GET /v3/settlements
```

Description

Filters the settlements with provided criteria and page request.
All criteria parameter values should be URL-Encoded.
Possible Error Codes:

- `page_number.max_exceeded`: The page number is exceeded its allowed value.
- `page_size.max_exceeded`: The page size is exceeded its allowed size.
- `client.not_active`: When the client is not active.
- `ip.not_trusted`: When the client IP is not trusted. Read More.
- `security.auth_required`: The bearer JWT token is not present in request header as the `Authorization` parameter.
  Read More.
- `token.verification_failed`: The access token is invalid or expired. Read More.
- `server.error`: Internal server error. Please tell us the value of fingerprint to be able to track the exact problem internally.

Authorization: Bearer JWT token in header. Read More.

Parameters

Type	Name	Description	Schema
Query	**page** *optional*	Represents the page number; One indexed. The `null` means the first page. The max page number is 20. Example: `1`	integer (int32)
Query	**settlementId** *optional*	Finds a settlement by its identifier. The `null` means no filtering at all. Example: `1234455`	integer (int64)
Query	**size** *optional*	Represents the page size. The `null` means to use default size. The default page size is 25. The max page size is 250. Example: `10`	integer (int32)

Responses

HTTP Code	Description	Schema
200	The list of paginated settlements.	Paginated«DetailedSettlementDto»

Example HTTP request

```
GET /ppg/v3/settlements?settlementId=12345&page=1&size=20 HTTP/1.1
Host: napi.jibit.ir
Authorization: Bearer {client.jwt.accessToken}
```

Example HTTP response

```
HTTP/1.1 200 OK
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Content-Type: application/json
Content-Length: 693

{
  "pageNumber" : 1,
  "size" : 20,
  "numberOfElements" : 1,
  "hasNext" : false,
  "hasPrevious" : false,
  "elements" : [ {
    "settlementId" : 12345,
    "terminalId" : "6e890d5d-e707-42cf-a089-194c7f68f286",
    "fileName" : "file-name",
    "state" : "IN_PROGRESS",
    "amount" : 100000,
    "wage" : 555555,
    "directSettlement" : true,
    "currency" : "IRR",
    "cutoff" : "06:01:54.24712077",
    "acceptorCode" : "acceptor-code",
    "ledgerAccount" : "ledger-account",
    "iin" : 4433333,
    "paymentFacilitatorIban" : "IR9999",
    "settlementIban" : "IR3333",
    "createdAt" : "2024-11-13T02:31:54.247176214Z",
    "modifiedAt" : "2024-11-13T02:31:54.247178292Z"
  } ]
}
```

Example Curl request

```
$ curl 'https://napi.jibit.ir/ppg/v3/settlements?settlementId=12345&page=1&size=20' -i
-X GET \
    -H 'Authorization: Bearer {client.jwt.accessToken}'
```

2.6. Token API

Provides APIs to manage tokens of the client. These APIs are permitted to be called without the authorization token.

2.6.1. Generate Token

```
POST /v3/tokens
```

Description

Generates a pair of access/refresh tokens by using API/secret keys.

Possible Error Codes:

- `security.bad_credentials`: When provided api/secret keys are not valid.
- `client.not_active`: When the client is not active.
- `apiKey.is_required`: When api key is not provided.
- `secretKey.is_required`: When secret Key is not provided.
- `web.invalid_or_missing_body`: When the request body is not a valid JSON. Read More.
- `server.error`: Internal server error. Please tell us the value of fingerprint to be able to track the exact problem internally.

`Authorization` header parameter is NOT required.

Parameters

Type	Name	Description	Schema
Body	**dto** *required*	The dto encapsulating the information to generate a pair of access/refresh tokens.	GenerateTokenDto

Responses

HTTP Code	Description	Schema
200	The generated pair of access/refresh tokens.	TokenDto

Example HTTP request

```
POST /ppg/v3/tokens HTTP/1.1
Content-Type: application/json
Content-Length: 56
Host: napi.jibit.ir

{
  "apiKey" : "api-key",
  "secretKey" : "secret-key"
}
```

Example HTTP response

```
HTTP/1.1 200 OK
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Content-Type: application/json
Content-Length: 72

{
  "accessToken" : "access-token",
  "refreshToken" : "refresh-token"
}
```

Example Curl request

