

INTEGRATION SERVICE

IntegService API Guide

Authentication, permissions, endpoints, sample requests and responses

DBHSOFT · Version 1.0

Nov 18, 2026

Contents

Overview

1. Authentication (getting a token)

2. Permissions

3. Slip export request

4. Example requests

5. Response structure

6. Error codes and common situations

7. Invoices and orders

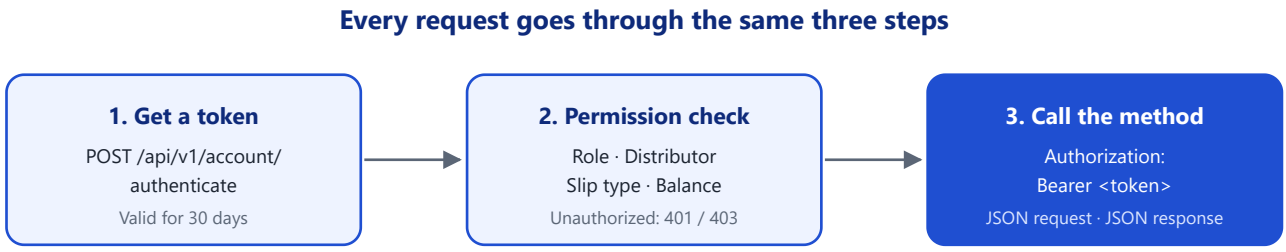
8. Customers, reports and properties

9. Payments

10. Products

11. Stocks and token check

Overview



When the token expires, /api/v1/account/check returns a new one, or call authenticate again.

IntegService is the REST service that exposes slip, customer, payment, product and stock data in SGO to external systems. All methods work with the same token and permission rules (sections 1 and 2); sections 3–6 describe the slip export method, and sections 7–11 describe the other methods.

The slip export service returns invoices, waybills and returns issued in SGO, together with their line items and discounts, as JSON. Usage takes two steps: first obtain a token with your username and password, then query slips with that token.

Group	Address prefix	Section
Getting a token	`{base_url}/api/v1/account/`	1
Slip export (`getinv` , `getinvxml`)	`{base_url}/api/v1/integ/invoices/`	3–6
Invoices and orders	`{base_url}/api/v1/integ/invoices/`	7
Customers, reports, properties	`{base_url}/api/v1/integ/custs/` , `/reports/` , `/props/`	8
Payments	`{base_url}/api/v1/integ/payments/`	9
Products	`{base_url}/api/v1/integ/metas/`	10
Stocks and token check	`{base_url}/api/stocks/` , `/api/v1/account/check`	11

`{base_url}` is the service address provided to you (e.g. separate addresses for the test and production environments). All requests and responses are sent and received with `Content-Type: application/json` .

Test and production environments. The service has separate addresses and separate users for test and production. The database you connect to is determined by the user: a test user sees test data, a production user sees production data. Validate your integration in the test environment first.

Common response structure. Most methods return the same envelope:

Field	Description
`success`	Whether the operation succeeded (`true` / `false`)
`message`	Result message; warnings are appended to the end of this text
`info.company` , `info.country`	Company and country code the token belongs to

Field	Description
`data`	The actual data (object or list)
`count`	Number of records returned; some methods always return `0`, so check the `data` list for the record count

Fields whose value is empty (`null`) are omitted from the response entirely. Dates in responses are in `yyyy-MM-dd HH:mm:ss` format. In requests, send dates as `yyyy-MM-dd` or `yyyy-MM-ddTHH:mm:ss`; the only exceptions are methods whose field description specifies a different format (`gg-aa-yyyy`, i.e. dd-MM-yyyy).

1. Authentication (getting a token)

Obtain a token before querying slips; the token is valid for 30 days and is sent in the `Authorization` header with every request.

POST `{base_url}/api/v1/account/authenticate`

Field	Type	Required	Description
`username`	text	Yes	Service username provided to you
`password`	text	Yes	User password
`company`	text	Yes	Company code (e.g. `DBH-TEST`)
`country`	text	Yes	Country code (e.g. `TR`)

Example request:

```
{
  "username": "servis_kullanici",
  "password": "********",
  "company": "DBH-TEST",
  "country": "TR"
}
```

Example response:

```
{
  "success": true,
  "message": "New Token",
  "info": { "company": "DBH-TEST", "country": "TR" },
  "data": {
    "token": "eyJhbGciOiJIUzI1NiIsInR5cCI6IkpXVCJ9 ... ",
    "expired": "2026-12-18 13:45:00",
    "expires_in": "2591999.5"
  },
  "count": 0
}
```

Response field	Description
`data.token`	Token to use in subsequent requests
`data.expired`	Time the token expires (UTC)
`data.expires_in`	Remaining time, in seconds

Send the token with every subsequent request in this header (the Bearer prefix is optional): `Authorization: Bearer <token>`. Copy the token without quotation marks; if a `"` remains at the start or end, the request is rejected with 401.

2. Permissions

The data a service user can see is restricted by four separate permissions; the permissions are configured for you when your account is set up. If a permission is not defined, no restriction is applied in that area.

Permission	What it restricts	Affected methods	What happens without it
Role (method) permission	Which methods you can call	All methods; `Integration.Invoices.GetInv` for slip export	`403` — `Your role is not authorized.`
Distributor permission	Which distributors' data you can see	`getinv`, `getinvxml`, `getall`, `getOutletListFull`, payment query methods	Unauthorized codes are removed from the result (see below)
Slip type permission	Which slip types you can read	`getall`	`403` — `Your slip type operation is not authorized for: <tip>.`
Balance permission	Customer balance fields	`getOutletListFull`	Balance fields are omitted from the response entirely

How the distributor permission is applied:

- If you do not send `dist_codes`, only slips of the distributors you are authorized for are returned.
- Any unauthorized or inactive codes among those you send are removed from the query and a warning is appended to the response message: `Warning Inactive and unauthorized distributor codes = 001,002`.
- If none of the codes you send are authorized, the response is `204 No Content`.

Balance permission: Without this permission, the `getOutletListFull` response does not contain the `OutletBalance`, `OutletOpenBalance`, `OutletEPoint`, `OutletCheckBalance` and `OutletBondBalance` fields. Other customer fields are unchanged.

If you need changes to your permission definitions, contact your service provider; changes take effect within 1 hour at the latest.

3. Slip export request

`getinv` returns the slips within the given date range that match the filters. Five fields are required; if any is missing, the request is rejected with `400`.

