1. Introduction

Overview: The API is designed to integrate third-party software with your point-of-sale (POS) equipment. The main purpose of the API is to automate and simplify the transaction process. While the device can be operated manually, speed is crucial for most retail stores. Using a small POS device manually can be inconvenient, which is why the API is provided to allow your software to integrate with the POS equipment, enhancing the efficiency of the retail operation.

Connecting to the API: To connect to the API, the following steps must be completed:

1. Connect the device to WiFi.
2. Launch the Caspos Mobile software and log in under the name "**CashierUserName**".
3. Go to settings and select the "**Enable Integration Service**" option.
4. When enabling the service, you will be prompted for a password. Enter the password "**password0101**," where "**0101**" represents the last four digits of the device's serial number.
5. After successfully enabling the service, the screen of the device will display the software version, battery status, and the device's IP address.
6. Use the IP address and local static port **5544** to connect to the API via the HTTP protocol.
7. Additionally, to properly display Azerbaijani fonts, use the UTF-8 encoding: **application/json; charset=utf-8.**
8. Required numerical format: **Quantity: 0.000, SalePrice: 0.00, PurchasePrice: 0.00.**

Exiting the Caspos Mobile Program:

1. Press and hold the power button.
2. In the upper right corner, click the "Exit Kiosk" button.
3. Enter the password "3223" (This password is required to exit the program and configure WiFi settings).

These steps ensure a successful connection and access to the API for performing the necessary operations.

Basics of Working with the API

- **Communication Protocol:** The API uses the HTTP protocol for data exchange.
- **Request and Response Format:**

Request Template: How to correctly compose a request to your API:

{"operation": "operation_name", "data": {"param1": "value1","param2": "value2"}}

Response Template: An example of the response format that the API returns to requests.

{ "code": 0,"message": "","data": {"param1": "value1", "param2": "value2"}}

Parameter Descriptions:

- **documentUUID**: A unique identifier for each transaction.
- **cashPayment**: Cash payment. If the amount entered in this field exceeds the total cost of goods, the program will automatically calculate the change. For example, if a product costs 10 AZN and 100 AZN is entered, the program will display: "Cash: 10 AZN, Given: 100 AZN, Change: 90 AZN."
- **creditPayment**: The amount processed as credit.
- **depositPayment**: Prepayment or deposit.
- **cardPayment**: Payment by bank card.
- **bonusPayment**: Payment with bonuses.
- **items**: List of items in the receipt.
    - **name**: Name of the item.
    - **code**: Barcode of the item.
    - **quantity**: Quantity of the item.
    - **salePrice**: Price per unit of the item.
    - **sum**: If the `sum` field is added to an item line, the price is automatically calculated by the application. This field is used to enter the total amount for an item line; in some cases, the software may round incorrectly. This field is optional.
    - **purchasePrice**: Cost price of the item.
    - **codeType**: Type of barcode.
    - **quantityType**: Type of unit of measurement.
    - **vatType**: Type of VAT.
    - **discountAmount**: Discount on the item.
    - **markingCodes**: An array of markings for the item.
- **clientName**: Client's name (optional).
- **clientTotalBonus**: Total bonus amount of the client.
- **clientEarnedBonus**: Bonuses earned for the purchase.
- **clientBonusCardNumber**: Client's bonus card number.
- **cashierName**: Full name of the cashier.
- **note**: Note (description) - an additional line to be printed at the end of the receipt.
- **rrn**: RRN number for the bank transaction.
- **transactionId**: ID of the bank transaction.
- **transactionNumber**: Number of the bank transaction.
- **approvalCode**: Approval code of the bank transaction.
- **currency**: Currency (always sent as AZN).
- **operation**: Type of operation performed.
- **creditContract**: Credit contract number.
- **creditPayer**: Name of the credit payer.
- **residue**: Remaining balance on the credit after payment.
- **parentDocumentId**: Identifier of the parent sales document.
- **parentDocumentSum**: Total amount of the receipt for the return.
- **document_id**: Document identifier.
- **document_number**: Document number.
- **shift_document_number**: Shift document number.
- **short_document_id**: Short document identifier.
- **totalSum**: Total amount of the receipt.

Login

To gain access to the API, an authorization request must be made. Upon successful authorization, the device generates and returns an access token. However, it is not necessary to include the token in subsequent requests, as it is automatically stored in the device's memory, allowing further operations to be performed without the need to pass the token.

Example Request:

"operation": "toLogin", "username": "username","password": "password"}

Example Response:

{ "data": { "access_token": "+jUvyzDwh7MSoqA7Q6tDNA=="} "code": "0","message": "Success operation"}

After receiving the access token, you can proceed with other API requests without needing to include the token in every request.

Logout

The logout request terminates the current user session by removing the stored token from the device's memory. After executing this request, access to other API operations will be restricted until reauthorization is performed.

Example Request:

{"operation": "toLogout","username": "username","password": "password"}

Example Response:

{ "code": "0","message": "Success operation"}

After a successful logout request, the user will be deauthorized, and a new login will be required to continue working with the API.

GetInfo

The `getInfo` request is used to obtain information about the current state of the device and to check its operational status. It returns detailed data about the device and its condition, allowing you to ensure that the device is functioning correctly and is properly connected.

Example Request:

{ "operation": "**getInfo**", "username": "username", "password": "password"}

Example Response:


```
{
    "data": {
        "application_version": "Versiya: 1.0.7.0”
        "cashregister_model": "SUNMI V1S",
        "cashbox_tax_number": "test_00086",
        "cashbox_factory_number": "00320029594B500B20373357",
        "cashregister_factory_number": "VS4319W60010",
        "company_name": "\"CASPOS\" MƏHDUD MƏSULİYYƏTLİ CƏMİYYƏTİ",
        "company_tax_number": "554534053",
        "last_doc_number": 10520,
        "last_online_time": "09.10.2023 09:53:10",
        "object_address": "AZ1007 BAKI ŞƏHƏRİ Jafar Khandan 50.",
        "object_name": "OFİS",
        "object_tax_number": "554534053-11001",
        "qr_code_url": "https://monitoring.e-kassa.gov.az/#/index?doc=",
        "state": "ACTIVE",
        "token_version": "2.35.0"
    },
    "code": "0",
    "message": "Success operation"
}
```


Property name	Type	Description
cashregister_model	String	Cashbox model
cashbox_tax_number	String	Cashbox Tax number
cashbox_factory_number	String	Cashbox Factory number
cashregister_factory_number	String	Cash Register Factory Number
company_name	String	Company name
company_tax_number	String	Company Tax number (VOEN)
last_online_time	DateTime	Last online time of token
last_doc_number	Float	Last document number
object_address	String	Object address
object_name	String	Object name
object_tax_number	String	Object tax number
qr_code_url	String	Url of Tax Service
state	String	State of Token
application_version	String	Application version
token_version	String	VN Token version

Get Shift Status

The Get Shift Status request is used to obtain the current status of a shift on the cash register equipment. The shift can either be open or closed.

Example Request:

```
{
   "operation": "getShiftStatus",
   "username": "username",
   "password": "password"
}
```

Property name	Type	Description
operation	String	Operation name

Example Response:

```
{
   "data": {
      "shift_open": true,
      "shift_open_time": "11.06.2021 12:07:16"
   },
   "code": "0",
   "message": "Success operation"
}
```

Property name	Type	Description
shiftOpen	Boolean	State of Shift
shiftOpenTime	DateTime	Opening time of shift

Note: This request is used to obtain the shift status on the cash register equipment, including information on whether the shift is open and the time it was opened.

Open Shift

The `Open Shift` request is used to open a shift on the cash register equipment. If a shift is already open, the API will return an error message. If the shift is closed, it will be opened, and a successful response will be returned. The `sum` parameter allows you to specify the amount of the initial deposit that will be placed in the cash drawer when the shift is opened. This is useful for tracking the initial amount in the cash register.

Example Request:

```
{
    "data": {
        "sum": 1.0 //Open shift deposit sum
    },
    "operation": "openShift",  //Operation Type
    "cashierName": "Yeni Kassir", //Cashier full name
    "username": "username",
    "password": "password"
}
```

Property name	Type	Description
operation	String	Operation name
sum	Float	Deposit amount
cashierName	string	Cashier name

Example Response:

```
{
   "code": "0",
   "message": "Success operation"
}
```

Note: This request is used to open a shift on the cash register equipment. If the shift is already open, the system will return an error message.

Close Shift

The `Close Shift` request is used to close the operational day on the cash register equipment. It includes the following parameters:

- cashierName: Displays the cashier's name on the cash register receipt.

- DocumentUUID: Controls the re-sending of transactions. The unique `DocumentUUID` is managed by the integrator: if this UUID has not been used before, the operation will be executed. If the UUID is already present, the system will return the result of the previous operation along with a message about the duplicate transaction.

This feature is useful in cases where there is a loss of connection during the transaction and no response is received.