```
$ curl 'https://napi.jibit.ir/ppg/v3/tokens' -i -X POST \
    -H 'Content-Type: application/json' \
    -d '{
  "apiKey" : "api-key",
  "secretKey" : "secret-key"
}'
```

2.6.2. Refresh Token

```
POST /v3/tokens/refresh
```

Description

Refreshes the access/refresh tokens by using previously generated refresh token.

Possible Error Codes:

- `security.bad_credentials`: When provided refresh token is not valid.
- `client.not_active`: When the client is not active.

- `refreshToken.is_required`: When refresh token is not provided.
- `web.invalid_or_missing_body`: When the request body is not a valid JSON. Read More.
- `server.error`: Internal server error. Please tell us the value of fingerprint to be able to track the exact problem internally.

`Authorization` header parameter is NOT required.

Parameters

Type	Name	Description	Schema
Body	**dto** *required*	The dto encapsulating the information to refresh the access/refresh tokens for the client.	RefreshTokenDto

Responses

HTTP Code	Description	Schema
200	The generated pair of access/refresh tokens.	TokenDto

Example HTTP request

```
POST /ppg/v3/tokens/refresh HTTP/1.1
Content-Type: application/json
Content-Length: 38
Host: napi.jibit.ir

{
  "refreshToken" : "refresh-token"
}
```

Example HTTP response

```
HTTP/1.1 200 OK
Vary: Origin
Vary: Access-Control-Request-Method
Vary: Access-Control-Request-Headers
Content-Type: application/json
Content-Length: 72

{
  "accessToken" : "access-token",
  "refreshToken" : "refresh-token"
}
```

Example Curl request