POST `{base_url}/api/v1/integ/invoices/getinv`

Headers: `Authorization: Bearer <token>` and `Content-Type: application/json`

Field	Type	Required	Description
`doc_type`	number	Yes	Slip type, 1–10 (table below)
`start_date`	date	Yes	Start date, inclusive. The old dd-MM-yyyy format is not accepted. Format: `2026-10-01` or `2026-10-01T00:00:00`
`end_date`	date	Yes	End date, **exclusive**. For October 1–31, send `2026-11-01`
`exported`	number list	Yes	Transfer status: `0` = not yet retrieved, `1` = previously retrieved. `[0,1]` for both
`doc_status`	number list	Yes	Slip status: `0` = Canceled, `1` = Approved (printed), `2` = Unapproved, `3` = Pending
`dist_codes`	text list	No	Distributor codes. If empty, all distributors you are authorized for
`outlet_code`	text	No	Customer code
`outlet_integ_code`	text	No	Customer's integration code 1
`outlet_integ_code_2`	text	No	Customer's integration code 2
`vkn`	text	No	Customer's tax number
`doc_no`	text	No	Slip number (e.g. `TST0001`)
`e_inv`	number list	No	E-document type: 0 = e-archive, 1 = e-invoice. Not applied if left empty
`ettn`	text	No	ETTN number of the e-document
`dell_no`	text	No	Shipment number
`limit`	number	No	Maximum number of slips to return. `0` or empty = server upper limit (default 5000)
`update_exported`	true/false	No	Default `true`: returned slips are marked as "exported". Send `false` to retrieve the same slips again

Slip types (`doc_type`):

Value	Slip type	English name
1	Sales invoice	Sales Invoice
2	Purchase invoice	Buy Invoice

Value	Slip type	English name
3	Return invoice from customer (intact goods)	Return Invoice From Outlet
4	Return invoice from customer (damaged goods)	Corrupt Return Invoice From Outlet
5	Return invoice to headquarters (intact goods)	Return Invoice To Headquarter
6	Return invoice to headquarters (damaged goods)	Corrupt Return Invoice To Headquarter
7	Sales waybill	Sales Waybill
8	Shipment waybill	Shipment Waybill
9	Warehouse transfer (intact goods)	Stock Transfer
10	Warehouse transfer (damaged goods)	Corrupt Stock Transfer

> **Warning**
> `update_exported` is on by default. When you pull new slips with `exported: [0]`, those slips are marked and will not be returned by the next `[0]` query. If you only want to look, send `update_exported: false`.

XML output (`getinvxml`). If you call `{base_url}/api/v1/integ/invoices/getinvxml` with the same body, the slips are returned as XML. The XML text is in the `data` field of the response; the fields and rules are the same as for `getinv`. Systems that transfer data to accounting software via XML can use this address.

4. Example requests

Getting a token (curl):

```
curl -X POST "{base_url}/api/v1/account/authenticate" \
  -H "Content-Type: application/json" \
  -d '{"username":"servis_kullanici","password":"********","company":"DBH-TEST","country":"TR"}'
```

Sales invoices not yet retrieved (October 2026, approved):

```
curl -X POST "{base_url}/api/v1/integ/invoices/getinv" \
  -H "Authorization: Bearer <token>" \
  -H "Content-Type: application/json" \
  -d '{
    "doc_type": 1,
    "start_date": "2026-10-01",
    "end_date": "2026-11-01",
    "exported": [0],
    "doc_status": [1],
    "limit": 100
  }'
```

Retrieving a single slip again without marking it:

```
{
  "doc_type": 1,
  "start_date": "2026-10-19",
  "end_date": "2026-10-20",
  "exported": [0, 1],
  "doc_status": [0, 1, 2, 3],
  "dist_codes": ["D001"],
  "doc_no": "TST0001",
  "update_exported": false
}
```

A customer's return invoices (by tax number):

```
{
  "doc_type": 3,
  "start_date": "2026-01-01",
  "end_date": "2027-01-01",
  "exported": [0, 1],
  "doc_status": [1],
  "vkn": "1234567890",
  "update_exported": false
}
```

5. Response structure

The response returns slips under `data.slips`; each slip contains customer information (`outlets`), line items (`items`) and each item's discounts (`discounts`). Amounts are in TRY; unless stated otherwise, they exclude VAT.

Example response (abbreviated, slip TST0001):

```
{
  "success": true,
  "message": "The transaction is successful.",
  "info": { "company": "DBH-TEST", "country": "TR" },
  "data": {
    "slips": [
      {
        "dist_code": "D001",
        "slip_type": "Normal",
        "slip_id": 100001,
        "slip_number": "TST0001",
        "canceled_slip_value": "Printed",
        "slip_date": "2026-10-19 18:05:00",
        "outlet_name": "DBH TEST MÜŞTERİ",
        "store_code": "91",
        "pay_type_code": 3,
        "gross_amount": 2376.0,
        "total_discount": 1171.28,
        "vat": 12.05,
        "gross_total": 1216.77,
        "due_date": "2026-11-09 00:00:00",
        "seller_code": "175",
        "outlets": { "branch": { "central_code": "C0001", "central_name": "DBH TEST MÜŞTERİ",
"tax_no": "1234567890" } },
        "items": [
          {
            "item_no": 2,
            "product_code": "P0002",
            "product_name": "TEST ÜRÜN B 20'li KL",
            "quant": 40.0,
            "product_unit_price": 17.82,
            "row_price": 712.8,
            "price": 361.416,
            "product_real_unit_price": 9.0354,
            "unit": "Koli",
            "vat": 1.0,
            "discounts": [
              { "item_no": 2, "code": "Cust", "sign_discount": "Customer", "type": "Percent",
"discount_rate": 0.03, "discount_val": 21.384, "total_price": 712.8 },
              { "item_no": 2, "code": "EPoint", "sign_discount": "E Point", "type": "Price",
"discount_rate": 0.4773, "discount_val": 330.0, "total_price": 691.416 }
            ]
          }
        ]
      }
    ],
    "failedSlips": []
  },
  "count": 1
}
```


General fields

Field	Description
`success`	Whether the operation succeeded
`message`	Result message; a warning is appended if there are unauthorized distributors or failed slips
`count`	Number of slips returned

Field	Description
`data.slips`	Slips
`data.failedSlips`	Slips that could not be built: `slip_id`, `slip_number`, `dist_code`, `error`. An error in one slip does not block the others

Slip (`slips[]`)

Field	Description
`dist_code`	Distributor code
`slip_type`	Slip kind (`Normal`, `Return`, etc.)
`slip_id`	Unique ID of the slip in the system
`slip_number`	Slip number
`canceled_slip_value`	Slip status: `Canceled` (canceled), `Printed` (approved), `Unconfirmed` (unapproved), `Waiting` (pending)
`slip_date`	Slip date and time
`slip_code`	Alternate number of the slip (e-document number, etc.)
`outlet_name`, `tax_office`, `tax_code`	Customer name, tax office, tax number
`store_code`, `store_name`	Source warehouse
`store_code2`, `store_name2`	Destination warehouse (for transfers)
`pay_type_code`	Payment type code
`gross_amount`	Gross amount (before discounts)
`total_discount`	Total discount amount
`vat`	Total VAT amount
`gross_total`	Total payable for the slip (after discounts, **including VAT**)
`due_date`	Due date
`del_code`, `veh_code`	Delivery person and vehicle code
`seller_code`, `seller_integ_code`	Sales representative code and integration code
`slip_explanation`, `slip_explanation2`	Slip notes
`outlets.head`, `outlets.branch`	Head office and branch customer information (codes, address, channel, tax details)

Item (`items[]`)

Field	Description
`item_no`	Line item sequence number
`product_code`, `product_name`	Product code and name

Field	Description
`quant`	Quantity, in **pieces** (5 cases × 12 = 60)
`unit`	Unit used on the slip (`Koli` = case, `Adet` = piece)
`product_unit_price`	List price per piece
`row_price`	Line gross amount (`quant` × `product_unit_price`)
`price`	Line net amount (after all discounts, excluding VAT)
`product_real_unit_price`	Net price per piece (`price` / `quant`)
`vat`	VAT rate, percent (`1.0` = 1%)
`minimum_value` , `defined_one_value` , `defined_box_value` , `defined_package_value`	The product's unit definitions
`deposit`	`1` if the product carries a deposit
`discounts`	Discounts applied to the item, in order of application

Discount (`discounts[]`)

Field	Description
`code` , `sign_discount`	Discount code and name (`Cust` = customer discount, `EPoint` = e-points discount)
`type`	`Percent` (percentage), `Price` (amount), `Free` (free of charge)
`discount_rate`	Discount rate, as a fraction (`0.03` = 3%)
`discount_val`	Discount amount
`total_price`	Amount this discount was applied to (after previous discounts)

Discounts are applied in sequence: in the example, the 3% customer discount deducts TRY 21.38 from 712.80, and the e-points discount deducts TRY 330.00 from the remaining 691.42; the line net amount becomes TRY 361.42.

