SPARROW

*Transact, Grow, SPARROW.*

# Merchant Public Services

Transaction Query API

**Version** 1.2

**Released** February 2017

## Revision History

| Date | Revision | Comments | Author |
|---|---|---|---|
| 2016-03-10 | 1.0 | Initial document created | Vlad Volosuk |
| 2016-03-23 | 1.1 | Action route URL is corrected | Vlad Volosuk |
| 2017-01-19 | 1.2 | Streamline Authentication is added | Sergey Belov |
| 2017-03-01 | 1.3 | Review document | Ilya Tretyakov |

## *Table of Contents*

# Overview

The Sparrow Gateway provides a set of services which can be used for integrating the gateway with merchant's IT systems. Merchant Public API provides the following RESTfull services:

- **Account** service – it is auxiliary service to obtain the authentication token required for invoking other Merchant Public API services
- **Transaction** service - this service allows querying transactions processed by the merchant in the Sparrow Gateway
- **Terminal** service – this service provides control functions over transaction processing

This document provides architectural overview of Merchant Public API and describes the services included into this API.

# Architectural Overview

Merchant Public API consists of a number of *Actions* which are logically grouped into services. This section describes the structure of HTTP requests which are used for invoking *Actions*.
The legend:

- `<TERM>` indicates a parameter that will be described below of the snippet
- `[any-text]` indicates an optional part that can be not provided in requests for some *Actions* (see the specification of the particular *Action*)

### *Action Route*

A route to an *Action* is a URL-based path that uniquely identifies the *Action* for invocation and invocation parameters:

```
<URL>/api/public/merchant/<SERVICE>/<ACTION>[/<PATH-PARAMETERS>][?<QUERY-PARAMETERS>]
```

| Term | Description |
|---|---|
| **<URL>** | The address of the Sparrow Gateway server where the service is hosted.<br><br>Merchants have to use the following URL for the production (life) integrations:<br><br>https://secure.sparrowone.com |
| **<SERVICE>** | The name of the service where the *Action* is located. Examples: **account**, **transactions**, **terminal**. See documentation of services below in this document. |
| **<ACTION>** | The name of the *Action*. See documentation of services below in this document.<br><br>Note: normally the name of the action must be always presented in *Action Route*. However, the service can have a default Action. In this case, this action is invoked without the name in the *Action Route.* Example: **authenticate** action in the **account** service. |

| | |
|---|---|
| **<PATH-PARAMETERS>** | Path parameters are used to clarifies an entity for which the action has to be invoked |
| **<QUERY-PARAMETERS>** | Additional parameters provided in the form of HTTP query string |

### *Action Request*

An *Action* can be invoked by a HTTP request:

```
<HTTP-VERB> <ACTION-ROUTE> HTTP/1.1
Content-Type: <REQUEST-CONTENT-TYPE>
Accept: <RESPONSE-CONTENT-TYPE>
Authorization: <REQUEST-AUTH-PARAMS>
[other HTTP headers]

<REQUEST-DATA>
```

| Term | Description |
|---|---|
| **<HTTP-VERB>** | HTTP verb for invoking the service *Action*. Possible values are:<br>▪ GET<br>▪ POST<br>▪ PUT<br>▪ DELETE<br>It is not mandatory for *Actions* to support all 4 verbs (but at least one verb must be supported). The verb can be interrupted as a sub-action supported by the *Action*. See the service documentation to select the proper verb for the *Action.* |
| **<ACTION-ROUTE>** | An URL-based path that uniquely identifies the *Action* for invocation and invocation parameters. See the section above for details. |
| **<REQUEST-CONTENT-TYPE>** | The content type of the **<REQUEST-DATA>**. Possible values are:<br>▪ *application/xml*<br>▪ *application/json*<br><br>See *JSON Objects* and *XML* Objects sections below for object format details. |
| **<RESPONSE-CONTENT-TYPE>** | The expected content type for the response. Possible values are:<br>▪ *application/xml*<br>▪ *application/json*<br><br>If *Accept* HTTP header is not provided in the request, then the response will be returned in JSON format.<br><br>See *JSON Objects* and *XML* Objects sections below for object format details. |
| **<REQUEST-AUTH-PARAMS>** | The authorization parameters. See *Authentication* section. |
| **<REQUEST-DATA>** | Request data object formatted according **<REQUEST-CONTENT-TYPE>**. Names of request parameters are case-sensitive. |

| | |
|---|---|
| | See the *Action* documentation for parameters supported by the *Action* in **<REQUEST-DATA>.** |