Example Request:

```
{
   "data": {
      "documentUUID": "7c3168f3-1749-4fe4-976c-6ef195d10401",
      "cashierName": "Yeni Kassir"
   },
   "operation": "closeShift",
   "username": "username",
   "password": "password"
}
```

Property name	Type	Description
operation	String	Operation name
cashierName	String	Cashier Name
documentUUID	String	Unical Document ID

Example Response:

The request returns information about the shift closure, including detailed data on the transactions performed during the shift.

```
{
    "data": {
        "currencies": [
            {
                "correctionBonusSum": 0.0,
                "correctionCashSum": 0.0,
                "correctionCashlessSum": 0.0,
                "correctionCount": 0,
                "correctionCreditSum": 0.0,
                "correctionPrepaymentSum": 0.0,
                "correctionSum": 0.0,
                "correctionVatAmounts": [],
                "creditpayBonusSum": 0.0,
                "creditpayCashSum": 0.0,
                "creditpayCashlessSum": 0.0,
                "creditpayCount": 0,
                "creditpaySum": 0.0,
                "creditpayVatAmounts": [],
                "currency": "AZN",
                "depositCount": 1,
                "depositSum": 1.0,
                "moneyBackBonusSum": 0.0,
                "moneyBackCashSum": 0.0,
                "moneyBackCashlessSum": 0.0,
```

```
                "moneyBackCount": 0,
                "moneyBackCreditSum": 0.0,
                "moneyBackPrepaymentSum": 0.0,
                "moneyBackSum": 0.0,
                "moneyBackVatAmounts": [],
                "periodSum": 3000.0,
                "periodVatFreeSum": 2000.0,
                "periodVatSum": 1000.0,
                "periodVatZeroSum": 0.0,
                "prepayBonusSum": 0.0,
                "prepayCashSum": 1000.0,
                "prepayCashlessSum": 0.0,
                "prepayCount": 1,
                "prepaySum": 1000.0,
                "prepayVatAmounts": [
                    {
                        "vatSum": 1000.0
                    }
                ],
                "rollbackBonusSum": 0.0,
                "rollbackCashSum": 0.0,
                "rollbackCashlessSum": 0.0,
                "rollbackCount": 0,
                "rollbackCreditSum": 0.0,
                "rollbackPrepaymentSum": 0.0,
                "rollbackSum": 0.0,
                "rollbackVatAmounts": [],
                "saleBonusSum": 0.0,
                "saleCashSum": 50.0,
                "saleCashlessSum": 1950.0,
                "saleCount": 4,
                "saleCreditSum": 5500.0,
                "salePrepaymentSum": 0.0,
                "saleSum": 2000.0,
                "saleVatAmounts": [
                    {
                        "vatSum": 1000.0
                    },
                    {
                        "vatPercent": 18.0,
                        "vatSum": 1000.0
                    }
                ],
                "withdrawCount": 1,
                "withdrawSum": 1.0
            }
        ],
        "docCountToSend": 8,
        "document_id": "4nwBfG2mZMSq4ztGGadNcWtXh5ieLW9BbJXKk1TVbkJx",
        "firstDocNumber": 10521,
        "lastDocNumber": 10527,
        "createdAtUtc": "09.10.2023 12:58:09",
        "reportNumber": "1831",
        "shiftOpenAtUtc": "09.10.2023 12:57:18"
    },
    "code": "0",
```

```
    "message": "Successful operation"
}
```

Property name	Type	Description
documentId	String	Doc id (Token Generate)
createdAtUtc	DateTime	Report created date
shiftOpenAtUtc	DateTime	Shift opening date
reportNumber	Integer	Report number
firstDocNumber	Integer	First document number
lastDocNumber	Integer	Last document number
docCountToSend	Integer	Count sending docs to token

This response provides detailed information about the transactions conducted during the shift, including data on sales, returns, deposits, and other transactions.

Deposit

The Deposit request is used to add a deposit amount to the cash drawer. During the workday, the manager may add or withdraw amounts from the cash register, and this request allows such operations to be recorded.

Example Request:

```
{
    "data": {
        "documentUUID": "unique ID for each transaction",  //Unical code this cheque
        "sum": 1.0,  //Deposite sum
        "cashierName": "New Cashier",  //Cashier Full Name
        "currency": "AZN"
    },
    "operation": "deposit",  //Operation type
    "username": "username",
    "password": "password"
}
```

Property name	Type	Description
operation	String	Operation name
sum	Float	Deposit sum
cashierName	String	Cashier name
currency	String	Currency code
documentUUID	String	Unical Document ID

Example Response:

```
{
    "data": {
        "document_id": "9eHUL46hMN3M3A6i4b39mNwFEVc8LD9f992R4tnQTupz",
        "document_number": 10525,
        "shift_document_number": 5,
        "short_document_id": "9eHUL46hMN3M",
        "totalSum": 0.0
    },
    "code": "0",
    "message": "Successoperation"
}
```

Property name	Type	Description
documentId	String	Document id
documentNumber	String	Document number
shiftDocumentNumber	String	Shift document number
short_document_id	String	Fiscal ID

Note: This request allows you to record the deposit of funds into the cash drawer. The unique identifier documentUUID helps prevent duplicate transactions.

Withdraw

The Withdraw request is used to withdraw money from the cash drawer. This allows you to record amounts that have been removed from the cash drawer, such as for deposit into a bank.

Example Request:

```
{
    "data": {
        "documentUUID": "unique ID for each transaction",  //Unical code this cheque
        "sum": 1.0, //Deposite sum
        "cashierName": "New Cashier",  //Cashier Full Name
        "currency": "AZN"
    },
    "operation": "withDraw",  //Operation type
    "username": "username",
    "password": "password"
}
}
```

Property name	Type	Description
operation	String	Operation name
sum	Float	Deposit sum
cashierName	String	Cashier name
currency	String	Currency code
documentUUID	String	Unical Document ID

Example Response:

```
{
    "data": {
        "document_id": "4uNkX5FuprnWuQTZnQaiVQut9hZEXopKESvk9LmHYNEs",
        "document_number": 10524,
        "shift_document_number": 4,
        "short_document_id": "4uNkX5FuprnW",
        "totalSum": 0.0
    },
    "code": "0",
    "message": "Successoperation"
}
```

Property name	Type	Description
documentId	String	Document idCre
documentNumber	String	Document number
shiftDocumentNumber	String	Shift document number
short_document_id	String	Short fiscal ID

Note: This request allows you to record the withdrawal of funds from the cash drawer. The unique identifier `documentUUID` helps prevent duplicate transactions.

Sales

The `Sales` request is used to process sales transactions on the cash register equipment. It includes parameters for various payment methods, information about the items on the receipt, and optional client data.

Example Request:

```
{
    "data": {
        "documentUUID":  "{{$guid}}", //Unical code this cheque
        "cashPayment": 00.0, //Cash payments
        "creditPayment": 0.0, //Credit payments
        "depositPayment": 0.0, //Advance payment
        "cardPayment": 0.01, //Bank kart payments
        "bonusPayment": 0.0, //Bonus Payments
            "items": [
            {
                "name": "Product Name", //Product full name
                "code": "Product Barcode", //Product barcode
                "quantity": 1.0, //Quantity product
                "salePrice": 0.01, //Price per item
                "purchasePrice": 0, //Product cost
                "codeType": 1, //Bacrode type,
                "quantityType": 1, //Quantity type
                "vatType": 1, //vat Type
                //"markingCode": "124686",  //Unical Marcing code
                "itemUuid": "12343265437yy645", //Unical Item Code GUID
                "discountAmount": 0.0,  // Discount on all products
                "markingCodes": [
                    "ooo333",
                    "qqq444"
                ]
            }
        ],
        "clientName": "Test Customer", //Customer discount card
        "clientTotalBonus": 0.0, //Customer total bonus
        "clientEarnedBonus": 0.0, //Customer bonuses this cheque
        "clientBonusCardNumber": "Customer Bonus card", //Customer discount card
        "cashierName": "New Cashier", //Cashier Full Name
        "note": "Text Description ", //Description
        "rrn":"", //card Payment - pos terminal cheque RRN number
        "currency": "AZN"
    },
    "operation": "sale", //Operation Type
    "username": "username",
    "password": "password"
}
```

Property name	Type	Description
Operation	String	Operation name
cashPayment	Float	Cash payment
cardPayment	Float	Card payment
bonusPayment	Float	Bonus payment
creditPayment	Float	Credit payment
depositPayment	Float	Deposit payment
clientName	String	Client name
clientTotalBonus	Float	Client total bonus
clientEarnedBonus	Float	Client earned bonus
clientBonusCardNumber	String	Card number
cashierName	String	Cashier name
currency	String	Currency code
prepymentDocumentId	String	Advance sale Doc ID
documentUUID	String	Unical Document ID
rrn	String	Bank transaction RRN
approval_code	String	Bank Approval Code
transaction_id	String	Bank Transaction ID
transaction_number	String	Bank Transaction Number
ITEMS		
name	String	Product name
code	String	Product code
quantity	Float	Quantity of product
salePrice	Float	Sale price of product
purchasePrice	Float	Purchase price of product
vatType	Integer	Vat type of product
quantityType	Integer	Unit type
codeType	Integer	Barcode Type
sum	Float	Sum of line (Not required)
markingCodes	String	Marking Codes Product

Note: This request enables the sale of goods with various payment methods. The unique identifier `documentUUID` helps prevent transaction duplication, and client information (such as bonuses and discount cards) is used for tracking discounts and bonus allocations.