Customer (`outlets.head` = head office, `outlets.branch` = branch)

Field	Description
`integration_code1` , `integration_code2`	Customer's integration codes
`central_code` , `central_name`	Customer code and name
`city` , `town` , `adress` , `post_code`	Province, district, address (the field name is returned with this spelling), postal code
`type_code` , `type_name`	Customer type
`channel_code` , `channel_name`	Channel
`national_chain_outlets` , `national_chain_outlets_code` , `national_chain_outlets_name`	National chain information

Field	Description
`telephone`, `fax_number`	Contact details
`tax_no`, `tax_office`	Tax number and tax office
`related_person`	Contact person
`seasonal_outlet`	Seasonal customer
`tax_free`	`1` = tax-exempt customer

Other item fields

Field	Description
`product_type`	Product type
`weight`, `volume`	Product weight and volume (dm^3)
`minimum_unit`, `minimum_value`	Product's unit and sub-unit values
`defined_one_value`, `defined_box_value`, `defined_package_value`	Piece, box and case unit names
`deposit`	`0` = no deposit, `1` = product with deposit

Legacy TXT/XML transfer codes. If you transfer data to accounting software using the legacy TXT/XML format, the field mappings are:

JSON field	Legacy code	JSON field	Legacy code
`dist_code`	DISTCODE	`item_no`	KALEMNO
`slip_type`	FISTIPI	`product_code`	URUNKODU
`slip_id`	FISID	`product_name`	URUNADI
`slip_number`	FISNO	`quant`	MIKTAR
`canceled_slip_value`	IPTAL	`product_unit_price`	BIRIMFIYAT
`slip_date`	TARIH	`row_price`	SATIRTOPLAMI
`slip_code`	MUSKODU	`price`	FIYAT
`outlet_name`	MUSADI	`unit`	BIRIM
`tax_office`	VERGIDAIRESI	`vat`	KDV
`tax_code`	VERGINO	`product_type`	TIPI
`store_code`, `store_code2`	DEPOKODU, DEPOKODU2	`weight`	AGIRLIK
`pay_type_code`	ODEMETIPI	`volume`	HACIM
`gross_amount`	TOPLAMBRUT	`minimum_unit`	UNITE
`total_discount`	TOPLAMISKONTO	`minimum_value`	ALTBIRIM
`vat`	TOPLAMKDV	`defined_one_value`	BIRIM0
`gross_total`	TOPLAMNET	`defined_box_value`	BIRIM1

JSON field	Legacy code	JSON field	Legacy code
`due_date`	VADETARIHI	`defined_package_value`	BIRIM2
`del_code`	DAGITICIKODU	`deposit`	DEPOZITO
`veh_code`	ARACKODU	`discounts[].code`	KODU
`seller_code`	SATISTEMSILCISIKODU	`discounts[].sign_discount`	ADI
`slip_explanation`	FISACIKLAMA	`discounts[].discount_rate`	ORAN
`slip_explanation2`	FISACIKLAMA2	`discounts[].total_price`	TUTAR

6. Error codes and common situations

HTTP code	Message	Cause	What to do
`200`	`The transaction is successful.`	Operation succeeded	—
`200`	`... slip(s) failed, see FailedSlips`	Some slips could not be built	Forward the errors in the `data.failedSlips` list to your service provider
`200`	`Warning Inactive and unauthorized distributor codes = ...`	Some of the distributor codes sent are unauthorized or inactive	Check the codes; slips for the other distributors have been returned
`204`	(empty response)	No slips match the criteria, or none of the distributors sent are authorized	Check the date range and the `exported` and `doc_status` filters
`400`	`Request değerleri boş` (request values are empty)	One of the required fields is missing (`doc_type` , `start_date` , `end_date` , `exported` , `doc_status`)	Add the missing field
`400`	Format error including the field name	A field was sent with the wrong type (e.g. text instead of a number, invalid date)	Send the field using the type in the table
`401`	`Invalid or missing token.`	Token missing, expired or copied incorrectly	Get a new token; make sure there are no quotation marks at the start or end
`403`	`Your role is not authorized.`	The user is not authorized to call this method	Ask your service provider for the `Integration.Invoices.GetInv` permission

Frequently asked questions

- **The same slip does not come back in the second query.** Because `update_exported` is on by default, the slip was marked as "exported" in the first query. To retrieve it again, send `exported: [0, 1]` .
- `quant` **differs from the quantity shown on screen.** `quant` is always in pieces; the screen may show cases. Number of cases = `quant` / pieces per case.
- **Slips on the end date are not returned.** `end_date` is exclusive; send the following day.

7. Invoices and orders

These methods read, create, revise and change the status of slips (invoices, orders, waybills). All of them are `POST`, with the address prefix `{base_url}/api/v1/integ/invoices`.

Method	Address	Function	Required role
`getall`	`/getall`	Returns not-yet-retrieved slips with their line items	`Integration.Invoices.Get`
`bill`	`/bill`	Creates a new slip	`Integration.Invoices.bill`
Revise	`` (prefix, no suffix)	Revises existing slips (list)	`Integration.Invoices.Update`
`cancel`	`/cancel`	Cancels a slip by number and distributor	`Integration.Invoices.Cancel`
`cancelbyid`	`/cancelbyid`	Cancels a slip by ID	`Integration.Invoices.CancelByIDInv`
`confirmbyid`	`/confirmbyid`	Approves a slip	`Integration.Invoices.ConfirmByIDInv`
`unconfirmbyid`	`/unconfirmbyid`	Sets a slip to unapproved	`Integration.Invoices.UnconfirmByIDInv`
`getInvStatus`	`/getInvStatus`	Returns the status and dates of slips	`Integration.Invoices.GetInvStatus`

The `Integration.Invoices` role grants access to all of these methods.

Slip types (`invShapes` and `targetSlipType`; written in upper case, exactly as shown)

Name	Meaning	Customer and seller code
`BUYORDER`	Purchase order placed with headquarters	Not required
`BUYORDEROUTLETDETAILED`	Purchase order with customer details	Required
`ORDER`	Customer sales order	Required
`VOLUMETRICORDER`	Volumetric order	Required
`NCORDER`	Central-type order	Required
`AVAILABILITY`	Availability record	Required
`SALESINVOICE`	Sales invoice	Required
`HOTSALESINVOICE`	Hot sale (van sales) invoice	Required
`BUYINVOICEFROMHEADQUARTER`	Purchase invoice from headquarters	Not required
`BUYINVOICE`	Purchase invoice	Required
`SERVICEINVOICE`	Service invoice	Required

Name	Meaning	Customer and seller code
`RETURNINVOICETOHEADQ`	Return invoice to headquarters (intact goods)	Not required
`CORRUPTRETURNINVOICETOHEADQ`	Return invoice to headquarters (damaged goods)	Not required
`NCRETURNINVOICETOHEADQ`	Central-type return invoice to headquarters	Required
`RETURNINVOICEFROMOUTLET`	Return invoice from customer (intact goods)	Required
`CORRUPTRETURNINVOICEFROMOUTLET`	Return invoice from customer (damaged goods)	Required
`SALESWAYBILL`	Sales waybill	Required
`HOTSALESWAYBILL`	Hot sale (van sales) waybill	Required
`NCWAYBILL`	Central-type waybill	Required
`BUYWAYBILL`	Purchase waybill	Not required
`RETURNWAYBILLFROMOUTLET`	Return waybill from customer (intact goods)	Required
`CORRUPTRETURNWAYBILLFROMOUTLET`	Return waybill from customer (damaged goods)	Required

Slip statuses (`docStatus`): `Canceled` = canceled, `Printed` = numbered/approved, `Unconfirmed` = unapproved, `Waiting` = approved, pending.