### *Action Response*

Here is a format of *Action* responses:

```
HTTP/1.1 <HTTP-CODE> <HTTP-CODE-DESCRIPTION>
Content-Type: <RESPONSE-CONTENT-TYPE>
WWW-Authenticate: <RESPONSE-AUTH-TOKEN>
[other HTTP headers]

<RESPONSE-DATA>
```

| Term | Description |
|---|---|
| **<HTTP-CODE>** | A standard HTTP response code. Here are codes (with HTTP descriptions) commonly used in Merchant Public API:<br>▪ 200 OK<br>The action was executed successfully<br>▪ 403 Forbidden<br>The current user is not allowed to execute this *Action*<br>▪ 404 Not Found<br>The Action was requested for the entity that is not found on the system<br>▪ 400 Bad Request<br>The Action was executed with an error. More information about the error is returned in **<RESPONSE-DATA>**.<br>▪ 500 Internal Server Error<br>It is unwanted, but possible response that indicates about internal (unexpected) error during processing the *Action*.<br><br>Particular *Actions* may support other HTTP response codes, see the *Action* documentation for details. |
| **<HTTP-CODE-DESCRIPTION>** | A general message that clarifies the **<HTTP-CODE>**. See above for examples. |
| **<RESPONSE-CONTENT-TYPE>** | The content type of **<RESPONSE-DATA>**. Possible values are:<br>▪ *application/xml* (see *XML Objects* section for details)<br>▪ *application/json* (see *JSON Objects* section for details)<br><br>See *JSON Objects* and *XML* Objects sections below for object format details. |
| **<RESPONSE-AUTH-TOKEN>** | The authentication token. The response contains WWW-Authenticate HTTP header only when the request contains valid authentication parameters: credentials or API Key. See *Authentication* section. |
| **<RESPONSE-DATA>** | Response parameters formatted according **<REQUEST-CONTENT-TYPE>**.<br>Names of response parameters are case-sensitive. |

| | |
|---|---|
| | See the *Action* documentation for parameters supported by the *Action* in **<REQUEST-DATA>.** |

### *JSON Objects*

Action requests and responses can be formatted as JSON objects:

```
{
	"Username":"JoeDoe",
	"Password":"pwd"
}
```

See *Action* documentation for supported request/response parameters.

### *XML Objects*

Action requests and responses can be formatted as XML objects:

```
<LoginResponse>
  <AuthToken>TIUPQV+R1R5RGORA4IZVUIXIYEP+OFNB</AuthToken>
</LoginResponse>
```

See *Action* documentation for supported request/response parameters.

The name of the root element is composed as shown below:

- The root element for *Action* requests is named ***<ActionName>****Request* (e.g. LoginRequest)
- The root element for *Action* requests is named ***<ActionName>****Response* (e.g. LoginResponse)

where ***<ActionName>*** is the name of the current *Action*, the first letter is the upper case, others – in the low case.

### *Authentication*

Only authenticated users are allowed to invoke service *Actions*. Merchant Public API supports two action invocation workflows which differ in a way how authentication information is obtained and specified in API calls: explicit authentication and streamline authentication.

#### Streamline Authentication

Streamline authentication is a preferable way to work with Merchant Public API: authentication information is specified with the every API call in HTTP headers as Authorization parameter (see **Action Request** section above). The following forms are supported for this parameter:

- **Authorization: credentials=<credentials-string>**
  where <credentials-string> is <username>:<password> encoded by BASE64

  This form is used for the first API call if the user-based authentication scheme is used. Please note, the username cannot contains ":" symbol, otherwise the expression cannot be parsed correctly (the gateway's policy does not allow using ":" in user names as well).

- **Authorization: apikey=<api-string>**
  This form is used for the first API call if the API-based authentication scheme is used.

- **Authorization: token=<auth-token>**
  **Authorization: <auth-token>**

  These two forms are equal. They are recommended for using with second and forth API calls. <auth-token> is provided as WWW-Authenticate HTTP header in the response for the first API call: **WWW-Authenticate: <auth-token>** (see **Action Response** section above).

