Nexo documentation

This document contains information about data structure of NEXO protocol, as well as examples of requests and responses.
Version of NEXO Sale Protocol to which this documentation refers is **3.1**.

IMPORTANT: Before using the NEXO, please check whether the terminal protocol is set to NEXO in the Settings -> Transaction settings -> ECR protocol.

Version control

Version of this documentation is **2.11**.

Version	Date	Changes	Author
2.0	28.11.2022	Creation of documentation version 2.0	ML
2.1	14.12.2022	Tweak of kotlin intent call example	ML
2.2	05.01.2023	Rework of kotlin intent call example	ML
2.3	10.01.2023	Added more detailed explanation of `SignatureRequired` parameter in `ProprietaryTags` (in Payment response)	ML
2.4	11.01.2023	Added second example of calling POI in Java using `registerForActivityResult`	ML
2.5	19.04.2023	Added description of asynchronous call method of POI	ML
2.6	20.04.2023	Added `MerchantDisplayOutput` to the TransactionStatusResponse	ML

Version	Date	Changes	Author
2.7	01.06.2023	Added additional description for `SaleId`, `POIID`, `MessageClass`. FAQ section added	ML
2.8	16.11.2023	Fix of payment request schema - `VariableSymbol` was missing in `ProprietaryTags`	ML
2.9	18.03.2024	Fix of duplicit chapers. Added mock responses info in the Error handling section and FAQ	ML
2.10	18.03.2024	Added `Tip` transaction type in reconciliation (from payment app version `4.10.0`)	ML
2.11	09.04.2024	Added information about Intent launch of payment app (`queries` tag needs to be added to AndroidManifest in Android 11+)	ML

Glossary

	Description
ECR	Electronic cash register
POS	Point of Sale. This is the cash register (ECR), AKA Sale system. This term may refer to the hardware or the software of the POS.
POI	Point of Interaction. This is the payment terminal, AKA PED.
PED	PIN Entry Device. Same as POI.
NEXO	A standard protocol for communication between POS and POI. See nexo-standards.org
JSON	JavaScript Object Notation

	Description
HTTP	HyperText Transfer Protocol.The Nexo messages will be sent using HTTP over TCP.
POIID	String that identifies POI. Generally we are using TID
TID	Terminal identifier - string identifier assigned by card acquirer to the merchant
SN	Serial Number. String that uniquely identifies terminal (physical device)

Communication protocol

There are 3 ways POS can communicate with the POI:

- HTTP POST call
- Android Intent call
- Cloud (Payment Middleware) call

HTTP POST call

Default **port** of NEXO HTTP service on POI is **7500**.

> Please ensure that the the devices (POS and POI) are on the same network before testing.

Each HTTP request must contain following headers:

```
Content-Type:application/json
Content-Length: <the length of the JSON request string (the HTTP body)>
Authorization: Basic base64(<username>:<password>)
```

Example:

```
Content-Type:application/json
Content-Length: 1024
Authorization: Basic dXNlcjpwYXNz
```

Authentication

Authentication of caller is performed by HTTP Basic Authentication. For more information about HTTP Basic authentication, please visit this site.

- Default username: **user**
- Default password: **pass**

Example of HTTP Authentication header:

```
Authorization: Basic dXNlcjpwYXNz
```

Kotlin example of HTTP NEXO call:

```
val ip = "127.0.0.1"
val port = 7500
val auth = "dXNlcjpwYXNz"
val url = URL("http://$ip:$port")
try {
    val connection = url.openConnection() as HttpURLConnection
    this.httpConnection = connection
    connection.requestMethod = "POST"
    connection.setRequestProperty("Authorization", "Basic $auth")
    connection.setRequestProperty("Content-Type", "application/json")
    connection.doInput = true
    connection.doOutput = true
    connection.connectTimeout = 1000
    connection.readTimeout = 60_000

    val os = connection.outputStream
    os.write(request.toByteArray())
    os.close()

    val response = connection.inputStream.readBytes()
        //TODO process response
} catch (e: IOException) {
    //TODO Process error
}
```

Android Intent call

If POI and POS are installed on the same Android device, you can call the POI using Android Intent. The authentication in this case is not required.