7.1 getall

Slip type and distributor permissions apply (section 2). If a slip type you are not authorized for is requested, `403` is returned.

Field	Type	Required	Description
`startDate`	date	Yes	Start, inclusive
`endDate`	date	Yes	End, **exclusive**
`invShapes`	text list	No	Slip types; default `["BUYORDER"]`
`distCode` / `distCodes`	text / text list	No	Distributor codes
`limit`	number	No	Maximum number of slips; empty = 5000
`exported`	true/false	No	If empty, only slips not yet retrieved; `true` = previously retrieved slips
`includeUnconfirmed`	true/false	No	`true` = unapproved slips are also returned
`ignoreSellerLock`	true/false	No	`true` = mobile orders of sellers who have not closed their day are also returned

```
{ "startDate": "2026-11-01", "endDate": "2026-11-18", "invShapes": ["BUYORDER", "ORDER"],
"distCodes": ["1001"], "limit": 500 }
```

For each slip, the response contains `idInv` , `shapeName` , `docStatus` , `invoiceNumber` , `distCode` , `evDate` , `shipmentDate` , customer and branch information (`customerCode` , `customerName` , `customerBranchCode`), `sellerCode` , warehouse information, amounts (`gross` , `discount` , `kdv` , `price`), notes, `items[]` (`metaCode` ,

`amount`, `idUnit`, `unitName`, `unitPrice`, `realPrice`, `discount`, `itemNo`) and `gainedDiscs[]` (discount code, name, amount and percentage).

> **Warning**
> Returned slips are marked as "retrieved" and will not be returned on the next call. To retrieve the same slips again, send `exported: true`.

7.2 bill (creating a slip)

Field	Type	Required	Description
`distCode`	text	Yes	Distributor code
`centralCode`	text	Yes	Head office code
`invoiceNumber`	text	Yes	Slip number; for non-order types it must be unique within the distributor
`evDate`	date	Yes	Slip date
`items`	list	Yes	Line items (below)
`targetSlipType`	text	No	Slip type; default `BUYINVOICEFROMHEADQUARTER`
`customerCode` or `customerIntegCode`	text	Depends on slip type	One of them is required for the "Required" types in the table
`sellerCode` or `sellerIntegCode`	text	Depends on slip type	Same rule as the customer code
`fromStockCode`, `toStockCode`	text	No	Source and destination warehouse; default warehouse if empty
`invoiceNumberOther`	text	No	Second number; default `invoiceNumber`
`docStatus`	text	No	`Printed`, `Unconfirmed` or `Waiting`; determined by type if empty
`paymentType`	text	No	Payment type; default cash
`gross`, `discount`, `kdv`, `price`, `weight`, `volume`	number	No	Header amounts; written as sent, the server does not recalculate them
`explanation`, `explanation2`	text	No	Notes

Leave the `idInv` and `cancelReason` fields empty for a new slip; they are used for revisions. Additional header fields: custDisRat1 (customer discount, %), priceCurr (promissory note amount), rebateDis (signage discount). Weight is in grams, volume in dm³.

Line items (`items[]`)

Field	Type	Required	Description
`metaCode`	text	Yes	Product code
`amount`	number	Yes	Quantity, in the `idUnit` unit; must be greater than 0
`idUnit`	number	Yes*	Unit: `0` piece, `1` box, `2` case; must be defined for the product

Field	Type	Required	Description
`sapUnit`	text	No	Unit short name; if filled, it is used instead of `idUnit`
`itemNo`	text	Yes	Item number; an empty string (`""`) may be sent
`explanation`	text	Yes	Line note; an empty string (`""`) may be sent
`unitPrice`	number	No	Unit price; if 0, the product's list price
`realPrice`	number	No	Net line amount; if empty, calculated from price, quantity and discounts
`unitKdvRat` , `unitKdvRat2`	number	No	VAT rates, as a fraction (`0.18`)
`unitDisRat2` – `unitDisRat4`	number	No	Additional user discount rates
`unitTaxBase` , `tax`	number	No	Unit tax base and tax amount; 0 if empty
`firstAmount`	number	No	Initially entered quantity; default `amount`
`batch`	text	No	Batch; the product is looked up by the code `metaCode-batch`
`disc[]`	list	No	Line discounts (below)

* If `sapUnit` is not sent.

Line discounts (`items[].disc[]`)

Field	Type	Required	Description
`discCode`	text	Yes	Discount code
`discName`	text	Yes	Discount name
`discRate`	number	No	Discount rate, between 0 and 1 (`0.05` = 5%)
`discAmount`	number	No	Discount amount
`discOrderNo`	number	No	Order of application

```
{ "distCode": "1001", "centralCode": "C1001", "invoiceNumber": "F2026000123", "evDate": "2026-11-18 10:00:00", "targetSlipType": "ORDER", "customerCode": "M001", "sellerCode": "S01", "items": [ { "metaCode": "P100", "amount": 10, "idUnit": 2, "unitPrice": 120, "unitKdvRat": 0.18, "itemNo": "10", "disc": [ { "discCode": "D1", "discName": "Campaign", "discRate": 0.05, "discAmount": 60, "discOrderNo": 1 } ] } ] }
```

In the response, `data.idInv` is the ID of the created slip.

HTTP	Message	Cause
`400`	`Target slip type is not found`	Invalid `targetSlipType`
`400`	`The <kod> Distributor Code was not found!`	Distributor does not exist
`400`	`Customer is not found` / `Seller is not found`	Customer or seller code not sent for a type that requires it
`400`	`Duplicate invoice: InvNo = X, Dist Code = Y!`	Slip number already used
`400`	`<kod> Product Code value/s not found!`	Product does not exist

HTTP	Message	Cause
`400`	`The unit of idUnit <n> of the product with code <kod> was not found!`	Unit is not defined for the product
`400`	`DiscRate must be between 0-1`	Invalid discount rate
`400`	`The DistCode field is required.` etc.	Required field missing

7.3 Revise (updating an existing slip)

POST `{base_url}/api/v1/integ/invoices` — the body is a list. Each element carries the same fields as `bill`; in addition, `idInv` (the slip to revise) is required. The old slip is closed and a new slip linked to it is created in its place; the slip type is taken from the old slip. For non-order types, a new `invoiceNumber` must be sent.

```
[ { "idInv": 605090001, "distCode": "1001", "centralCode": "C1001", "invoiceNumber": "SIP-778",
"items": [ { "metaCode": "P100", "amount": 8, "idUnit": 2, "unitPrice": 120, "unitKdvRat": 0.18 } ]
} ]
```

The response is a list with a separate result for each element; `data.idInv` is the ID of the new slip. If any element fails, the response is `400`, but successful elements remain saved. If `idInv` is missing, `IDInv value must be sent!` is returned; if it is not found, `Invoice is not found: <id>` is returned.