```
$ curl 'https://napi.jibit.ir/ppg/v3/tokens/refresh' -i -X POST \
    -H 'Content-Type: application/json' \
    -d '{
  "refreshToken" : "refresh-token"
}'
```

Chapter 3. Definitions

3.1. Balance

Name	Description	Schema
amount *optional*	Represents the amount of current balance type.	integer (int64)
balanceType *optional*	Represents the balance type. Its possible values are: `WLT`: Indicates the default (not settleable) wallet of the client. It is the default balance type. `STL`: Indicates the settleable wallet of client. `FEE`: Indicates the post-paid fee that client needs to pay as the service subscription fee. `SHW`: Indicates the pre-paid fee (Shaparak Wage) that we would subtract from settlement amount as the service subscription fee. `BLK`: Indicates the blocked balance from the client `STL` balance type.	string
currency *optional*	Represents the currency of current balance type.	enum (IRR)

3.2. ClientBalances

Name	Description	Schema
balances *optional*	Represents the list of ledger account balances which may contain the default wallet and pre-paid/post-paid fees and settleable wallet.	< Balance > array

3.3. CreatePurchaseDto

Name	Description	Schema
additionalData *optional*	Represents the additional data provided by client in JSON format. Optional. Example: `{"someTag": "some-value"}`	object
amount *required*	Represents the amount of purchase. Example: `250000` **Minimum value** : `5000`	integer (int64)

Name	Description	Schema
callbackUrl *required*	Represents a callback which we should call in order to notify the client about a particular purchase state. For example, when the user pays the purchase, we would redirect the user to this callback and let the client know about the payment. This must be a valid URL address. Maximum length is 1024 characters. Example: `https://api.client.ir/purchases/123563/callback`	string
clientReferenceNumber *required*	Represents a reference number provided by client to trace purchase in our system. The reference number between client purchases must be unique. We will check this on our side. The maximum length is 50 characters. Example: `5435436`	string
currency *required*	Represents the currency of `amount` and `wage`.	enum (IRR)
description *optional*	Represents the client-provided description. Optional. The Maximum length is 256 characters. Example: `some related description`	string
payerCardNumber *optional*	Bank card number which is required to do the transaction. Only this card number can be used to do the transaction. Optional. This feature is not enabled by default on certain PSPs and must be enabled by admin. If Client uses this feature when it is not enabled, PPG simply ignores it and creates the purchase as normal. Example: `6037997122223333`	string
payerCardNumbers *optional*	Bank card numbers which are required to do the transaction. Only these card numbers can be used to do the transaction. Optional. This feature is not enabled by default on certain PSPs and must be enabled by admin. If Client uses this feature when it is not enabled, PPG simply ignores it and creates the purchase as normal. Example: `["6037997122223333", "6219861922223333"]`	< string > array
payerMobileNumber *optional*	Represents the mobile number of the payer. It will be sent to Psp to autocomplete the payment fields. Example: `09131234321` **Pattern** : `"\\s*(?:(?:\\+|00)98|0)9\\d{9}\\s*"`	string

Name	Description	Schema
payerNationalCode *optional*	Represents the national code of card owner. Used to verify the identity of the card owner. Optional. This feature is not enabled by default and must be enabled by admin. If Client uses this feature when it is not enabled, PPG simply ignores it and creates the purchase as normal. Example: `0921456778`	string
userIdentifier *optional*	Represents the user identifier in the client system. Maximum length is 50 characters. Example: `a.pourtaghi` or `09185674534`	string
wage *optional*	Represents the wage of purchase. The user will pay the `amount` + `wage`. The Default value is 0. This field can not be bigger that 15 % of the `amount` value. Example: `5000` **Minimum value** : `0`	integer (int64)

3.4. DetailedPurchaseDto

Name	Description	Schema
additionalData *optional*	An optional additional data provided by the client in JSON format. Example: `{"someTag": "some-value"}`	object
affiliateFee *optional*	Represents the reseller affiliate fee for this purchase.	integer (int64)
amount *optional*	Represents the amount of purchase.	integer (int64)
billingDate *optional*	When was the billing date? Used to generate the billing receipt. Example: `2021-04-04T14:21:25Z`	string (date-time)