**Explicit Authentication**

Explicit authentication can be used if an integration does not have ability to specify/fetch HTTP header parameters. This workflow assumes that a special API call is made just to obtain an auth token.

- The app invoke **Authenticate** action (see **Account Service** section below). If the action is successful, then <auth-toke> is returned in the response.
- The app makes the first API call with providing **token=<auth-token>** in the request's query string
- The second and forth API calls are perfomed in the same way: by providing **token=<auth-token>** in the request's query string

# Account Service

The **Account** service is auxiliary service that allows retrieving the authentication token required for invoking other *Actions*.

### *Authenticate Action*

This action is used to obtain the authentication token for the specified user.

| Name | Value |
|---|---|
| **Service** | Account |
| **Action Name** | (default) |
| **Method** | POST |
| **Path Parameters** | No |
| **Query Parameters** | No |
| **Request** | The following items must be provided in **<REQUEST-DATA>**:<br>▪ **Username**<br>▪ **Password** |
| **Response** | The action returns the following items in **<RESPONSE-DATA>**:<br>▪ **AuthToken** – an authentication token for subsequent service actions calls |

# Transactions Service

The **Transactions** service provides a set of *Actions* to query transactions executed by the merchant.

### *Get Transaction Details Action*

This action is used to retrieve details about the particular transaction.

| Name | Value/Description |
|---|---|
| **Service** | Transactions |
| **Action Name** | Detail |
| **Method** | GET |
| **Path Parameters** | **/<TRANSACTION-ID>**<br><br>**<TRANSACTION-ID>** - is the identifier of the transaction (transid) which details are requested |
| **Query Parameters** | The following items can be provided in **<QUERY-PARAMETERS>**:<br>▪ **token** – the authentication token (required for explicit authentication scheme) |
| **Request** | No |
| **Response** | The details of the requested transaction (see below).<br><br>For XML formatter – the root element of the response is **<TransactionResponse>**. |

The table below enumerates transaction parameters which are returned by this action:

| Name | Data Type | Description |
|---|---|---|
| TransactionId | Integer | Payment transaction ID |
| MerchantAccountName | String | Merchant account name |
| CreationTime | Date/time | Payment operation date/time |
| ExternalId | String | External ID |
| Status | String | Operation status |
| CustomerCurrency | String | Currency |
| Message | String | Message |
| Amount | Money | Amount |
| FormattedAmount | String | Formatted amount |
| PaymentTypeName | String | Payment type name |
| AuthCode | String | Auth-code |
| AvsStatus | String | AVS status |
| CvvStatus | String | CVV status |
| CardLevelResultsText | String | Card level results |
| CommercialCardIndicatorText | String | Commercial card indicator |
| ProcessorName | String | Payment processor's name |
| TransactionType | String | Payment transaction's type |
| IsApproved | Boolean | Indicator whether the payment transaction was approved |
| **Billing** | | This element contains billing information |
| Billing/FullName | String | Full name |
| Billing/Company | String | Company name |
| Billing/Address1 | String | Address (first line) |
| Billing/Address2 | String | Address (second line) |

| | | |
|---|---|---|
| Billing/Country | String | Country |
| Billing/Phone | String | Phone number |
| Billing/Fax | String | Fax number |
| Billing/Email | String | E-mail |
| Billing/State | String | State |
| Billing/City | String | City |
| Billing/Zip | String | Zip |
| Billing/PostalCode | String | Postal code |
| **Customer** | | This element that contains customer information |
| Customer/BirthDate | String | Birth date |
| Customer/CourtesyCardId | String | Courtesy Card ID |
| Customer/DriverLicenseNumber | String | Driver's license number |
| Customer/DriverLicenseCountry | String | Driver's license country |
| Customer/DriverLicenseState | String | Driver's license state |
| **Shipping** | | This element contains shipping information |
| Shipping/FullName | String | Full name |
| Shipping/Company | String | Company name |
| Shipping/Address1 | String | Address (first line) |
| Shipping/Address2 | String | Address (second line) |
| Shipping/Country | String | Country |
| Shipping/Phone | String | Phone number |
| Shipping/Fax | String | Fax number |
| Shipping/Email | String | E-mail |
| Shipping/State | String | State |
| Shipping/City | String | City |
| Shipping/Zip | String | Zip |
| Shipping/PostalCode | String | Postal code |
| **Order** | | This element contains order information |
| Order/OrderId | String | Order ID |
| Order/Description | String | Order description |
| Order/PurchaseNumber | String | Purchase number |
| Order/ShippingAmount | String | Shipping amount |
| Order/TaxAmount | String | Tax amount |
| **Payment** | | This element contains payment information |
| Payment/MaskedAccountNumber | String | Masked account number |
| Payment/BankName | String | Bank name (for ACH and Checks) |
| Payment/RoutingNumber | String | Routing number (for ACH and Checks) |
| Payment/AccountType | String | Account type (for ACH and Checks) |
| Payment/AccountSubType | String | Account sub-type (for Checks) |
| Payment/CheckNumber | String | Check number (for Checks) |
| Payment/CardType | String | Card type (for Credit and Star cards) |
| Payment/ExpirationDate | String | Expiration date in MMYY format (for Credit cards) |
| **CustomFields** | | This element contains the array of custom fields included into the transaction |
| CustomFields/CustomField/Name | String | The name of the custom field |
| CustomFields/CustomField/Value | String | The value of the custom field |
| **Items** | | This element contains the array of purchase items |
| Items/Item/Id | Integer | Item ID |
| Items/Item/SkuNumber | String | SKU number |
| Items/Item/Description | String | Description |