7.4 Cancellation and approval

Method	Body	Result
`cancel`	`distCode` (required), `invoiceNumber` (required), `cancelReason` (optional cancellation reason code)	Cancels a slip created through this service
`cancelbyid`	`idInv` (required), `cancelReason` (optional)	Cancels the slip
`confirmbyid`	`idInv` (required)	Approves the slip: `Printed` if it has a number, otherwise `Waiting`
`unconfirmbyid`	`idInv` (required)	Sets the slip to `Unconfirmed`

```
{ "distCode": "1001", "invoiceNumber": "F2026000123", "cancelReason": "IPTAL01" }
```

```
{ "idInv": 605090001 }
```

A successful response message shows the slip that was processed: `The transaction is successful- idInv: 605090001`.

HTTP	Message	Cause
`400`	`No records found with the receipt number X or distributor code Y.`	`cancel`: slip not found
`400`	`The receipt number X of the Y distributor code has been canceled already.`	`cancel`: slip already canceled
`400`	`Slip could not be found!`	No slip found with the ID

HTTP	Message	Cause
`400`	`Unauthorized operation!`	`confirmbyid` / `unconfirmbyid` : slip is canceled or already numbered

7.5 getInvStatus

Returns the status of slips; it does not mark slips as "retrieved".

Field	Type	Required	Description
`startDate` , `endDate`	date	Yes*	Date range, end exclusive
`invShapes`	text list	No	Slip types; default `["BUYORDER"]`
`idInv`	number	No	Single slip
`limit`	number	No	Empty = 5000

* If `idInv` is not sent.

```
{ "startDate": "2026-11-01", "endDate": "2026-11-18", "invShapes": ["BUYORDER"] }
```

For each slip, the response returns `idInv` , `createDate` (creation date), `confirmDate` (last modification date) and `docStatus` .

8. Customers, reports and properties

These methods read customer cards, add or update customers, and return the customer account report and property definitions. All of them are `POST`.

Method	Address	Function	Required role
`GetOutletList`	`/api/v1/integ/custs/GetOutletList`	Customer list (with route and seller)	`Integration.Custs.GetOutletList`
`getOutletListFull`	`/api/v1/integ/custs/getOutletListFull`	Full customer card, properties and routes	`Integration.Custs.GetOutletListFull`
`GetSaleOutletWise`	`/api/v1/integ/custs/GetSaleOutletWise`	Sales quantity by customer and day	`Integration.Custs.GetSaleOutletWise`
`customer`	`/api/v1/integ/custs/customer`	Adds or updates a customer	`Integration.Custs.Customer`
`getCustomerAccountCumulativeReport`	`/api/v1/integ/reports/getCustomerAccountCumulativeReport`	Cumulative customer account report	`Integration.Reports.GetCustomerAccountCumulativeReport`
`list`	`/api/v1/integ/props/list`	Property definitions and values	`Integration.Props.GetProps`

8.1 GetOutletList and getOutletListFull

Field	Type	Required	Description
`distCodes`	text list	No	Distributor codes

```
{ "distCodes": ["194"] }
```

- `GetOutletList` returns code, name, address, GPS, contact person, distributor, route and seller information for each customer. A customer with more than one route is returned on a separate row for each route. If the result is empty, `data: []` is returned.
- `getOutletListFull` returns the full customer card: codes, tax details, type and channel, status (`Active` / `Passive` / `Disabled`), credit information (`creditLimit`, `creditCode`, `creditControlStatus`), payment type, payment term, contact and address, GPS, balances, `customerProperties[]` (property code and value) and `customerRoutes[]` (seller, route, visit day and frequency). If the result is empty, `204` is returned.
- Distributor and balance permissions apply to `getOutletListFull`; for users without the balance permission, the `outletBalance`, `outletOpenBalance`, `outletEPoint`, `outletCheckBalance`, `outletBondBalance` fields are not returned (section 2). The distributor permission is not applied to `GetOutletList`.

8.2 GetSaleOutletWise

Field	Type	Required	Description
`startDate`	date	Yes	Start, inclusive
`finishDate`	date	Yes	End, **exclusive**
`distCodes`	text list	No	Distributor codes

```
{ "startDate": "2026-10-01", "finishDate": "2026-11-01", "distCodes": ["194"] }
```

Each row in the response contains `distributorId`, `outletId`, `monthNo` (short month name, e.g. `Oct`), `saleDate` and `sdCases` (net sales after returns, in cases). If a date is missing, `422` and `Parameter (StartDate) cannot be null.` are returned.

8.3 customer (adding and updating customers)

The customer is looked up by `distributorCode` and `code`. If not found, it is added; if found, only the fields sent are updated. The customer code cannot be changed with this method.

Field	Type	Required	Description
`distributorCode`	text	Yes	Distributor code
`code`	text	Yes	Customer code
`name`	text	For new customers	Legal name
`shortName`	text	For new customers	Short name
`customerType`	text	For new customers	Customer type name (same as the name defined in the system)
`creditCode`	text	For new customers	`Advance` = prepaid, `Cash` = cash, `Deferred` = on credit terms
`paymentType`	text	For new customers	`Cash`, `Credit Card`, `Bank Transfer`, `Check`, `Bond`, `Open Account`
`creditStatus`	text	No	`Unlimited`, `Warning`, `Error`
`creditLimit`, `dueDays`	number	No	Credit limit, payment term in days
`mainCustomerCode`	text	No	Head office customer it belongs to
`integrationCode`, `integrationCode2`	text	No	Integration codes
`sellerDistributorCode`, `sellerCode`	text	No	Sales representative of the card
`taxOffice`, `taxNo`, `idNo`, `taxTypeName`	text	No	Tax office, tax number, national ID number, tax class

Field	Type	Required	Description
`countryName`, `cityName`, `townName`	text	No	Address, using the names defined in the system
`address1`, `address2`, `postCode`	text	No	Address
`workTel1`, `mobileTel1`, `email` etc.	text	No	Contact details
`gpsCoordX`, `gpsCoordY`	number	No	Longitude, latitude
`passive`, `dontSell`, `disabled`	true/false	No	Passive, do not sell, disabled

Values must be written exactly, including case and spaces; an unrecognized value falls back to the default (`Cash`, `Advance`, `Unlimited`). For a new customer, date fields that are not sent take today's date.

```
{ "distributorCode": "194", "code": "C-1001", "name": "Örnek Gıda Ltd.", "shortName": "Örnek Gıda", "customerType": "Bakkal", "creditCode": "Deferred", "paymentType": "Open Account", "creditLimit": 50000, "dueDays": 30, "cityName": "İstanbul", "townName": "Kadıköy", "sellerDistributorCode": "194", "sellerCode": "001" }
```

The response returns the record that was sent. For a new customer with a missing field, `400` is returned with `You must send Credit Code`, `You must send Payment Type`, `You must send Customer Type`, `You must send Name for Customer` or `You must send Short Name for Customer`.

8.4 getCustomerAccountCumulativeReport

Field names use underscores.