Example Response:


```
{
  "data": {
    "approval_code": "537959",
    "document_id": "EMnVW3qyEbUSsJ4xTJbMytDStusgXMDauaCQxdJE1wuM",
    "document_number": 7045,
    "number": "180724A1A1197045",
    "rrn": "420012558257",
    "shift_document_number": 2,
    "short_document_id": "EMnVW3qyEbUS",
    "totalSum": 0.03,
    "transaction_id": "null",
    "transaction_number": "null"
  },
  "code": "0",
  "message": "Successful operation"
}
```

Property name	Type	Description
document_id	String	Document id
document_number	Integer	Document number
shift_document_number	Integer	Shift document number
short_document_id	String	Short document id
number	String	Document barcode number
totalSum	Float	Total Check SUM
rrn	String	Bank transaction RRN
approval_code	String	Bank Approval Code
transaction_id	String	Bank Transaction ID
transaction_number	String	Bank Transaction Number

Credit Sale

The `Credit Sale` request is used for selling goods on credit. It is identical to a regular sale, but the amount processed as credit is entered in the `creditPayment` field. The remaining amount (down payment) can be paid using any other available method.

Example Request:

```
{
    "data": {
        "documentUUID": "unique ID for each transaction", //Unical code this cheque
        "cashPayment": 0.0, //Cash payments
        "creditPayment": 400.0, //Credit payments
        "depositPayment": 0.0, //Advance payment
        "cardPayment": 80, //Bank kart payments
        "bonusPayment": 0.0, //Bonus Payments
          "items": [
            {
                "name": "Product Name", //Product full name
                "code": "Product Barcode", //Product barcode
                "quantity": 1.0, //Quantity product
                "salePrice": 480.0, //Price per item
                "purchasePrice": 1.17, //Product cost
                "codeType": 1, //Bacrode type,
                "quantityType": 1, //Quantity type
                "vatType": 1, //vat Type
                "discountAmount": 20.0  // Discount on all products
            }
        ],
        "clientName": "Test Customer", //Customer full name
        "clientTotalBonus": 670.0, //Customer total bonus
        "clientEarnedBonus": 480.0, //Customer bonuses this cheque
        "clientBonusCardNumber": "Customer Bonus card", //Customer discount card
        "cashierName": "New Cashier", //Cashier Full Name
        "note": "Text Description ", //Description
        "rrn":"123456789101", //card Payment - pos terminal cheque RRN number
        "creditContract": "A1", //Credit contract number
        "creditPayer": "CreditPayerName", //Payer Full Name
        "currency": "AZN"
    },
    "operation": "sale", //Operation Type
    "username": "username",
    "password": "password"
}
```

Property name	Type	Description
Operation	String	Operation name
cashPayment	Float	Cash payment
cardPayment	Float	Card payment
bonusPayment	Float	Bonus payment
depositPayment	Float	Advance payment
creditPayment	Float	Credit payment
clientName	String	Client name
clientTotalBonus	Float	Client total bonus
creditContract	String	Credit Contract number
creditPayer	String	Credit payer name
documentUUID	String	Unical Document ID
clientEarnedBonus	Float	Client earned bonus
clientBonusCardNumber	String	Card number
cashierName	String	Cashier name
currency	String	Currency code
ITEMS		
name	String	Product name
code	String	Product code
quantity	Float	Quantity of product
salePrice	Float	Sale price of product
purchasePrice	Float	Purchase price of product
vatType	Integer	Vat type of product
quantityType	Integer	Unit Type
codeType	Integer	Barcode Type
sum	Float	Sum of line (Not required)
markingCodes	String	Marking Codes Product

Example Response:


```
{
  "data": {
    "approval_code": "537959",
    "document_id": "EMnVW3qyEbUSsJ4xTJbMytDStusgXMDauaCQxdJE1wuM",
    "document_number": 7045,
    "number": "180724A1A1197045",
    "rrn": "420012558257",
    "shift_document_number": 2,
    "short_document_id": "EMnVW3qyEbUS",
    "totalSum": 0.03,
    "transaction_id": "null",
    "transaction_number": "null"
  },
  "code": "0",
  "message": "Successful operation"
}
```


Property name	Type	Description
document_id	String	Document id
document_number	Integer	Document number
shift_document_number	Integer	Shift document number
short_document_id	String	Short document id
number	String	Document barcode number
totalSum	Float	Total Check SUM
rrn	String	Bank transaction RRN
approval_code	String	Bank Approval Code
transaction_id	String	Bank Transaction ID
transaction_number	String	Bank Transaction Number

Credit Pay

Example Request and Response for "Credit Pay" Operation

Example Request:

```
{
    "data": {
        "documentUUID": "unique ID for each transaction",
      //Unical code this cheque
        "parentDocumentId": "DD4x53TAhSanAhS4DwSi7XDbUUrHKwQvK2rch",
      //Credit Sale documentID
        "cashPayment": 0.0, //Cash payments
        "cardPayment": 80, //Bank kart payments
        "items": [
            {
                "name": "Product Name", //Product full name
                "code": "Product Barcode", //Product barcode
                "quantity": 1.0, //Quantity product
                "salePrice": 80.0, //Payment SUM
                "realPrice": 480, //Price per item
                "codeType": 1, //Bacrode type,
                "quantityType": 1, //Quantity type
                "vatType": 1 //vat Type
            }
        ],
        "paymentNumber": "1", //Quantity of payments
        "residue": 400, //Remaining debt
        "creditContract": "A1", //Contract number
        "creditPayer": "CreditOwner", //Payer full name
        "clientName": "Test Customer", //Customer full name
        "clientTotalBonus": 670.0, //Customer total bonus
        "clientEarnedBonus": 480.0, //Customer bonuses this cheque
        "clientBonusCardNumber": "Customer Bonus card", //Customer discount card
        "cashierName": "New Cashier", //Cashier Full Name
        "note": "Text Description ", //Description
        "rrn":"123456789101", //card Payment - pos terminal cheque RRN number
        "currency": "AZN"
    },
    "operation": "credit", //Operation Type
    "username": "username",
    "password": "password"
}
```

Property name	Type	Description
Operation	String	Operation name
cashPayment	Float	Cash payment
cardPayment	Float	Card payment
bonusPayment	Float	Bonus payment
clientName	String	Client name
clientTotalBonus	Float	Client total bonus
clientEarnedBonus	Float	Client earned bonus
clientBonusCardNumber	String	Card number
cashierName	String	Cashier name
currency	String	Currency code
creditPayer	String	Credit payer name
creditContract	String	Credit Contract number
residue	Float	Balance owed
parentDocumentId	String	Sale Credit DocİD
documentUUID	String	Unical document ID
ITEMS		
name	String	Product name
code	String	Product code
quantity	Float	Quantity of product
salePrice	Float	Payment amount
realPrice	Float	Product Unit Price
vatType	Integer	Vat type of product
quantityType	Integer	Unit Type
codeType	Integer	Barcode type

Example Response:


```
{
   "data": {
      "approval_code": "537959",
      "document_id": "EMnVW3qyEbUSsJ4xTJbMytDStusgXMDauaCQxdJE1wuM",
      "document_number": 7045,
      "number": "180724A1A1197045",
      "rrn": "420012558257",
      "shift_document_number": 2,
      "short_document_id": "EMnVW3qyEbUS",
      "totalSum": 0.03,
      "transaction_id": "null",
      "transaction_number": "null"
   },
   "code": "0",
   "message": "Successful operation"
}
```


Property name	Type	Description
document_id	String	Document id
document_number	Integer	Document number
shift_document_number	Integer	Shift document number
short_document_id	String	Short document id
number	String	Document barcode number
totalSum	Float	Total Check SUM
rrn	String	Bank transaction RRN
approval_code	String	Bank Approval Code
transaction_id	String	Bank Transaction ID
transaction_number	String	Bank Transaction Number

Advance (Deposit) Sale

The `Advance (Deposit) Sale` operation is used to record an advance payment made by a customer, which can be used for future purchases. The advance amount is subject to VAT, and the advance can only be used for future purchases within the same VAT position.