| | | |
|---|---|---|
| Items/Item/Quantity | Number | Quantity |
| Items/Item/Amount | Money | Amount |
| Items/Item/TotalAmount | Money | Total amount |
| **Operations** | | This element contains the array of operations made in the scope of this transaction |
| Operations/Operations/Id | Integer | Payment operation ID |
| Operations/Operations/OperationId | Integer | Operation ID |
| Operations/Operations/Type | String | Operation type |
| Operations/Operations/Status | String | Operation status |
| Operations/Operations/Message | String | Message |
| Operations/Operations/ModuleName | String | Module name |
| Operations/Operations/ProcessorResponse | String | Payment processor response |
| Operations/Operations/CreationTime | Date/time | Operation date/time |
| Operations/Operations/Username | String | Username |
| Operations/Operations/SettlementStatus | String | Settlement status |
| Operations/Operations/Amount | Money | Operation amount |
| Operations/Operations/SettlementAmount | Money | Settlement amount |
| Operations/Operations/SettlementId | Integer | Settlement operation ID |

### *Query Transactions Action*

This action is used to retrieve a list of transaction.

| Name | Value/Description |
|---|---|
| **Service** | Transactions |
| **Action Name** | (default) |
| **Method** | GET |
| **Path Parameters** | No |
| **Query Parameters** | The following items can be provided in **<QUERY-PARAMETERS>**:<br>▪ **token** – the authentication token (required for explicit authentication scheme)<br>▪ **$filter** – filter expression (see OData section below)<br>▪ **$orderby** – field list (see OData section below)<br>▪ **$skip** – number of the items to skip (see OData section below)<br>▪ **$top** – number of the items in the result (see OData section below) |
| **Request** | None |
| **Response** | OData response (see OData section below). The structure of OData item is described below.<br><br>For XML formatter – the root element of the response is **<PageResponseOfTransactionReportItem>** |

Here is the structure of OData item returned in the response:

| Name | Data Type | Description |
|---|---|---|
| PaymentOperationId | Integer | Payment operation ID |
| TransactionId | Integer | Payment transaction ID |

| | | |
|---|---|---|
| TypeName | String | Operation type |
| OperationStatusName | String | Operation status |
| MerchantAccountName | String | Merchant account name |
| CustomerName | String | Customer name |
| OperationTime | Date/time | Operation time |
| OperationAmount | String | Formatted amount of the payment operation |
| Currency | String | Currency of the payment transaction |
| CompanyName | String | Customer's company name |
| PaymentTypeName | String | Payment type |
| Username | String | Username |
| ResponseCodeDescription | String | Response code description provided by the payment processor |
| Message | String | Message |
| AuthCode | String | Auth-code provided by the payment processor |
| Phone | String | Customer's phone number |
| OrderDescription | String | Order description |
| Address1 | String | Customer's address (first line) |
| City | String | Customer's city |
| PostalCode | String | Customer's zip |
| Email | String | Customer's e-mail |
| IpAddress | String | IP address |
| OrderId | String | Order ID |
| ClientExternalId | String | Id of the Data Vault customer |
| ExternalId | String | Id of the payment transaction on the payment processor's side |
| CreditCardIssuerName | String | Credit card's issuer |
| MerchantCompany | String | Merchant company name |
| BatchId | String | Batch ID |
| UserDepartment | String | User department |
| TransactionStatus | String | Status of the payment transaction |
| ModuleName | String | Module name |
| Address2 | String | Customer's address (second line) |
| CountryName | String | Customer's country |
| StateName | String | Customer's state |
| SourceType | String | Source type |
| ShippingFirstName | String | Shipping first name |
| ShippingLastName | String | Shipping last name |
| ShippingAddress1 | String | Shipping address (first line) |
| ShippingAddress2 | String | Shipping address (second line) |
| ShippingCity | String | Shipping city |
| ShippingEmail | String | Shipping e-mail |
| ShippingCountryName | String | Shipping country |
| ShippingStateName | String | Shipping state |
| ShippingPhone | String | Shipping phone number |
| ShippingPostalCode | String | Shipping zip |