Field	Type	Required	Description
`start_date`	text	Yes	Period start, `gg-aa-yyyy` (dd-MM-yyyy), inclusive
`end_date`	text	Yes	Period end, `gg-aa-yyyy` (dd-MM-yyyy), exclusive
`limit`	number	Yes	Number of accounts to return (maximum 50000); if not sent, the account list comes back empty
`dist_codes`	text list	Yes	Distributor codes
`consider_chain_stores`	number	Yes	`0` = per branch, `1` = aggregate under the chain head office
`outlet_codes`	text list	No	Customer codes
`outlet_shape_codes`, `outlet_shape_names`	text list	No	Customer type
`outlet_passives`	number list	No	`1` = passive customers
`seller_codes`, `seller_integ_codes`	text list	No	Sales representative
`route_days`	number list	No	Route day, `0` = Monday
`acc_status`	number	No	`0` = no debt, `1` = in debt

Field	Type	Required	Description
`max_rec_dept`	true/false	No	`true` : last collection and last invoice dates are also calculated

```
{ "dist_codes": ["194"], "start_date": "01-10-2026", "end_date": "01-11-2026", "limit": 1000, "consider_chain_stores": 0, "acc_status": 1, "max_rec_dept": true }
```

The response contains five lists: `customers` (customer card), `customerDists` (distributors), `customerAccounts` (`revolve` carried over, `debt` period debit, `receivable` period credit, `balance` balance, `sales` turnover, `credit_limit` , `risk` , `max_receivable_date` , `max_dept_date`), `customerSellers` and `sellers` .

8.5 props/list

Keys in the body are case-sensitive; they must be written exactly as shown below.

Field	Type	Required	Description
`distcode`	text	Yes	An active distributor code
`subject`	text	Yes	Property subject, e.g. `Cust` (customer), `Meta` (product)

```
{ "distcode": "194", "subject": "Cust" }
```

The response is a list without an envelope: for each property, `propCode` , `propName` and its possible values (`items[]`). If a key is missing, `400` is returned with `Discode Value Is Required!` or `subject Value Is Required!` .

9. Payments

The payment methods read and write collection, payment, service invoice and balance adjustment records. All methods are `POST`, with the address prefix `{base_url}/api/v1/integ/payments/`.

Method	Function	Required role
`getPay`	Returns collection/payment slips with their line items (legacy format, underscore field names)	`Integration.Payments.GetPayments`
`GetPayment`	Returns collection/payment slips with their line items	`Integration.Payments.GetPayment`
`GetServiceInvoice`	Returns service invoices	`Integration.Payments.GetServiceInvoice`
`BalanceEqualizing`	Returns balance adjustment records	`Integration.Payments.BalanceEqualizing`
`add`	Adds a collection/payment slip	`Integration.Payments.Add`
`Update`	Updates a collection/payment slip	`Integration.Payments.Update`
`Cancel`	Cancels a slip	`Integration.Payments.Cancel`
`AddServiceInvoice`	Adds a service invoice	`Integration.Payments.AddServiceInvoice`
`UpdateServiceInvoice`	Updates a service invoice	`Integration.Payments.UpdateServiceInvoice`

The `Integration.Payments` role grants access to all of these methods. The distributor permission applies to the read methods (section 2).

Common codes

Field	Values
`direction`	`Collection` = collection, `Payment` = payment
`paymentType`	`Cash` = cash, `CreditCard` = credit card, `BankTransfer` = bank transfer, `Check` = check, `Bond` = promissory note

Values are case-sensitive; an unrecognized value is treated as `Collection` and `Cash`.

Read methods mark records as "exported". `getPay`, `GetPayment`, `GetServiceInvoice` and `BalanceEqualizing` mark the records they return. When you call again with `exported: false`, the next page is returned. Records that are updated or canceled revert to "not exported" and are returned again.

9.1 getPay

Field	Type	Required	Description
`start_date`	text	Yes	Start, format `gg-aa-yyyy` (dd-MM-yyyy, e.g. `01-01-2026`), inclusive
`end_date`	text	Yes	End, format `gg-aa-yyyy` (dd-MM-yyyy), **exclusive**
`exported`	number list	Yes	`0` = not retrieved, `1` = retrieved
`doc_status`	number list	Yes	`0` = canceled, `1` and `3` = valid
`dist_codes`	text list	No	Distributor codes
`outlet_code`, `outlet_integ_code`, `outlet_integ_code_2`, `vkn`	text	No	Customer filters
`doc_no`	text	No	Linked invoice number
`receipt_no`	text	No	Receipt number
`is_service`	number	No	`0` = no filter, `1` = service, `2` = balance adjustment
`doc_type`	number	No	`1` cash, `2` credit card, `3` bank transfer, `4` check, `5` promissory note; if given, only items of that type are returned
`seller_code`, `seller_integ_code`	text	No	Sales representative
`limit`	number	No	Maximum number of records; empty = server limit (5000)

```
{ "dist_codes": ["131"], "start_date": "01-01-2026", "end_date": "01-02-2026", "exported": [0], "doc_status": [1, 3], "limit": 100 }
```

For each record, the response contains `pay_id`, `dist_code`, `outlet_code`, `outlet_name`, `collection_date`, `receipt_number`, `dir` (`COLLECTION` / `PAYMENT`), `shape` (`NORMAL` / `CHECK` / `BONDS`), `price`, `seller_code`, `is_service` and line items (`pay_items[]`: `doc_type`, `pay_type_name`, `amount`, `document_no`, `due_date`, `bank`, `branch`).

9.2 GetPayment, GetServiceInvoice, BalanceEqualizing

All three use the same request fields.

Field	Type	Required	Description
`startDate`	date	Yes*	Start, inclusive (`2026-01-01`)
`endDate`	date	Yes*	End, **exclusive**
`direction`	text	Yes (`GetPayment`, `GetServiceInvoice`)	`Collection` / `Payment`. `GetServiceInvoice` always returns collections, but the field must still be sent
`exported`	true/false	No	Default `false`: only records not yet retrieved. `true`: only retrieved records

Field	Type	Required	Description
`distCodes`	text list	No	Distributor codes
`idPay`	number	No	Fetches a single record; if given, the date and `exported` filters are not used
`limit`	number	No	Maximum number of records; empty = 5000

* Required if `idPay` is not sent.

```
{ "distCodes": ["131"], "startDate": "2026-01-01", "endDate": "2026-02-01", "exported": false, "direction": "Collection", "limit": 500 }
```

Method	Main fields in the response
`GetPayment`	`idPay`, `distCode`, `evDate`, `receiptNo`, `sellerCode`, `customerCode`, `dir` (`1` collection, `2` payment), `idInv`, `docStatus`, `isCanceled`, `items[]` (`paymentType`, `price`, `documentNo`, `dueDate`, `bank`, `bankOffice`)
`GetServiceInvoice`	`idPay`, `distCode`, `evDate`, `invoiceNumber`, `customerCode`, `price`, `serviceType`
`BalanceEqualizing`	`idPay`, `distCode`, `evDate`, `invoiceNumber`, `customerCode`, `price`, `direction` (`Decrease` / `Increase`), `type` (e.g. `Decrease Balance`)

9.3 add and Update (writing collections/payments)

Field	Type	Required	Description
`idPay`	number	Only `Update`	Slip to update. Ignored in `add`; the new ID is returned in the response
`distCode`	text	Yes	Distributor code
`customerCode`	text	Yes	Customer code
`evDate`	date	Yes	Document date
`items`	list	Yes	Line items; the slip amount is the sum of the items
`items[].paymentType`	text	No	Payment type (Common codes)
`items[].price`	number	No	Item amount
`items[].explanation`	text	No	Item note
`sellerCode`	text	No	Sales representative
`receiptNumber`	text	No	Receipt number
`direction`	text	No	Default `Collection`
`explanation`	text	No	Note
`idInv`	number	No	Linked invoice ID

```
{ "distCode": "131", "customerCode": "M001", "sellerCode": "S01", "evDate": "2026-01-15",
"receiptNumber": "TH-12", "direction": "Collection", "items": [ { "paymentType": "Cash", "price":
1000 }, { "paymentType": "CreditCard", "price": 500 } ] }
```

The response returns the record you sent inside `data`; for `add`, `idPay` is the ID of the new slip. `Update` deletes and rewrites the line items, and makes a canceled slip valid again.