callbackUrl *optional*	Represents a callback which we should call in order to notify the client about a particular purchase state. For example, when the user pays the purchase, we would redirect the user to this callback and let the client know about the payment.	string
clientReferenceNumber *optional*	A string value from client side to trace purchase in our system. This value is unique between client purchases.	string
createdAt *optional*	When the purchase created? Example: `2021-04-03T13:21:25Z`	string (date-time)

Name	Description	Schema
currency *optional*	Represents the currency type.	enum (IRR)
description *optional*	An optional client provided description for the purchase.	string
expirationDate *optional*	When has the purchase been expired?	string (date-time)
fee *optional*	Represents the Jibit fee for this purchase.	integer (int64)
feePaymentType *optional*	Represents the payment type of purchase fee.	enum (POST_PAID, PRE_PAID)
hasContradiction *optional*	Indicates whether the purchase had a contradiction or not.	boolean
initPayerIp *optional*	The IP address of the user calling the payment URL generated by us.	string
netAmount *optional*	Represents the net amount of purchase that will be settled with the client. `Net Amount == Amount + Wage - Prepaid Fee - Prepaid Affiliate Fee - Shaparak Fee`. If the client has no reseller, affiliate fee will be assumed as 0 in the above formula. Fee and Affiliate Fee are decreased from purchase amount only in pre-paid mode. Notes: In any case, The `Amount + Wage` will be added to the client's wallet default balance type (WLT) on purchase success. After Shaparak (or PSP) settlement, the net amount will be added to the client's settleable balance type (STL) or settled directly to the client's IBAN. This is available only if the **Early Fee Calculation** feature is enabled.	integer (int64)
payerCardNumber *optional*	Represents the user card number that the purchase is forced to do the transaction.	string
payerMobileNumber *optional*	Represents the user mobile number.	string
payerNationalCode *optional*	Represents the user national code that is used to check the owner of the PSP payer card.	string

Name	Description	Schema
pspCardOwner *optional*	Represents the card owner name.	string
pspFailReason *optional*	Represents the PSP fail reason. **DEPRECATED. For removal in the future. Use `pspFailReasons` instead.**	string
pspFailReasons *optional*	Represents the list of PSP fail reasons.	< PspFailReason > array
pspHashedCardNumber *optional*	Hashed bank card number used to do the transaction. Provided by PSP.	string
pspMaskedCardNumber *optional*	Masked bank card number used to do the transaction.	string
pspName *optional*	Represents the PSP name which the purchase belongs to.	string
pspReferenceNumber *optional*	Represents the PSP reference number.	string
pspRrn *optional*	Represents the PSP Retrieval Reference Number, a key to uniquely identify a card transaction based on the ISO 8583 standard.	string
pspSettled *optional*	Whether the PSP settled the payment.	boolean
pspSettledAt *optional*	When the PSP settled the payment? Example: `2021-04-04T14:21:25Z`	string (date-time)
pspTraceNumber *optional*	Represents the PSP trace number.	string
purchaseId *optional*	Represents the purchase identifier.	integer (int64)
purchaseIdStr *optional*	Represents the purchase identifier as String to bypass deserialization issues in some programming languages e.x. Javascript and Typescript.	string
redirectPayerIp *optional*	The IP address of the user reported by the PSP.	string

Name	Description	Schema
refundInquiryResult *optional*	Refund Inquiry result from transferor.	TransferorInquiryResult
refundableAmount *optional*	Represents the refundable amount. It is equal to or less than `amount + wage`.	integer (int64)
refunded *optional*	Whether the purchase is refunded.	boolean
resellerCode *optional*	Represents the reseller code for this purchase.	string
settlementId *optional*	Indicates the identifier of the settlement which the purchase is settled by. Example: `453467457326547`	integer (int64)
shaparakFee *optional*	Represents the Shaparak fee for this purchase.	integer (int64)
state *optional*	Represents the simple purchase state. The following states indicate the purchase had a contradiction and now is resolved. * `MANUALLY_SUCCESS`: The purchase has been successful manually. * `REVERSED`: The purchase has been reversed manually.	enum (EXPIRED, FAILED, IN_PROGRESS, MANUALLY_SUCCESS, READY_TO_VERIFY, REVERSED, SUCCESS, UNKNOWN)
userIdentifier *optional*	Identifies the user in the client system.	string
verifiedAt *optional*	When was the payment verification date? Example: `2021-04-03T13:25:25Z`	string (date-time)
wage *optional*	Represents the wage of purchase given by client.	integer (int64)

3.5. DetailedPurchaseHistoryDto