Пример запроса

```
{
    "data": {
        "documentUUID": "unique ID for each transaction",
      //Unical code this cheque
        "sum": 710.0, //Advance deposite sum
        "vatType": 1,//vat Type
        "cashPayment": 0.0, //Cash payments
        "cardPayment": 710, //Bank kart payments
        "bonusPayment": 0.0, //Bonus Payments
        "clientName": "Test Customer", //Customer full name
        "clientTotalBonus": 670.0, //Customer total bonus
        "clientEarnedBonus": 480.0, //Customer bonuses this cheque
        "clientBonusCardNumber": "Customer Bonus card", //Customer discount card
        "cashierName": "New Cashier", //Cashier Full Name
        "currency": "AZN"
    },
    "operation": "prepayment", //Operation Type
    "username": "username",
    "password": "password"
}
```

Property name	Type	Description
Operation	String	Operation name
cashPayment	Float	Cash payment
cardPayment	Float	Card payment
bonusPayment	Float	Bonus payment
clientName	String	Client name
sum	Float	Sale price of product
vatType	Float	Vat type of product
cashierName	String	Cashier name
currency	String	Currency code
documentUUID	String	Unical Document ID
Rrn	String	Bank Transaction RRN

Example Response:


```
{
  "data": {
    "approval_code": "537959",
    "document_id": "EMnVW3qyEbUSsJ4xTJbMytDStusgXMDauaCQxdJE1wuM",
    "document_number": 7045,
    "number": "180724A1A1197045",
    "rrn": "420012558257",
    "shift_document_number": 2,
    "short_document_id": "EMnVW3qyEbUS",
    "totalSum": 0.03,
    "transaction_id": "null",
    "transaction_number": "null"
  },
  "code": "0",
  "message": "Successful operation"
}
```


Property name	Type	Description
document_id	String	Document id
document_number	Integer	Document number
shift_document_number	Integer	Shift document number
short_document_id	String	Short document id
number	String	Document barcode number
totalSum	Float	Total Check SUM
rrn	String	Bank transaction RRN
approval_code	String	Bank Approval Code
transaction_id	String	Bank Transaction ID
transaction_number	String	Bank Transaction Number

Advance Sale Items

The `Advance Sale Items` operation is used to add items to an advance payment that a customer makes for future purchases. All items in the request must be subject to the same type of VAT.

Example Request:

```
{
  "data": {
    "documentUUID": "unique ID for each transaction", //Unical code this cheque
    "sum": 710.0, //Advance deposite sum
    "vatType": 1,//vat Type
    "cashPayment": 0.0, //Cash payments
    "cardPayment": 710, //Bank kart payments
    "bonusPayment": 0.0, //Bonus Payments
    "items": [
      {
        "name": "Product Name", //Product full name
        "code": "Product Barcode", //Product barcode
        "quantity": 1.0, //Quantity product
        "salePrice": 710.0, //Payment SUM
        "codeType": 1, //Bacrode type,
        "quantityType": 1, //Quantity type
        "vatType": 1 //vat Type
      }
    ],
    "clientName": "Test Customer", //Customer full name
    "clientTotalBonus": 670.0, //Customer total bonus
    "clientEarnedBonus": 480.0, //Customer bonuses this cheque
    "clientBonusCardNumber": "Customer Bonus card", //Customer discount card
    "cashierName": "New Cashier", //Cashier Full Name
    "currency": "AZN",
    "note": "Text text text" //Description
  },
  "operation": "prepaymentProducts", //Operation Type
  "username": "username",
  "password": "password"
}
```

Property name	Type	Description
operation	String	Operation name
documentUUID	String	Unical document ID
cashPayment	Float	Cash payment
cardPayment	Float	Card payment
bonusPayment	Float	Bonus payment
clientName	String	Client name
cashierName	String	Cashier name
currency	String	Currency code
sum	Float	Product Total Price
rrn	String	Bank transaction RRN
ITEMS		
name	String	Product name
code	String	Product code
quantity	Float	Quantity of product
salePrice	Float	Payment amount

vatType	Integer	Vat type of product
quantityType	Integer	Unit Type
codeType	Integer	Barcode type

Example Response:


```
{
   "data": {
      "approval_code": "537959",
      "document_id": "EMnVW3qyEbUSsJ4xTJbMytDStusgXMDauaCQxdJE1wuM",
      "document_number": 7045,
      "number": "180724A1A1197045",
      "rrn": "420012558257",
      "shift_document_number": 2,
      "short_document_id": "EMnVW3qyEbUS",
      "totalSum": 0.03,
      "transaction_id": "null",
      "transaction_number": "null"
   },
   "code": "0",
   "message": "Successful operation"
}
```


Property name	Type	Description
document_id	String	Document id
document_number	Integer	Document number
shift_document_number	Integer	Shift document number
short_document_id	String	Short document id
number	String	Document barcode number
totalSum	Float	Total Check SUM
rrn	String	Bank transaction RRN
approval_code	String	Bank Approval Code
transaction_id	String	Bank Transaction ID
transaction_number	String	Bank Transaction Number

Advance Pay

The `Advance Pay` operation is used for selling goods using a previously issued advance payment receipt. When creating such a request, you need to specify the identifier of the advance payment document (`prepaymentDocumentId`), which is the parent document. The advance amount should be entered in the `depositPayment` field, and the remaining amount can be distributed among other payment methods (such as cash, card, or bonuses).

Example Request:

```
{
  "data": {
    "documentUUID": "unique ID for each transaction", //Unical code this cheque
    "cashPayment": 500.0, // Banknotes provided by the client in cash
    "depositPayment": 0.0, // The amount that they want to pay in advance
    "cardPayment": 80, // Payment amount by card - cashless payment
    "bonusPayment": 0.0, // Payment amount with bonus card
    "prepaymentDocumentId": "4QbqqMxBHhQbADgDhvEdhxtgEAAJcKXLp",
        //Advance sale documentID
    "items": [
      {
        "name": "Product Name", //Product full name
        "code": "Product Barcode", //Product barcode
        "quantity": 1.0, //Quantity product
        "salePrice": 480.0, //Price per item
        "purchasePrice": 1.17, //Product cost
        "codeType": 1, //Bacrode type,
        "quantityType": 1, //Quantity type
        "vatType": 1, //vat Type
        "discountAmount": 20.0  // Discount on all products
      }
    ],
    "clientName": "Test Customer", //Customer full name
    "clientTotalBonus": 670.0, //Customer total bonus
    "clientEarnedBonus": 480.0, //Customer bonuses this cheque
    "clientBonusCardNumber": "Customer Bonus card", //Customer discount card
    "cashierName": "New Cashier", //Cashier full name
    "currency": "AZN"
  },
  "operation": "sale", //Operation Type
  "username": "username",
  "password": "password"
}
```

Property name	Type	Description
Operation	String	Operation name
cashPayment	Float	Cash payment
cardPayment	Float	Card payment
bonusPayment	Float	Bonus payment
depositPayment	Float	Advance payment
clientName	String	Client name
prepaymentDocumentId	String	Advance Doc ID
cashierName	String	Cashier name
currency	String	Currency code
documentUUID	String	Unical Document ID
rrn	String	Bank Transaction RRN
ITEMS		
name	String	Product name
code	String	Product code
quantity	Float	Quantity of product
salePrice	Float	Sale price of product
vatType	Integer	Vat type of product
quantityType	Integer	Unit Type
codeType	Integer	Barcode type

Example Response:


```
{
  "data": {
    "approval_code": "537959",
    "document_id": "EMnVW3qyEbUSsJ4xTJbMytDStusgXMDauaCQxdJE1wuM",
    "document_number": 7045,
    "number": "180724A1A1197045",
    "rrn": "420012558257",
    "shift_document_number": 2,
    "short_document_id": "EMnVW3qyEbUS",
    "totalSum": 0.03,
    "transaction_id": "null",
    "transaction_number": "null"
  },
  "code": "0",
  "message": "Successful operation"
}
```


Property name	Type	Description
document_id	String	Document id
document_number	Integer	Document number
shift_document_number	Integer	Shift document number
short_document_id	String	Short document id
number	String	Document barcode number
totalSum	Float	Total Check SUM
rrn	String	Bank transaction RRN
approval_code	String	Bank Approval Code
transaction_id	String	Bank Transaction ID
transaction_number	String	Bank Transaction Number

Money Back (Refound)

The `Money Back` operation is used for returning goods based on previously issued receipts. The `parentDocumentId` field must specify the `DocumentID` of the receipt for which the return is being processed. The `moneyBackType` field determines the type of document for the return: sale, advance, or credit payment.