## Appendix: OData Protocol

The Open Data Protocol (OData) is a data access protocol built over HTTP and it provides a way to query data over web in the RESTful manner.

This section describes OData protocol features supported by SPARROW Merchant Public API.

### *OData Request*

OData request is a pure HTTP GET request. The source of data for querying is represented by the request's URL. The query parameters are passed in the HTTP query string.
SPARROW Merchant Public API supports the following parameters to be passed into OData services:

| Name | Data Type | Description |
|---|---|---|
| $skip | Integer | Indicates how may items in the query result should be skipped before putting them into the response. This parameter is used to organize paging in OData responses. |
| $top | Integer | Indicates how may items should be put into the response. This parameter is used to organize paging in OData responses.<br>By default: $top=1000 |
| $filter | String | Actually contains a query expression. OData service returns only items which meet this filter condition.<br><br>Example: *TransactionId eq 12345 and OperationTime gt 2016-03-04T12* |
| $orderby | String | Allows specifying item ordering.<br><br>Example: TransactionId asc,CreationTime desc |

**OData Logical Operations**

It is possible to use the following logical operations for composing OData query (see $filter parameter):

| Operation Name | Operator | Description |
|---|---|---|
| **Equals** | eq | The eq operator returns true of the left operand is equal to the right operand, otherwise it returns false. The null value is equal to itself, and only to itself.<br>Example: Message eq 'APPROVE' |
| **Not Equals** | ne | The ne operator returns true if the left operand is not equal to the right operand, otherwise it returns false. The null value is not equal to any value but itself.<br>Example: Message ne 'APPROVE' |
| **Greater Than** | gt | The gt operator returns true if the left operand is greater than the right operand, otherwise it returns false. If any operand is null, the operator returns false.<br>For Boolean values true is greater than false.<br>Example: Amount gt 2.33 |
| **Greater Than Or Equal** | ge | The ge operator returns true if the left operand is greater than the right operand, otherwise it return false. |

| | | |
|---|---|---|
| | | If only operand is null, the operator returns false. If both operands are null, it returns true because null is equals to itself.<br>Example: Amount ge 2.33 |
| **Less Than** | lt | The lt operator returns true if the left operand is less than the right operand, otherwise it returns false.<br>If any operand is null, the operator returns false.<br>Example: Amount lt 2.33 |
| **Less Than or Equal** | le | The le operator returns true if the left operand is less than or equal to the right operand, otherwise it returns false.<br>If only one operand is null, the operator returns false. If both operands are null, it returns true because null is equal to itself.<br>Example: Amount le 2.33 |
| **And** | and | The and operator returns true if both the left and right operands evaluate to true, otherwise it returns false.<br>The null value is treated as unknown, so if one operand evaluates to null and the other operand to false, the and operator returns false. All other combinations with null return null.<br>Example: Message eq 'APPROVE' and Amount gt 2.33 |
| **Or** | or | The or operator returns false if both the left and right operands both evaluate to false, otherwise it returns true.<br>The null value is treated as unknown, so if one operand evaluates to null and the other operand to true, the or operator return true. All other combinations with null return null.<br>Example: Message eq 'APPROVE' or Message eq 'Success' |
| **Not** | not | The not operator returns true if the left operand returns false, otherwise it returns false.<br>The null value is treated as unknown, so not null returns null.<br>Example: not contains(Message, 'rejected') |