Name	Description	Schema
clientRefNum *optional*	Represents the client reference number. A string value from client side to trace purchase in our system. This value is unique between client purchases.	string
createdAt *optional*	Represents the creation time of this purchase history, which indicates when purchase status got changed. Example: 2024-10-20T13:25:30.946Z	string (date-time)

Name	Description	Schema
newState *optional*	Represents the simplified version of purchase new state. The following states indicate the purchase had a contradiction and now is resolved. * `MANUALLY_SUCCESS`: The purchase has been successful manually. * `REVERSED`: The purchase has been reversed manually.	enum (EXPIRED, FAILED, IN_PROGRESS, MANUALLY_SUCCESS, READY_TO_VERIFY, REVERSED, SUCCESS, UNKNOWN)
purchaseId *optional*	Represents the purchase identifier.	integer (int64)

3.6. DetailedSettlementDto

Name	Description	Schema
acceptorCode *optional*	Represents the acceptor code.	string
affiliateFee *optional*	Represents the reseller's affiliate fee.	integer (int64)
amount *optional*	Represents the settlement amount.	integer (int64)
createdAt *optional*	When was the settlement created. Example: `2021-04-04T14:21:25Z`	string (date-time)
currency *optional*	Represents the currency type.	enum (IRR)
cutoff *optional*	Cutoff time.	LocalTime
directSettlement *optional*	Represents whether settlement is direct settled.	boolean
fileName *optional*	Represents the file name which was sent to shaparak.	string
iin *optional*	Represents the iin.	integer (int64)
ledgerAccount *optional*	Represents ledger account of client.	string
modifiedAt *optional*	When was the settlement modified. Example: `2021-04-03T13:25:25Z`	string (date-time)
paymentFacilitatorIban *optional*	Represents the pf Iban.	string

Name	Description	Schema
resellerCode *optional*	Represents the reseller code	string
settledAt *optional*	When was the settlement settled. Example: `2021-04-03T13:25:25Z`	string (date-time)
settlementIban *optional*	Represents the settlement Iban.	string
settlementId *optional*	Represents the settlement identifier.	integer (int64)
shaparakFailReason *optional*	Represents the shaparak fail reason.	enum (DATA_FORMAT_MISMATCH, DATA_LENGTH_MISMATCH, DUPLICATE_DATA, ERROR_ACCESSING_DATA, EXPIRED_REFERENCE, EXTERNAL_SERVICE_UNREACHABLE, GENERAL_EXTERNAL_SERVICE_ERROR, INCONSISTENT_DATA, INTERNAL_ERROR, INVALID_EXTERNAL_SERVICE_RESPONSE, MISSING_VALUE, NOT_ENOUGH_RESOURCES, NO_ACH_REF_NUM, OUT_OF_BOUNDS_DATA, REFERENCED_DATA_NOT_FOUND, UNKNOWN, UNKNOWN_DATA_PROVIDED)
shaparakReferenceNumber *optional*	Represents the shaparak reference number.	string

Name	Description	Schema
shaparakTrackingNumber *optional*	Represents the shaparak tracking number.	string
shaparakTransactionId *optional*	Represents the shaparak tracking id.	string
state *optional*	Represents the settlement state.	enum (EARLY_SETTLED, FAILED, IN_PROGRESS, OBSOLETE, SETTLED)
terminalId *optional*	Represents the terminal Id.	string (uuid)
wage *optional*	Represents the settlement wage.	integer (int64)

3.7. GenerateTokenDto

Name	Description	Schema
apiKey *required*	Represents the api key of client. Example: `aYPFphbbJF`	string
secretKey *required*	Represents the secret key of client. Example: `yUQO1dAD4iJDnSEkzt9BuNN_LEZifpns_L27C8Jjm091XxbzNP`	string

3.8. IdPaginated«DetailedPurchaseHistoryDto»

Name	Description	Schema
elements *optional*	The list of elements in the current page.	< DetailedPurchaseHistoryDto > array
hasNext *optional*	Indicates if there is another page following the current one.	boolean
nextPageId *optional*	The identifier for fetching the next page of results. This provides more efficient pagination compared to using a traditional page number. Note: This can be null if there are no more results or if the result set is empty.	string
size *optional*	The number of elements in the current page.	integer (int32)

Name	Description	Schema
totalCount *optional*	The total number of elements. If null, the total count was not calculated for faster response times.	integer (int64)

3.9. LocalTime

Type : object

3.10. Paginated«DetailedPurchaseDto»

Name	Description	Schema
elements *optional*	The actual paginated elements.	< DetailedPurchaseDto > array
hasNext *optional*	Can this page be followed by another page after it?	boolean
hasPrevious *optional*	Is there a previous page for this page?	boolean
numberOfElements *optional*	Total number of elements.	integer (int64)
pageNumber *optional*	The current 1-index page number.	integer (int32)
size *optional*	The size of current page.	integer (int32)

3.11. Paginated«DetailedSettlementDto»

Name	Description	Schema
elements *optional*	The actual paginated elements.	< DetailedSettlementDto > array