Example Request:

```
{
    "data": {
        "parentDocumentId": "712EyaqxZNtFT9FFg5k8s2tvqPxiKzeXr2yo7aBjq1Dm",
       // Fiscal ID of the parent's check
        "documentUUID": "unique ID for each transaction", //Unique ID for a check
        "cashPayment": 0.0, // Banknotes provided by the client in cash
        "creditPayment": 0.0, // The amount that is issued on credit
        "depositPayment": 0.0, // The amount that they want to pay in advance
        "cardPayment": 1.0, // Payment amount by card - cashless payment
        "bonusPayment": 0.0, // Payment amount with bonus card
        "items": [
                {
                "name": "Product Name",  // Product Name
                "code": "Product Barcode", // Product Barcode
                "quantity": 1.0, // Product Quantity
                "salePrice": 2.0, // One Product Sale Price
                "purchasePrice": 1.0, // One Product Seed Price
                "codeType": 1, // Bacrode type,
                "quantityType": 3, // Quantity type
                "vatType": 1, // vat Type
                "markingCode": "124686",  //Unical Marcing code
                "itemUuid": "12343265437645", //Unical Item Code GUID
                "parentItemUuid": "12343265437645", //Return item UID
                "discountAmount": 0.0,  // Discount on all products

            }
        ],
        "isManual": true,
        "moneyBackType": 0, //Money back type
        "clientName": "Test Customer", //Customer discount card
        "clientTotalBonus": 670.0, //Customer total bonus
        "clientEarnedBonus": 480.0, //Customer bonuses this cheque
        "clientBonusCardNumber": "Customer Bonus card", //Customer discount card
        "cashierName": "New Cashier", //Cashier Full Name
        "currency": "AZN",
        "rrn": "420411938397",//Bank transaction RRN Number
        "transactionId": "", //Bank transaction approval code
        "transactionNumber": "", //Bank transaction ID
        "approvalCode": "", //Bank transaction number
        "note": "Text Description " //Description
    },
    "operation": "moneyBack",
    "username": "username",
    "password": "password"
}
```

Property name	Type	Description
operation	String	Operation name
parentDocumentId	String	Document id of Sale
cashPayment	Float	Cash payment
cardPayment	Float	Card payment
bonusPayment	Float	Bonus payment
depositPayment	Float	Deposit payment
creditPayment	Float	Credit payment
clientName	String	Client name
clientTotalBonus	Float	Client total bonus
clientEarnedBonus	Float	Client earned bonus
clientBonusCardNumber	String	Card number
cashierName	String	Cashier name
currency	String	Currency code
moneyBackType	Integer	Sale, Advance Sale, Credit Pay
documentUUID	String	Unical Barcode ID
rrn	String	Bank transaction RRN
ITEMS		
name	String	Product name
code	String	Product code
quantity	Float	Quantity of product
salePrice	Float	Product Sale Price
purchasePrice	Float	Product purchase price
vatType	Integer	Vat type
quantityType	Integer	Quantity type
codeType	Integer	Barcode Type
MarkingCodes	String	Marking Codes Product

Example Response:


```
{
  "data": {
    "approval_code": "537959",
    "document_id": "EMnVW3qyEbUSsJ4xTJbMytDStusgXMDauaCQxdJE1wuM",
    "document_number": 7045,
    "number": "180724A1A1197045",
    "rrn": "420012558257",
    "shift_document_number": 2,
    "short_document_id": "EMnVW3qyEbUS",
    "totalSum": 0.03,
    "transaction_id": "null",
    "transaction_number": "null"
  },
  "code": "0",
  "message": "Successful operation"
}
```


Property name	Type	Description
document_id	String	Document id
document_number	Integer	Document number
shift_document_number	Integer	Shift document number
short_document_id	String	Short document id
number	String	Document barcode number
totalSum	Float	Total Check SUM
rrn	String	Bank transaction RRN
approval_code	String	Bank Approval Code
transaction_id	String	Bank Transaction ID
transaction_number	String	Bank Transaction Number

Notes:

- **parentDocumentId**: Used for specifying the receipt from which the return is being processed.
- **moneyBackType**: Indicates the type of return:
  - `0` — Sale
  - `6` — Advance
  - `7` — Credit Payment

Money Back Manual

The `Money Back Manual` method is used for returning goods and/or services similarly to the standard return method, but without performing additional checks on the device. This method is useful in situations where data on the device might be damaged or lost, or if the return operation needs to be performed on another device.

Important: This method does not validate the correctness of the return data in the device's database, so it can be used in cases where a return needs to be processed without validation.

Example Request:

```
{
    "data": {
        "parentDocumentId": "712EyaqxZNtFT9FFg5k8s2tvqPxi",  // Fiscal ID of the parent's check
        "documentUUID": "unique ID for each transaction", //Unique ID for a check
        "cashPayment": 0.0, // Banknotes provided by the client in cash
        "creditPayment": 0.0, // The amount that is issued on credit
        "depositPayment": 0.0, // The amount that they want to pay in advance
        "cardPayment": 1.0, // Payment amount by card - cashless payment
        "bonusPayment": 0.0, // Payment amount with bonus card
        "items": [
                {
                "name": "Product Name",  // Product Name
                "code": "Product Barcode", // Product Barcode
                "quantity": 1.0, // Product Quantity
                "salePrice": 2.0, // One Product Sale Price
                "purchasePrice": 1.0, // One Product Seed Price
                "codeType": 1, // Bacrode type,
                "quantityType": 3, // Quantity type
                "vatType": 1, // vat Type
                "markingCode": "124686",  //Unical Marcing code
                "itemUuid": "12343265437645", //Unical Item Code GUID
                "parentItemUuid": "12343265437645", //Return item UID
                "discountAmount": 0.0,  // Discount on all products
                "markingCodes": [
                    "ooo333",
                    "qqq444"
                ]
            }
        ],
        "isManual": true,
        "moneyBackType": 0, //Money back type
        "clientName": "Test Customer", //Customer discount card
        "clientTotalBonus": 670.0, //Customer total bonus
        "clientEarnedBonus": 480.0, //Customer bonuses this cheque
        "clientBonusCardNumber": "Customer Bonus card", //Customer discount card
```

```
        "cashierName": "New Cashier", //Cashier Full Name
        "currency": "AZN",
        "rrn": "420411938397",//Bank transaction RRN Number
        "transactionId": "", //Bank transaction approval code
        "transactionNumber": "", //Bank transaction ID
        "approvalCode": "", //Bank transaction number
        "note": "Text Description " //Description
    },
    "operation": "moneyBack",
    "username": "username",
    "password": "password"
}
```

Property name	Type	Description
operation	String	Operation name
parentDocumentId	String	Document id of Sale
isManual	Boolean	Manual MoneyBack
cashPayment	Float	Cash payment
cardPayment	Float	Card payment
bonusPayment	Float	Bonus payment
depositPayment	Float	Deposit payment
creditPayment	Float	Credit payment
clientName	String	Client name
clientTotalBonus	Float	Client total bonus
clientEarnedBonus	Float	Client earned bonus
clientBonusCardNumber	String	Card number
cashierName	String	Cashier name
currency	String	Currency code
moneyBackType	Integer	Sale, Advance Sale, Credit Pay
documentUUID	String	Unical Document ID
rrn	String	Bank transaction RRN
approval_code	String	Bank Approval Code
transaction_id	String	Bank Transaction ID
transaction_number	String	Bank Transaction Number
ITEMS		
name	String	Product name
code	String	Product code
quantity	Float	Product quantity
salePrice	Float	Product sale price
vatType	Integer	Product vat type
purchasePrice	Float	Product purchase price
quantityType	Integer	Product quantity type
codeType	Integer	Product barcode type
markingCodes	String	Marking code product

Example Response:


```
{
  "data": {
    "approval_code": "537959",
    "document_id": "EMnVW3qyEbUSsJ4xTJbMytDStusgXMDauaCQxdJE1wuM",
    "document_number": 7045,
    "number": "180724A1A1197045",
    "rrn": "420012558257",
    "shift_document_number": 2,
    "short_document_id": "EMnVW3qyEbUS",
    "totalSum": 0.03,
    "transaction_id": "null",
    "transaction_number": "null"
  },
  "code": "0",
  "message": "Successful operation"
}
```


Property name	Type	Description
document_id	String	Document id
document_number	Integer	Document number
shift_document_number	Integer	Shift document number
short_document_id	String	Short document id
number	String	Document barcode number
totalSum	Float	Total Check SUM
rrn	String	Bank transaction RRN
approval_code	String	Bank Approval Code
transaction_id	String	Bank Transaction ID
transaction_number	String	Bank Transaction Number

Notes:

- **parentDocumentId**: Used for specifying the receipt from which the return is being processed.
- **moneyBackType**: Indicates the type of return:
  - 0 — Sale
  - 6 — Advance
  - 7 — Credit Payment

Correction

The `Correction` method is used in cases where the cash register equipment was non-functional for a certain period, but transactions continued to occur. A correction receipt allows you to specify the period during which the device was not operational and to record the amounts processed in cash or by card during this time without the involvement of the cash register equipment.

Example Request:

```
{
  "data": {
    "documentUUID": "unique ID for each transaction1x11xx", //Unical code this cheque
    "firstOperationAtUtc": "2021-08-31T03:39:11Z", //Date of first document
    "lastOperationAtUtc": "2021-08-31T04:39:11Z", //Date of last document
    "currency": "AZN",
    "cashPayment": 100.0, //Cash payments
    "cardPayment": 450, //Bank kart payments
    "vatType": 1,
    "note": "" //Description
  },
  "operation": "correction", //Operation type
  "cashierName": "Cashier Name", //Cashier Full name
  "username": "username",
  "password": "password"
}
```

Property name	Type	Description
operation	String	Operation name
cashPayment	Float	Cash payment
cardPayment	Float	Card payment
cashierName	String	Cashier name
currency	String	Currency code
vatType	Integer	Product vat type
firstOperationAtUtc	datetime	Date of first document
lastOperationAtUtc	datetime	Date of last document
documentUUID	String	Unical code this cheque
note	String	Description

Example Response:

```
{
  "data": {
    "document_id": "q3W9xUnrrQYJeCTRPVoYRSrHsxrP27U6AeZPys1kZAf",
    "document_number": 4911,
    "number": "280823111164911",
    "shift_document_number": 7,
    "short_document_id": "q3W9xUnrrQYJ",
    "totalSum": 87.56
  },
  "code": "0",
  "message": "Success operation"
}
```

RollBack

The `RollBack` method is used to cancel a receipt associated with the current shift when there is a need to annul a transaction. It allows you to reverse transactions only within the current shift; for receipts from previous shifts, only the `Money Back` request is available.