9.4 Cancel

Field	Type	Required	Description
`idPay`	number	Yes	Slip to cancel

```
{ "idPay": 5013 }
```

A canceled slip is returned again by the read methods with `isCanceled: true`.

9.5 AddServiceInvoice and UpdateServiceInvoice

Field	Type	Required	Description
`idPay`	number	Only `UpdateServiceInvoice`	Record to update
`distCode`	text	Yes	Distributor code
`customerCode`	text	Yes	Customer code
`evDate`	date	Yes	Date
`serviceType`	text	Yes	Service name; must exactly match a service name defined in the system
`price`	number	No	Amount
`invoiceNumber`	text	No	Invoice number; if filled, the record is created as approved, otherwise as pending
`sellerCode`, `explanation`, `idInv`	—	No	Representative, note, linked invoice

```
{ "distCode": "131", "customerCode": "M001", "evDate": "2026-01-20", "invoiceNumber": "HZ-1",
"price": 250, "serviceType": "Raf Kirası" }
```

Errors: The read methods use the codes in section 6. On error, the write methods (`add`, `Update`, `Cancel`, `AddServiceInvoice`, `UpdateServiceInvoice`) return `400` with only the error text; if `items` is not sent, the text is `Object reference not set to an instance of an object.`

10. Products

The product methods read and write the product card, its units, prices, properties and translations. Address prefix: `{base_url}/api/v1/integ/metas`.

Method	HTTP method and address	Function	Required role
Product list	POST `/getmetalist`	All products, with their units	`Integration.Metas.Post`
Single product	GET `/{idMeta}`	A single product by product ID	`Integration.Metas.GetById`
Product units	POST `/metaunits`	Units of a product	`Integration.Metas.GetMetaUnits`
Save product	POST `` (prefix, no suffix)	Adds or updates products (list)	`Integration.Metas.Save`
Delete product	DELETE `/{idMeta}`	Permanently deletes a product	`Integration.Metas.Delete`
Deactivate / activate	POST `/delist`	Deactivates or activates a product	`Integration.Metas.DeList`
Read prices	GET `/prices/{idDist}/{idMeta}`	Price records of a product	`Integration.Metas.GetMetaPrices`
Write prices	POST `/prices`	Adds or updates prices (list)	`Integration.Metas.AddMetaPrices`
Read properties	POST `/getprop`	Properties of a product	`Integration.Metas.GetMetaProps`
Write property	POST `/addprop`	Adds a property value to a product	`Integration.Metas.AddMetaProps`
Write translation	POST `/translate`	Product name translation (list)	`Integration.Metas.Translate`
Suggested order	POST `/suggestedOrder`	Uploads a suggested product list for a customer	`Integration.Metas.SuggestedOrder`

The `Integration.Metas` role grants access to all of these methods. If a product code includes a batch, it is written in the form `CODE-BATCH`, e.g. `P0001-ABC`. Language codes: `en`, `tr`, `ru`, `ar`, `id`, `ro`, `de`, `fr`, `dk`, `vn`, `th`, `my`, `fa`.

10.1 Reading products (getmetalist, single product, metaunits)

`getmetalist` body:

Field	Type	Required	Description
`isoCode`	text	No	Language of product names; default `en`. The key must be written exactly as `isoCode`

```
{ "isoCode": "tr" }
```

For each product, the response returns `idMeta`, `metaCode`, `genericCode`, `batch`, `name`, `shortName`, `kdvRatio`, `weight`, `volume`, `barcode`, `disabled`, `price`, `buyPrice`, `unitCase`, `dontSell` and `units[]` (`idUnit`, `amount`, `name`, `shortName`). Inactive products are listed too. The single product method returns the same fields without units; if the product is not found, `data` is not returned.

`metaunits` body:

Field	Type	Required	Description
`distCode`	text	Yes	Distributor code
`metaCode`	text	Yes	Product code

```
{ "distCode": "TEST", "metaCode": "P0001" }
```

10.2 Saving products

The body is a list. The product is looked up in the distributor by `metaCode` (and `batch`, if any); if not found, it is added, and if found, it is updated. **Fields not sent in an update are reset**, so send all fields of the product.

Field	Type	Required	Description
`distCode`	text	Yes	Distributor code
`metaCode`	text	Yes	Product code
`batch`	text	No	Batch; if filled, the product code becomes `metaCode-batch`
`name`, `shortName`	text	No	Product name and short name
`idMetaShape`	number	No	`1` normal, `2` promotional, `3` consignment, `4` competitor
`idMinUnit`, `idDefUnit`	number	No	Smallest and default unit: `0` piece, `1` box, `2` case
`kdvRatio`, `kdvRatio2`	number	No	VAT rate, as a fraction (`0.20` = 20%)
`weight`, `volume`, `palette`, `unitCase`, `subUnit`, `fullCase`, `alcoholRate`, `epoint`	number	No	Physical and logistics information
`barcode`	text	No	Barcode
`noStock`, `isFinancial`, `dontSell`, `fractional`, `dayMetaControl`	number	No	`1` = yes, `0` = no
`dayMetaLife`	number	No	Shelf life, in days
`disabled`	true/false	No	Update only: `true` = inactive
`units[]`	list	No	Units: `idUnit`, `amount` (quantity in the smallest unit), `name`, `shortName`
`dic[]`	list	No	Name translations: `lang`, `value`

Price fields are not written by this method; use `/prices` for prices.

```
[{ "distCode": "TEST", "metaCode": "P0001", "idMetaShape": 1, "idMinUnit": 0, "idDefUnit": 2,
"name": "Ürün A", "shortName": "ÜrnA", "kdvRatio": 0.2, "barcode": "869000000001", "dic": [{ "lang":
"en", "value": "Product A" }], "units": [{ "idUnit": 0, "amount": 1, "name": "Adet", "shortName":
"AD" }, { "idUnit": 2, "amount": 12, "name": "Koli", "shortName": "KL" }] }]
```

The response contains a separate result for each product; an error in one product does not block the others. If at least one product fails, the response is `400` and the failed product is marked with `success: false`.

10.3 Deleting and deactivating

- **Delete** (`DELETE /{idMeta}`) permanently deletes the product together with its stock and unit records; it cannot be undone. To only remove the product from sale, use `delist`.
- The **delist** body is a list:

Field	Type	Required	Description
`distCode`	text	Yes	Distributor code
`metaCode`	text	Yes	Product code (`CODE-BATCH` for batch products)
`disabled`	true/false	No	`true` = deactivate, `false` = activate

```
[{ "distCode": "TEST", "metaCode": "P0001", "disabled": true }]
```

10.4 Prices

`GET /prices/{idDist}/{idMeta}` returns the product's price records: `firstDate`, `lastDate`, `price`, `buyPrice`, `normalReturnPrice`, `corruptReturnPrice`, `advicePrice`, `ruleNo`.