hasNext *optional*	Can this page be followed by another page after it?	boolean
hasPrevious *optional*	Is there a previous page for this page?	boolean
numberOfElements *optional*	Total number of elements.	integer (int64)
pageNumber *optional*	The current 1-index page number.	integer (int32)

Name	Description	Schema
size *optional*	The size of current page.	integer (int32)

3.12. PspFailReason

Name	Description	Schema
code *optional*	Represents the Shaparak error code. Example: 54	integer (int32)
description *optional*	Represents a human-readable description of the error	string

Name	Description	Schema
pspError *optional*	**DEPRECATED. Use `code` instead** Possible values are: CARD_HAS_NO_CREDIT_ACCOUNT,INSUFFICIENT_FUNDS,ALREADY_PAID,CARD_IS_STOLEN,NO_ROUTE_TO_DESTINATION,CONTRADICTION_RESOLVING_WAS_SUCCESSFUL,ACCEPTOR_NOT_SUPPORTED_BY_SWITCH,INSUFFICIENT_RESOURCES_OR_LEGAL_ISSUE,CARD_NO_GENERAL_ACCOUNT,SHAPARAK_UNKNOWN,CARD_HAS_BEEN_EXPIRED,INVALID_TRANSACTION,CANCELLED_BY_CARD_ISSUER,SUSPECTED_FRAUD,PSP_PAYER_CARD_NOT_MATCHED,SYSTEM_ERROR,INVALID_TRANSACTION_DAY,TRANSACTION_NOT_ALLOWED_BY_CARD_OWNER,DUPLICATE_TRANSACTION,KEY_CHANGE_IN_PROGRESS,UNKNOWN,CANCELLED,TRANSACTION_NOT_ALLOWED_BY_TERMINAL,AMOUNT_EXCEEDS_LIMIT,INVALID_TRANSACTION_AMOUNT,MESSAGE_FORMAT_ERROR,CARD_HAS_NO_TRANSACTION_ACCOUNT,NO_DATA_FOUND,INVALID_ACCEPTOR_WAGE,CARD_TRACK3_UPDATE_REQUIRED,TRANSACTION_FAILED,SIGN_OFF_BY_PROVIDER,CARD_IS_LIMITED,END_OF_DAY_OPERATION,CARD_HAS_NO_SAVINGS_ACCOUNT,INVALID_CARD_NUMBER,UNKNOWN_ERROR_INQUIRY_FROM_AGENT,INVALID_CARD,MINIMUM_AMOUNT_LIMIT,SIGN_OFF_BY_SHOPPER_OR_PROVIDER,DESTINATION_BANK_UNAVAILABLE,CARD_NOT_ACTIVE,CARD_IS_LOST,OTHER_BANK_ERRORS,CARD_SUSPECTED_FRAUD,TOO_MANY_INCORRECT_PASSWORD,DESTINATION_BANK_INACTIVE,TRANSACTION_TIMED_OUT_IN_BANK_NETWORK,CARD_HAS_NO_INVESTMENT_ACCOUNT,CARD_RESTRICTED,CANCELLED_BY_USER_INQUIRY_FROM_AGENT,RETRY_TRANSACTION,REQUEST_IN_PROGRESS,CARD_CAPTURED,SECURITY_VIOLATION,UNKNOWN_ERROR_OCCURRED,INVALID_PASSWORD,EXCESSIVE_TRANSACTION_REQUESTS,INVALID_ACCOUNT_OR_CONNECTION,INVALID_CARD_ISSUER,CANCELLED_BY_USER,STORE_ACCEPTOR_IS_INVALID,UNSUPPORTED_OPERATION,TRANSACTION_SUCCESSFUL_WITH_IDENTITY,TRANSACTION_TIMED_OUT,CARD_CAPTURED_SPECIAL_CONDITIONS	string

3.13. PurchaseCreationResult

Name	Description	Schema
affiliateFee *optional*	Represents the affiliate fee of client’s reseller. This is available only if the **Early Fee Calculation** feature is enabled and the client has a reseller.	integer (int64)

Name	Description	Schema
clientReferenceNumber *optional*	Represents the reference number provided by client to trace the purchase in our system. The reference number between client purchases must be unique.	string
currency *optional*	Represents the currency type.	enum (IRR)
fee *optional*	Represents the Jibit fee of the purchase. This is available only if the **Early Fee Calculation** feature is enabled.	integer (int64)
netAmount *optional*	Represents the net amount of purchase that will be settled with the client. `Net Amount == Amount + Wage - Prepaid Fee - Prepaid Affiliate Fee - Shaparak Fee`. If the client has no reseller, affiliate fee will be assumed as 0 in the above formula. Fee and Affiliate Fee are decreased from purchase amount only in pre-paid mode. Notes: In any case, The `Amount + Wage` will be added to the client's wallet default balance type (WLT) on purchase success. After Shaparak (or PSP) settlement, the net amount will be added to the client's settleable balance type (STL) or settled directly to the client's IBAN. This is available only if the **Early Fee Calculation** feature is enabled.	integer (int64)
pspSwitchingUrl *optional*	The URL pointing to our PSP switching web service. The client must redirect its user to this URL. The user will be redirected to a PSP after initializing the purchase.	string
purchaseId *optional*	The identifier for the just created purchase. It is generated by our service.	integer (int64)
purchaseIdStr *optional*	The identifier for the just created purchase as String to bypass deserialization issues in some programming languages e.x. Javascript and Typescript. It is generated by our service.	string
shaparakFee *optional*	Represents the Shaparak fee of the purchase.	integer (int64)

3.14. RefreshTokenDto

Name	Description	Schema
refreshToken *required*	Represents the refresh token. Used to generate a new pair of access/refresh tokens for the client.	string

3.15. RefundActionDto

Name	Description	Schema
transferId *optional*	This is the unique identifier of a partial refund. * If there is only one refund in the batch, you do not need to specify `transferId`. * If there are multiple refunds in the batch: You must specify `transferId` to indicate the position of the partial refund.	string