Example Request:

```
{
    "data": {
        "documentUUID": "unique ID for each transaction",
       //Unical code this cheque
        "parentDocumentId": "CxWJQV3esBSfnVuUWbsKPyi4hgMux",
       //Sale or Correction DocumentID
        "sum": 0.70, //Rollback total sum
        "cashPayment": 0.70, //Cash payments
        "creditPayment": 0.0, //Credit payments
        "depositPayment": 0.0, //Advance payment
        "cardPayment": 0.0, //Bank card payments
        "bonusPayment": 0.0, //Bonus Payments
        "vatAmounts": [
            {
                "vatType": 1, //vatType ID
                "vatSum": 0.70 //Vat SUM
            }
        ],
        "note": "123456" //Description
    },
    "cashierName": "New Cashier", //Cashier Full Name
    "currency": "AZN",
    "operation": "rollBack", //Operation type
    "username": "username",
    "password": "password"
}
```

Example Response:

```
{
    "data": {
        "approval_code": "537959",
        "document_id": "EMnVW3qyEbUSsJ4xTJbMytDStusgXMDauaCQxdJE1wuM",
        "document_number": 7045,
        "number": "180724A1A1197045",
        "rrn": "420012558257",
        "shift_document_number": 2,
        "short_document_id": "EMnVW3qyEbUS",
        "totalSum": 0.03,
        "transaction_id": "null",
        "transaction_number": "null"
    },
    "code": "0",
    "message": "Successful operation"
}
```

X-Report

The `X Report` method is used to generate and obtain an intermediate report on cash register operations conducted over a specified period. This report does not close the shift but serves to provide information on the current state of the cash drawer, including sales, returns, and other transactions.

Example Request:

```
{
   "data": {
      "cashierName": "Cashier Name"
   },
   "operation": "getXReport",
   "username": "username",
   "password": "password"
}
```

Property name	Type	Description
operation	String	Operation name
cashierName	String	Cashier Name

Example Response:

```
{
    "data": {
        "currencies": [
            {
                "correctionBonusSum": 0.0,
                "correctionCashSum": 0.0,
                "correctionCashlessSum": 0.0,
                "correctionCount": 0,
                "correctionCreditSum": 0.0,
                "correctionPrepaymentSum": 0.0,
                "correctionSum": 0.0,
                "correctionVatAmounts": [],
                "creditpayBonusSum": 0.0,
                "creditpayCashSum": 0.0,
                "creditpayCashlessSum": 0.0,
                "creditpayCount": 0,
                "creditpaySum": 0.0,
                "creditpayVatAmounts": [],
                "currency": "AZN",
                "depositCount": 0,
                "depositSum": 0.0,
                "moneyBackBonusSum": 0.0,
                "moneyBackCashSum": 2.0,
                "moneyBackCashlessSum": 0.2,
                "moneyBackCount": 1,
                "moneyBackCreditSum": 0.0,
                "moneyBackPrepaymentSum": 0.0,
                "moneyBackSum": 2.2,
                "moneyBackVatAmounts": [
                    {
                        "vatSum": 2.2
                    }
                ],
```

```
                "periodSum": 0.5999999,
                "periodVatFreeSum": -0.8000001,
                "periodVatSum": 1.4,
                "periodVatZeroSum": 0.0,
                "prepayBonusSum": 0.0,
                "prepayCashSum": 0.0,
                "prepayCashlessSum": 0.0,
                "prepayCount": 0,
                "prepaySum": 0.0,
                "prepayVatAmounts": [],
                "rollbackBonusSum": 0.0,
                "rollbackCashSum": 0.0,
                "rollbackCashlessSum": 0.0,
                "rollbackCount": 0,
                "rollbackCreditSum": 0.0,
                "rollbackPrepaymentSum": 0.0,
                "rollbackSum": 0.0,
                "rollbackVatAmounts": [],
                "saleBonusSum": 0.0,
                "saleCashSum": 2.8,
                "saleCashlessSum": 0.0,
                "saleCount": 1,
                "saleCreditSum": 0.0,
                "salePrepaymentSum": 0.0,
                "saleSum": 2.8,
                "saleVatAmounts": [
                    {
                        "vatPercent": 18.0,
                        "vatSum": 1.4
                    },
                    {
                        "vatSum": 1.4
                    }
                ],
                "withdrawCount": 0,
                "withdrawSum": 0.0
            }
        ],
        "docCountToSend": 0,
        "firstDocNumber": 10498,
        "lastDocNumber": 10499,
        "createdAtUtc": "04.10.202315:04:09",
        "reportNumber": "1828",
        "shiftOpenAtUtc": "04.10.2023 15:02:13"
    },
    "code": "0",
    "message": "Success operation"
}
```

Property name	Type	Description
correctionCount	Integer	Correction count
correctionCashSum	Float	Correction cash sum
correctionCashlessSum	Float	Correction float sum
correctionBonusSum	Float	Correction bonus sum
correctionCreditSum	Float	Correction credit sum
correctionPrepaymentSum	Float	Correction prepayment sum
correctionSum	Float	Correction total sum
depositCount	Integer	Deposit count
depositSum	Float	Deposit total sum
withDrawCount	Integer	WithDraw count
withDrawSum	Float	WithDraw total sum
saleCount	Integer	Sale count
saleCashSum	Float	Sale cash sum
saleCashlessSum	Float	Sale cashless sum
saleBonusSum	Float	Sale bonus sum
saleCreditSum	Float	Sale credit Sum
salePrepaymentSum	Float	Sale prepayment sum
saleSum	Float	Sale total sum
moneyBackCount	Integer	Money back count
moneyBackSum	Float	Money back sum
moneyBackCashSum	Float	Money back cash sum
moneyBackCashlessSum	Float	Money back cashless sum
moneyBackCreditSum	Float	Money back credit sum
moneyBackPrepaymentSum	Float	Money back prepayment sum
moneyBackBonusSum	Float	Money back bonus sum
docCountToSend	Integer	Sending doc count to token
firstDocNumber	Integer	First sending doc number
lastDocNumber	Integer	Last sending doc number
createdAtUtc	DateTime	Report created date
shiftOpenAtUtc	DateTime	Shift opening date
reportNumber	Integer	Report number
Vat amounts		
vatSum	Float	Vat sum

Print Last

The `Print Last` method is used to print the most recent receipt processed by the cash register. This method is useful in situations where there is a need to reprint the last receipt for the customer or for internal record-keeping.

Example Request:

```
{
   "operation": "printLastCheque",
   "username": "username",
   "password": "password"
}
```

Property name	Type	Description
operation	String	Operation name

Example Response:

```
{
   "code": "0",
   "message": "Success operation"
}
```

Periodic Report

The `Periodic Report` method is used to generate a report for a specified period. This method provides detailed information on all transactions conducted with the cash register during the given time frame.

Example Request:

```
{
   "operation": "getPeriodicZReport",
   "username": "username",
   "password": "password",
   "data": {
      "startDate": "23.08.2021 00:00:00",
      "endDate": "23.08.2021 23:59:59"
   }
}
```