The `POST /prices` body is a list:

Field	Type	Required	Description
`distCode`	text	Yes	Distributor code
`metaCode`	text	Yes	Product code (without batch)
`firstDate`	date	Yes	First day the price is valid
`buyPrice`	number	Yes	Price; written as both the sales and the purchase price
`normalReturnPrice`, `corruptReturnPrice`, `advicePrice`	number	No	Intact-goods return, damaged-goods return, recommended price
`batch`	text	No	If filled, the price is written to the batch product; if the batch does not exist, it is created from the main product
`degree`	number	No	Price level
`customerFilterShape`, `customerFilterValue`	text	Together	Restricts the price to a customer group (e.g. `DISTRIBUTOR GROUP` and the group code). If both are empty, it is a general price

```
[{ "distCode": "DBH-TEST", "metaCode": "P0004", "firstDate": "2026-09-01", "buyPrice": 100,
"normalReturnPrice": 100, "corruptReturnPrice": 90, "advicePrice": 120, "degree": 1 }]
```

The server calculates the end date: when a new price starts, the previous price of the same product is closed, and the latest price remains open-ended. If you send more than one period for the same product in a list, they are ordered by `firstDate`. If an element fails, processing stops; elements before it remain saved.

10.5 Properties and translations

The `getprop` body is `{ "distCode", "metaCode" }`. The `addprop` body:

Field	Type	Required	Description
`distCode`	text	Yes	Distributor code
`metaCode`	text	Yes	Product code
`props[].propCode`	text	Yes	Product property code defined in the system
`props[].valueCode`	text	Yes	Value code, at most 50 characters
`props[].name`	text	No	Name of the value

```
{ "distCode": "TEST", "metaCode": "P0003", "props": [{ "propCode": "Category", "valueCode": "41",
"name": "Atıştırmalık" }] }
```

The value is also written to the product's batch variants. The `translate` body is a list: `distCode`, `metaCode`, `lang` and `value` (translation text) are required.

```
[{ "distCode": "TEST", "metaCode": "P0001", "lang": "en", "value": "Product A" }]
```

10.6 Suggested order (suggestedOrder)

The body is a list. The previous suggestion list for the same customer and type is deleted and the new one is written.

Field	Type	Required	Description
`distCode`	text	Yes	Distributor code
`outletCode`	text	Yes	Customer code
`typeCode`	text	Yes	`1` core, `2` strategic, `3` opportunity
`weekStartDate`, `weekEndDate`	date	Yes	Week for which the list is valid
`metaCodeTargets[].metaCode`	text	Yes	Product code, at most 10 characters
`metaCodeTargets[].targetCount`	number	No	Suggested quantity
`metaCodeTargets[].targetUnit`	number	No	Unit: `0` piece, `1` box, `2` case

```
[{ "distCode": "D001", "outletCode": "M0002", "typeCode": "1", "weekStartDate": "2026-11-23",
"weekEndDate": "2026-11-29", "metaCodeTargets": [{ "metaCode": "P0005", "targetCount": 5,
"targetUnit": 1 }] }]
```

The response is always `200`. `message` gives the number of successful and failed rows; `data` lists the codes that were not found (e.g. `"999: meta code bulunamadı!"`, i.e. meta code not found).

Customer-specific product integration methods are outside the scope of this guide.

11. Stocks and token check

The stock methods use a different address prefix: `{base_url}/api/stocks/` (without `/v1/integ`).

Method	HTTP method	Function	Required role
`PastStockMonitoringReportByProduct`	POST	A distributor's historical stock, broken down by product and warehouse	`Integration.Stocks.GetPastStockMonitoringReportByProduct`
`GetWarehouseStock`	POST	Product stocks per warehouse (case/box/piece)	`Integration.Stocks.GetWarehouseStock`
`GetDistSales`	POST	Monthly sales and return quantities of all distributors	`Integration.Stocks.GetDistSales`
`GetDistStock`	GET	Current stocks of all distributors	`Integration.Stocks.GetDistStock`
`account/check`	POST	Validates the token and renews it if expired	No role required

11.1 PastStockMonitoringReportByProduct

Field names use underscores. Stock is calculated from movements **before** the given day; that day's movements are not included.

Field	Type	Required	Description
`dist_code`	text	Yes	A single distributor code; must be authorized and active
`stock_day`	text	Yes	`gg-aa-yyyy`, `gg/aa/yyyy` or `gg.aa.yyyy` (day-month-year)
`meta_disabled`	number list	Yes	`0` = active products, `1` = disabled products; `[]` = all
`limit`	number	No	Maximum number of rows; empty = 5000
`product_owner`	text	No	`HO` = headquarters products only; default `D` = no filter
`stock_codes`, `meta_codes`, `seller_codes`	text list	No	Warehouse, product and sales rep filters

```
{ "dist_code": "228", "stock_day": "03-11-2026", "limit": 1000, "meta_disabled": [0],
"product_owner": "HO", "stock_codes": ["91"] }
```

In the response, `data.meta_stocks[]` contains, for each product, `meta` (`meta_code`, `meta_name`) and `stocks[]` (`store_code`, `store_name`, `seller_code`, `package` cases, `piece` total pieces, `liter`, `kilogram`, `sell_price`

and `buy_price` total amounts). `400` is returned in cases such as an invalid or unauthorized distributor or a missing date.

11.2 GetWarehouseStock

Field	Type	Required	Description
`date`	date	No	If given, stock before that date; otherwise current stock
`distCodes`	text list	No	Distributor codes
`productCodes`	text list	No	Product codes
`showOnlyAvailableStock`	true/false	No	`true` = non-zero stocks only

```
{ "date": "2026-11-01", "distCodes": ["228"], "productCodes": ["P0006"], "showOnlyAvailableStock": true }
```

Each row in the response contains `distributorCode`, `warehouseCode`, `warehouseName`, `productCode`, `productName`, `evDate`, `unit3Amount` / `unit3Name` (case), `unit2Amount` / `unit2Name` (box), `unit1Amount` / `unit1Name` (piece), `volume` and `weight`.

11.3 GetDistSales and GetDistStock

`GetDistSales` body:

Field	Type	Required	Description
`startDate`	date	Yes	Start, inclusive
`endDate`	date	Yes	End, inclusive; if no time is given, 00:00 of that day is used (send `2026-10-31T23:59:59`)

```
{ "startDate": "2026-10-01", "endDate": "2026-10-31T23:59:59" }
```

`GetDistStock` takes no parameters (`GET`). Both methods return the same fields: `material` (product code), `customer` (distributor code), `salesorg`, `distR_CHAN`, `date` (first day of the month), `koli` (cases), `adet` (pieces not making up a full case) and `kg`. Sales are summed as positive and returns as negative. These two methods return all distributors; the distributor permission is not applied.

11.4 account/check

POST `{base_url}/api/v1/account/check` — there is no body; the token is sent in the `Authorization` header.

Status	Response
Token valid	`success: true`, `message: "Valid Token"`, the same token and expiry time in `data`
Token expired	`message: "New Token"` and a **new token** in `data`. In this case `success` is returned as `false`; use the new token
Token invalid	`400`, `message: "Invalid token."` — call `authenticate` again

The service's translation and log methods (`/translate` , `/logs` , `/api/v2/log`) are for internal systems and are outside the scope of this guide. The `stockposition` and `stockmovement` methods are specific to a particular customer integration; if you need them, ask your service provider for separate documentation.