3.16. RefundPurchaseDto

Name	Description	Schema
amount *optional*	The amount which the client want to refund (for partial refund).	integer (int64)
cancellable *optional*	Whether to pend the refund request and perform it after one hour.	boolean
clientReferenceNumber *optional*	Represents the reference number provided by client to trace the purchase in our system.	string
purchaseId *optional*	Represents the purchase id.	integer (int64)

3.17. ReversePurchaseDto

Name	Description	Schema
clientReferenceNumber *optional*	Represents the reference number provided by client to trace the purchase in our system.	string
purchaseId *optional*	Represents the purchase id.	integer (int64)

3.18. ReverseResultDto

Name	Description	Schema
status *optional*	It is the status of reverse. Possible states are: * `SUCCESSFUL`: The reverse was successful. * `UNKNOWN`: The reverse was unknown (typically because of a network failure). The client must retry the reverse for this purchase. * `ALREADY_REVERSED`: The purchase is already reversed. * `NOT_REVERSIBLE`: The purchase is not prepared to reverse and thus is not reversible. * `FAILED`: The purchase reversion failed on PSP side.	enum (ALREADY_REVERSED, FAILED, NOT_REVERSIBLE, SUCCESSFUL, UNKNOWN)

3.19. TokenDto

Name	Description	Schema
accessToken *optional*	Represents the new access token.	string
refreshToken *optional*	Represents the new refresh token.	string

3.20. Transfer

Name	Description	Schema
amount *optional*		integer (int64)
bankTransferID *optional*		string
cancellable *optional*		boolean
createdAt *optional*		string (date-time)
currency *optional*		string
description *optional*		string
destination *optional*		string
destinationFirstName *optional*		string

Name	Description	Schema
destinationLastName *optional*		string
failReason *optional*		string
feeAmount *optional*		integer (int64)
feeCurrency *optional*		string
metadata *optional*		string
modifiedAt *optional*		string (date-time)
notifyURL *optional*		string
partialRefundIndex *optional*	Represents the position of a partial refund within a batch of refunds for a purchase. It is one-based, meaning it starts at 1. **DEPRECATED. Use `transferId` instead**	integer (int32)
paymentID *optional*		string
refundId *optional*	represents the purchase identifier of the refund.	integer (int64)
state *optional*	State can have these following values: TRANSFERRED means refund is completely done. FAILED means refund is failed. INITIALIZED and IN_PROGRESS mean refund is in progress. CANCELLING means it is in the middle of cancel process. CANCELLED means refund is cancelled.	enum (CANCELLED, CANCELLING, CORE_SUBMITTED, DESTINATION_IDENTIFIED, FAILED, FEE_COMPUTED, INITIALIZED, IN_PROGRESS, MANUALLY_FAILED, ON_HOLD, ON_HOLD_BALANCES_NOT_ENOUGH, ON_HOLD_WAIT_FOR_MANUAL_SUBMISSION, ON_HOLD_WAIT_FOR_VERIFY, RETRYING, TRANSFERRED)

Name	Description	Schema
transferID *optional*		string
transferMode *optional*		enum (ACH, NORMAL, RTGS)

3.21. TransferorInquiryResult

Name	Description	Schema
batchID *optional*	Represents the batch id.	string
refundedAmount *optional*		integer (int64)
transfers *optional*	Represents the list of Transfers.	< Transfer > array

3.22. TransferorResult

Name	Description	Schema
batchId *optional*	The batch id returned from transferor.	string
partialRefundIndex *optional*	Represents the position of a partial refund within a batch of refunds for a purchase. It is one-based, meaning it starts at 1. **DEPRECATED. Use `transferId` instead**	integer (int32)
refundId *optional*	represents the purchase identifier of the refund.	integer (int64)
transferId *optional*	The transfer id returned from transferor.	string

3.23. VerificationResultDto

Name	Description	Schema
status *optional*	It is the status of verification. Possible states are: * `SUCCESSFUL`: The verification was successful. * `FAILED`: The verification failed. The payment amount will return to the user bank account. * `REVERSED`: Because of a fraud in payment from user, the payment reversed. The payment amount will return to the user bank account. See Purchase Flow step 10. * `UNKNOWN`: The verification was unknown (typically because of a network failure). The client could retry the verification for this purchase before its expiration. The client also could inquiry the purchase later to identify the actual payment verification state. * `ALREADY_VERIFIED`: The purchase already verified. For example a purchase with `SUCCESS` state is already verified. * `NOT_VERIFIABLE`: The purchase is not prepare to verify and thus is not verifiable. Only purchases in `READY_TO_VERIFY` state can be verified.	enum (ALREADY_VERIFIED, FAILED, NOT_VERIFIABLE, REVERSED, SUCCESSFUL, UNKNOWN)