Example Response:

```
{
    "data": {
        "cancellationBonusTotal": 0.0,
        "cancellationCashTotal": 0.0,
        "cancellationCashlessTotal": 0.0,
        "cancellationCount": 0,
        "cancellationCreditTotal": 0.0,
        "cancellationPrepaymentTotal": 0.0,
        "cancellationTotal": 0.0,
        "cancellationVat": 0.0,
        "cancellationVatFreeTotal": 0.0,
        "cancellationVatTotal": 0.0,
        "cashInCount": 20,
        "cashInTotal": 6621.0,
        "cashOutCount": 7,
```

```
        "cashOutTotal": 786.0,
        "correctionBonusTotal": 0.0,
        "correctionCashTotal": 0.0,
        "correctionCashlessTotal": 0.0,
        "correctionCount": 0,
        "correctionCreditTotal": 0.0,
        "correctionPrepaymentTotal": 0.0,
        "correctionTotal": 0.0,
        "correctionVat": 0.0,
        "correctionVatFreeTotal": 0.0,
        "correctionVatTotal": 0.0,
        "creditBonusTotal": 0.0,
        "creditCashTotal": 0.0,
        "creditCashlessTotal": 0.0,
        "creditCount": 0,
        "creditVatFreeTotal": 0.0,
        "creditVatTotal": 0.0,
        "date": "04.10.2023 15:08:29",
        "endDate": "04.10.2023 23:59:59",
        "periodSum": 83331.6,
        "periodVatFreeSum": 1203.8201,
        "periodVatSum": 80920.28,
        "prepaymentBonusTotal": 0.0,
        "prepaymentCashTotal": 1775.99,
        "prepaymentCashlessTotal": 49754.01,
        "prepaymentCount": 5,
        "prepaymentVatFreeTotal": 1000.0,
        "prepaymentVatTotal": 50000.0,
        "saleBonusTotal": 23.0,
        "saleCashTotal": 107626.26,
        "saleCashlessTotal": 2641.35,
        "saleCount": 1746,
        "saleCreditTotal": 100.0,
        "salePrepaymentTotal": 33.0,
        "saleReturnBonusTotal": 18.0,
        "saleReturnCashTotal": 43502.93,
        "saleReturnCashlessTotal": 50363.0,
        "saleReturnCount": 65,
        "saleReturnCreditTotal": 0.0,
        "saleReturnPrepaymentTotal": 0.0,
        "saleReturnTotal": 93883.93,
        "saleReturnVatFreeTotal": 0.0,
        "saleReturnVatTotal": 92117.93,
        "saleTotal": 110290.61,
        "saleVatFreeTotal": 203.82,
        "saleVatTotal": 107700.32,
        "startDate": "01.05.2023 00:00:00"
    },
    "code": "0",
    "message": "Success operation"
}
```

Control Tape

The `Control Tape` method is used to obtain detailed information about all transactions conducted during the current shift. This method allows you to print a complete list of operations that occurred within the current shift on the cash register.

Example Request:

```
{
   "operation": "getControlTape",
   "username": "username",
   "password": "password"
}
```

Example Response:


```
{
    "data": {
        "createdAtUtc": "04.10.2023 15:10:27",
        "documents_quantity": 4,
        "shift_number": 1828,
        "shiftOpenAtUtc": "04.10.2023 15:02:13",
        "tape": [
            {
                "createdAtUtc": "04.10.2023 15:03:03",
                "document_id": "EUY93uDHu3VE92usK6iLSE1wv4mCH6u4VHVs4dvTBYq",
                "item_count": 2,
                "total_sum": 2.8,
                "total_vat": 0.21,
                "delivered": true,
                "number": 10498,
                "short_document_id": "EUY93uDHu3VE",
                "type": "sale"
            },
            {
                "createdAtUtc": "04.10.2023 15:03:25",
                "document_id": "2Gq6H2CdvdqasLAtcr9nd8hoZ1dzzgKPyL1dkiDphw4U",
                "item_count": 1,
                "total_sum": 2.2,
                "total_vat": 0.0,
                "delivered": true,
                "number": 10499,
                "short_document_id": "2Gq6H2Cdvdqa",
                "type": "money_back"
            },
            {
                "createdAtUtc": "04.10.2023 15:07:35",
                "document_id": "kZRUAvfYuLoVsBLAGT8zHrKykqwV2CuVkbsg8fgSa5R",
                "item_count": 0,
                "total_sum": 1.0,
                "total_vat": 0.0,
                "delivered": true,
                "number": 10500,
                "short_document_id": "kZRUAvfYuLoV",
                "type": "withdraw"
            },
```

```
            {
                "createdAtUtc": "04.10.2023 15:07:38",
                "document_id": "9WFcZocuqpn3hf8xBg7dxd9ePfpxUD5ubwW1X8TTmpFn",
                "item_count": 0,
                "total_sum": 1.0,
                "total_vat": 0.0,
                "delivered": true,
                "number": 10501,
                "short_document_id": "9WFcZocuqpn3",
                "type": "deposit"
            }
        ]
    },
    "code": "0",
    "message": "Success operation"
}
```


Printer Connection

The `Print Connection` method is used to obtain the status of the printer. This method provides information about the current connectivity and operational status of the printer connected to the cash register.

Example Request:


```
{
  "operation": "isConnectedPrinter",
  "username": "username",
  "password": "password"
}
```


Property name	Type	Description
operation	String	Operation name

Example Response:


```
{
  "code": "0",
  "message": "Success operation"
}
```


Open CashBox

Example Request:


```
{
  "operation": "openCashbox",
  "username": "username",
  "password": "password"
}
```


Property name	Type	Description
operation	String	Operation name

Example Response:


```
{
  "code": "0",
  "message": "Success operation"
}
```

Enum:

Vat types

Property name	Value	Description
Vat18	1	ƏDV 18 %
TradeSupplement	2	Ticarət ƏDV-si 18%
VatFree	3	ƏDV-dən azad
Vat0	5	ƏDV 0%
SV 2%	6	Sadaləşdirilmiş 2%
SV 8%	7	Sadaləşdirilmiş 8%

Document types

Property name	Value	Description
Sale	0	Sale document
MoneyBack	1	Money back document
Deposit	2	Deposit document
WithDraw	3	With draw document
X-Report	4	X-Report document
Z-Report	5	Close Shift document
Correction	6	Correction document
RollBack	7	RollBack document
Credit	8	Credit document
Prepayment	9	Prepayment document
CreditReturn	10	Credit Money back
PrepaymentReturn	11	Advance Money back

Quantity types

Property name	Value	Description
Ədəd	0	
KQ	1	
Litr	2	
Metr	3	
Kv. metr	4	
Kub. metr	5	

Code Types

Property name	Value	Description
Plain text	0	arbitrary value
EAN8	1	EAN8 barcode
EAN13	2	EAN13 barcode
Service	3	service code

Errors

Number	Description
0	Successful operation
1	Repeated DocumentUUID value
999	Error message:

Common Errors and Solutions

****405: pin locked too many invalid pins****

Description:

The key is locked due to too many incorrect PIN codes.

Solution:

Contact support, as the key is locked and will require assistance to unlock.

****401: invalid login session****

Description:

An invalid Access token is being used, or the token is not being sent at all.

Solution:

Check and update the Access token. Ensure that the token is correctly sent with the request.

****403: invalid user role****

Description:

Invalid user role. You need to log in as a cashier and enable integration mode.

Solution:

Log in with the cashier role and activate integration mode.

****503: document: invalid shift status****

Description:

Invalid shift status. The shift may be closed, and transactions cannot be sent to a closed shift.

Solution:

Check the shift status. Open a new shift if the current one is closed.

****504: document: invalid shift duration****

Description:

Shift status is invalid as more than 24 hours have passed since the shift was opened.

Solution:

Check the shift duration. If more than 24 hours have passed, open a new shift.

****601: cashbox number != value from company certificate****

Description:

Incorrect factory number of the cash register equipment.

Solution:

Contact support to check and correct the factory number of the cash register equipment.

602: cashregister number != value from company certificate

Description:

The tax key certificate has expired.

Solution:

Re-register to obtain a new certificate by contacting support.

604: document time not in permitted interval

Description:

The cash register equipment has exceeded the allowed offline mode period.

Solution:

Connect to the internet to transmit previous information to the tax authority.

607: invalid total sum

Description:

Invalid total amount to be paid.

Solution:

Check and correct the total amount to be paid in the document.

611: invalid items sum

Description:

Incorrect sum of items.

Solution:

Ensure that the item sum is correctly stated and matches the document.

612: invalid item code type

Description:

The type of barcode is not specified.

Solution:

Specify the correct barcode type for the item.

613: invalid vat amounts total

Description:

The total VAT amount is incorrect.

Solution:

Check and correct the total VAT amount in the document.

****616: invalid percentages****

Description:

Incorrect type of VAT.

Solution:

Ensure that the type of VAT is correctly specified in the document.

****622: invalid length of item marking code****

Description:

The length of the item marking code is incorrect.

Solution:

Check the length of the item marking code and correct it if necessary.

****629: invalid change sum value****

Description:

The change amount for the transaction is not specified.

Solution:

Update the device to the latest version to ensure automatic calculation of the change amount is performed correctly.

****646: invalid length of RRN****

Description:

Incorrect length of the banking transaction.

Solution:

Check the length of the RRN and correct it if necessary.

This format should help you clearly present the information and simplify the error resolution process for users.