**OData String Functions**

It is possible to use the following string functions for composing OData query (see $filter parameter):

| Function | Description |
|---|---|
| **contains** | The contains function returns true if the second parameter string value is a substring of the first parameter string value, otherwise it returns false.<br><br>Example: contains(Message, 'rejected') |
| **endswith** | The endswith function returns true if the first parameter string value ends with the second parameter string value, otherwise it returns false.<br><br>Example: endswith(ModuleName, 'ACH') |
| **startswith** | The startswith function returns true if the first parameter string value starts with the second parameter string value, otherwise it returns false.<br><br>Example: startswith(ModuleName, 'MIMS') |
| **length** | The length function returns the number of characters in the parameter value.<br><br>Example: length(Message) eg 0 |
| **indexof** | The indexof function returns the zero-based character position of the first occurrence of the second parameter value in the first parameter value.<br><br>Example: indexof(Message, 'ACH') gt 0 |
| **substring** | The two argument substring function returns a substring of the first parameter string value, starting at the Nth character and finishing at the last character (where N is the second parameter integer value). The three argument substring function returns a substring of the first parameter string value identified by selecting M characters starting at the Nth character (where N is the second parameter integer value and M is the third parameter integer value). |
| **tolower** | The tolower function returns the input string value with all uppercase characters converted to lowercase according to Unicode rules. |
| **toupper** | The toupper function returns the input parameter string value with all lowercase characters converted to uppercase according to Unicode rules. |
| **trim** | The trim function returns the input parameter string value with all leading and trailing whitespace characters, according to Unicode rules, removed. |

| | |
|---|---|
| | Example: trim(CompanyName) eq CompanyName |
| **concat** | The concat function returns a string that appends the second parameter string to the first.<br><br>Example: concat(concat(Country, ' '), State) |

**OData Date/Time Functions**

It is possible to use the following date/time functions for composing OData query (see $filter parameter):

| Function | Description |
|---|---|
| **year** | The year function returns the year component of the date parameter value.<br><br>Example: year(OperationTime) eq 2015 |
| **month** | The month function returns the month component of the date parameter value.<br><br>Example: month(OperationTime) eq 2 |
| **day** | The day function returns the day component of the date parameter value.<br><br>Example: day(OperationTime) eq 25 |
| **hour** | The hour function returns the hour component of the date parameter value.<br><br>Example: hour(OperationTime) eq 12 |
| **minute** | The minute function returns the minute component of the date or time parameter value.<br>Example: minute(OperationTime) eq 20 |
| **second** | The second function returns the second component of the date or time parameter value.<br><br>Example: second(OperationTime) eq 0 |
| **date** | The date function returns the date part of the date parameter value.<br><br>Example: date(OperationTime) eq 2015-02-25 |
| **time** | The time function returns the time part of the date parameter value.<br><br>Example: time(OperationTime) gt 10:20:00 |
| **now** | The now function returns the current point in time (date and time with time zone) as a date/time value. |

| | |
|---|---|
| | Example: custom_field gt now() |

**OData Constants**

It is possible to use constants for composing OData query (see $filter parameter). Here are formats for constants of different data types:

| Constant Type | Description |
|---|---|
| **Null** | null |
| **Boolean** | true/false |
| **Integer** | -256 or 0.12e1 |
| **Decimal** | 2.33 |
| **Date** | 2015-02-25 |
| **Date/time** | 2015-02-25T02:10:15Z |
| **Time** | 02:20:25.898 |
| **String** | 'some text' (single-quoted text) |
| **GUID** | 01234567-89ab-cdef-0123-456789abcdef |

### *OData Response*

Here is the structure of the OData response:

| Name | Data Type | Description |
|---|---|---|
| Items | | This element contains an array of items which are returned by this query. Every item in the response have the same structure. This structure is dictated by the particular OData service that returns this response. See the service documentation for details about OData item structure.<br><br>For XML formatter: the XML element that wraps OData item is dictated by the particular OData service as well. See the service documentation. |
| TotalCount | Integer | The number of items returned by this query. Note, OData response can be paged and only one page is returned in a single response. TotalCount – is a number of items at all, not just in the current page.<br><br>SPARROW Merchant Public API services restrict the page size to 1000 items by the default. |
| NextLink | String | If the current page is not the last one, this parameter contains URL that can be used to retrieve the next page. |