Error Messages:

0	Successful operation
101	generic initialization error
102	init hal drivers failed
103	init hal rcc failed
104	init hal rcc clock failed
105	init hal fram (fast memory) driver failed
106	init hal winbond (slow memory) driver failed
107	init random hal driver failed
108	init fast memory wrapper failed
109	init slow memory wrapper failed
110	memory overflow
111	generate new number with random module failed
112	initialization for bootloader: set firmware flag failed
113	erase sector with hal driver failed
114	write to flash memory failed (mem for firmware)
115	hal: init iwdg falied
116	post-production FAST memory test: write/read data mismatch
117	post-production SLOW memory test: write/read data mismatch
118	storage: unsuitable storage manager
119	assert failed
120	cpu memory: write error
121	cpu memory: erase error
122	cpu memory: read error
123	winbond: read err
124	winbond: write err
125	winbond: erase err
126	fram: read err
127	fram: write err
128	sec: master key: encrypt err
129	sec: master key: decrypt err
130	cpu memory: overflow
131	init: init in progress
201	io operation overflow memory
202	io object is not free
203	io object is not found
204	io operation isn't available in this state
205	io operation internal error
206	io invalid page number
207	io operation page overflow
208	storage page is not empty
209	storage page is corrupted
210	storage page: seek overflow
211	storage page: can not flush empty
212	storage page: invalid page type
213	storage file: storage overflow
214	storage file: read overflow
215	storage file: remove not first
216	storage file: file not found
217	storage has no free space
218	storage: document for remove isn't exist
219	storage: last document guid isn't matched
220	types: data already exists
221	types: data isn't exist
222	types: rw op overflow
223	lifo core: invalid page ordering
224	lifo core: next file overflow
225	lifo core: invalid next data page
226	lifo core: invalid file for commit

227	io file: flush with empty data
228	io updates: invalid state
229	activation: not activated
230	io: control tape overflow
231	last doc: no documents have been created yet
232	types: data hasn't been written correctly
233	migration: failed: all documents should be sent before migration
240	tx parity: init error
241	tx parity: invalid byte state
242	storage: overflow detected in the object header
243	storage: no enough elements in the object header
244	storage: too many items in header object
245	storage: object header: dec items error
246	storage: object header is empty
247	storage: page isn't included into header
248	storage: object header isn't empty
249	storage: object header isn't full
250	storage: header: r/w to invalid header page
251	storage: header page is not empty
252	storage: header page is not committed
253	storage: object storage is full
254	storage: invalid state of the object page
255	storage: invalid index of data page
256	storage: overflow of data page
257	storage: dynamic: invalid data size value
258	storage: dynamic: output buffer overflow
259	storage: dynamic: end page is expected
260	storage: current iteration isn't initialized
261	storage: overflow of persistent object
262	storage: nothing exists in the persistent object
263	storage: persistent object is not empty
264	storage: persistent object: invalid state
301	ecc: sign data failed
302	ecc: verify sign failed
303	ecc: verify sign failed
304	handler: invalid json schema
305	handler: unknown command
306	handler: unsupported command in this tok state
307	handler: invalid usb packet
308	handler: too big usb packet
309	handler: invalid crc in usb packet
310	handler: unknown ticket type
311	handler: base64 encoding error
312	handler: ticket rejected by server
313	handler: check max string size failed
314	usb: too big response
315	usb: invalid data len in usb packet
316	usb: invalid start byte in usb packet
317	usb: invalid end byte in usb packet
318	usb: too small packet
319	crypto: aes encrypt/decrypt failed
320	crypto: ecp write error
321	crypto: ecp read error
322	crypto: edch failed
323	base64: decode error
324	crc16: check failed
325	item names: signing fault
326	production UUID: wrong or incomplete
327	packed: encode error
328	packed: decode error
329	invalid time

330	handler: base58 decoding error
331	handler: base58 encoding error
332	production: data: invalid
401	invalid login session
403	invalid user role
404	incorrect pin code
405	pin locked too many invalid pins
406	incorrect puk code
407	puk locked too many invalid puks
408	cashregister_factory_number is needed
501	invalid document type
502	c memory allocation error
503	document: invalid shift status
504	document: invalid shift duration
505	document: invalid previous document number
601	cashbox number != value from company certificate
602	cashregister number != value from company certificate
603	document time must be longer than the previous one
604	document time not in permitted interval
605	document counter is overflow
606	invalid currency name
607	invalid total sum
608	too long cashier string
609	invalid length of item code string
610	items array is empty
611	invalid items sum
612	invalid item code type
613	invalid vat amounts total
614	invalid total sum in one item
615	invalid parent doc length
616	invalid percentages
617	invalid item margin price
618	invalid item margin sum
619	invalid item margin price quantity
620	too many items
621	expected no documents on the storage
622	invalid length of item marking code
623	ticket: invalid request token
624	audit: invalid status value
625	ticket: invalid factory number
626	invalid usb type value
627	sale doc for rollback cannot be found
628	you have entered a wrong parent doc
629	invalid change sum value
630	contract field has invalid length
631	payer name field has invalid length
632	credit/prepay sum is forbidden
633	invalid moneyback type value
634	invalid residue sum
635	invalid item name len
636	invalid comminication secret type
637	invalid secured request
638	total sum mismatch the parent document
639	the parent doc you have entered has been already rolled back
640	item names hash length is invalid
641	invalid parents quantity
642	invalid datetime value
643	car number field has invalid length
644	invalid latitude value
645	invalid longitude value
646	invalid length of RRN

647	invalid length of number of a bonus card
648	RRN cannot be used without cashless funds
649	bonus card cannot be used for this document type
650	document wasn't found on server
651	gas station fields partially filled
652	invalid length of fdp in gas station info
653	invalid length of pump in gas station info
654	invalid length of gun in gas station info
655	invalid length of reservoir in gas station info
656	mandatory fields are missong in committent info
657	committent tin must be string of 10 digits
658	invalid length of committent name
659	invalid length of agent contract in committent info
660	invalid agent commission percentage in committent info
701	generate csr failed
702	cert parsing error
703	invalid primary key type in the certificate
704	invalid primary key curve type in the certificate
705	invalid signature type in the certificate
706	certificate is already expired
707	failed to init mbedtls library
708	convert public key failed
709	convert primary key failed
710	keypairs are not equial
711	failed to verify certificate chain
712	failed to parse custom oid
713	uuid oid not found in the certificate
714	token uuid is not equal to uuid from certificate
715	invalid common name in the certificate
716	common name is not found
717	invalid common name in the certificate
718	updated certificate not after old certificate
719	updated certificate is not in permitted time interval
720	generate fake company ca certificate failed (for test only)
721	generate fake server signature failed (for test only)
722	there is an invalid BEGIN/END header/footer in the PEM chain
801	transacted object has invalid id
802	transaction is not started
900	offset of all cbor errors
901	cbor: unknown error
902	cbor: unknown field length
903	cbor: advance past eof
904	cbor: io error
905	cbor: it is a garbage at the end of container
906	cbor: unexpected eof
907	cbor: unexpected break
908	cbor: unknown unknown type
909	cbor: illegal value type
910	cbor: illegal number
911	cbor: illegal simple type
912	cbor: unknown simple type
913	cbor: unknown tag
914	cbor: inappropriate tag for type
915	cbor: duplicate object keys
916	cbor: invalid utf8 text string
917	cbor: excluded type
918	cbor: excluded value
919	cbor: improper value
920	cbor: overlong encoding
921	cbor: map key is not string
922	cbor: map is not sorted

923	cbor: map keys are not unique
924	cbor: too many items
925	cbor: too few items
926	cbor: data is too large
927	cbor: esting is too deep
928	cbor: unsupported type
929	cbor: internal error
930	cbor: out of memory
65500	unknown error
0	Successful operation
1000	unknown token error
1001	json: deserialize end of the input is missing
1002	json: input is not recognized
1003	json: deserialize: no memory
1004	json: not supported by the parser
1005	json: nesting limit was reached
1006	json: unknown parser error
1007	json: array deserialize error
1008	json: type: deserialize error
1009	json: type: serialize error
1010	json: type: overflow
1011	json: null raw!: deserialize error
1012	json: time: deserialize error
1013	json: time: serialize error
1014	cbor: serialize error
1015	cbor: deserialize error
1016	base64: encode error
1017	base64: decode error
1018	base58: encode error
1019	base58: decode error
1020	library: is not initialized
1021	json: deserialize error: empty input
1022	json: deserialize error: value isn't a float
1023	json: deserialize error: negative value in unsigned field
1101	hid open failed
1102	hid open failed: check product failed
1103	device hasn't been initialized
1104	too big paket to send
1105	usb write error
1106	usb read error
1107	invalid start byte recieved
1108	invalid packet size recieved
1109	invalid end byte recieved
1110	CRC check failed
1111	https: exec request failed
1112	https: invalid http status
1113	config: cannot open file
1114	config: file not found
1115	database: sqlite: statement couldn't be processed
1116	database: sqlite: rows doesn't exist
1117	config: bad config-file content
1118	config: saving file error
1119	IO: file: read/write error
1120	FD: update: not allowed to remove Fiscal Driver! Please send item names before.
1121	X509: unable to extract cert's serial number
1200	unknown operation id or operation version
1201	invalid operation id
1202	invalid operation version
1203	invalid document type
1204	invalid timestamp format
1205	Document creating has been completed successfully, but it was unable to handle Item Names properly

1206	There is an illegal character(s) in item name(s)
1207	FW update was not performed: (Target version) == (Current version)
1208	FW update was performed, but the version has not been changed
1209	FW update is going on
1210	There is at least a one doc, that should be sent. Please send it and try again.
1301	attestation: invalid file permissions
1302	attestation: read file error
1303	attestation: verify soft checksum error
1304	attestation: directory not found
1305	attestation: config error
1999	unknown pks error