Kotlin example:

First step is to register for activity result.
IMPORTANT: you need to call `registerForActivityResult()` before the Fragment or Activity is created. In case of Fragment the last method you can call `registerForActivityResult()` is `onViewCreated()` (For more information see Fragment lifecycle). In case of Activity you need to call it in `onCreate()`.

```
activityResultLauncher = registerForActivityResult(
        ActivityResultContracts.StartActivityForResult()
) { activityResult ->
  val nexoResponse = activityResult?.data?.getStringExtra("NEXO_RESPONSE")
    if(nexoResponse != null) {
        //TODO process NEXO response
  } else {
        //TODO process error state
  }
}
```

You can then use `activityResultLauncher` to launch the Intent when necessary.
IMPORTANT: you cannot launch the `ActivityResultLauncher` until the fragment or activity's `Lifecycle` has reached `CREATED`
IMPORTANT: Since Android 11 there is a behavior change that some apps will not provide info via `getLaunchIntentForPackage()` unless you add `queries` tag to the AndroidManifest like this:

```
    <queries>
        <package android:name="com.example.app" />
```

```
    </queries>
```

Example:

```
val intent = context.packageManager.getLaunchIntentForPackage("<package name>")
if(intent != null) {
    intent.putExtra("NEXO_REQUEST", gson.toJson(nexoRequest))
    intent.flags = 0
    activityResultLauncher.launch(intent)
} else {
    //TODO payment app was not found
}
```

Java example 1:

> The old way of launching Intent. You may get a warning that `startActivityForResult` is deprecated. If this is problem for you, please follow instructions of **Java example 2**

IMPORTANT: Since Android 11 there is a behavior change that some apps will not provide info via `getLaunchIntentForPackage()` unless you add `queries` tag to the AndroidManifest like this:

```
    <queries>
        <package android:name="com.example.app" />
    </queries>
```

Example:

```
final Intent intent = context.getPackageManager().getLaunchIntentForPackage("<pa
if(intent != null) {
        intent.putExtra("NEXO_REQUEST", gson.toJson(nexoRequest));
        intent.setFlags(0);
        activity.startActivityForResult(
                intent,
                REQUEST_CODE
        );
} else {
```

```
        //TODO payment app was not found
}
```

You will receive response in the `onActivityResult`:

```
@Override
protected void onActivityResult(int requestCode, int resultCode, Intent data) {
        super.onActivityResult(requestCode, resultCode, data);
        if(requestCode == REQUEST_CODE && resultCode == Activity.RESULT_OK && da
                String response = data.getStringExtra("NEXO_RESPONSE");
                if(response != null) {
                        //TODO process response
                } else {
                        //TODO process error
                }
        } else {
                //TODO
        }
}
```

Java example 2:

The new way of launching Intent consists of multiple steps. First step is to register for activity result.

IMPORTANT: you need to call `registerForActivityResult()` before the Fragment or Activity is created. In case of Fragment the last method you can call `registerForActivityResult()` is `onViewCreated()` (For more information see Fragment lifecycle). In case of Activity you need to call it in `onCreate()`.

```
activityResultLauncher = registerForActivityResult(
        new ActivityResultContracts.StartActivityForResult(),
                activityResult -> {
                        if(activityResult == null) {
                                //TODO process error
                                return;
                        }
                        final Intent data = activityResult.getData();
                        if(data == null) {
```

```
                        //TODO process error
                        return;
                }
                final String nexoResponse = data.getStringExtra("NEXO_RESPONSE"
                if (nexoResponse == null) {
                        //TODO process error
                } else {
                        //TODO process NEXO response
                }
        }
);
```

You can then use `activityResultLauncher` to launch the Intent when necessary.
IMPORTANT: you cannot launch the `ActivityResultLauncher` until the fragment or activity's `Lifecycle` has reached `CREATED`
IMPORTANT: Since Android 11 there is a behavior change that some apps will not provide info via `getLaunchIntentForPackage()` unless you add `queries` tag to the AndroidManifest like this:

```
    <queries>
        <package android:name="com.example.app" />
    </queries>
```

Example:

```
Intent intent = getPackageManager().getLaunchIntentForPackage("<package name>");
if(intent != null) {
        intent.setFlags(0);
        intent.putExtra("NEXO_REQUEST", request);
        activityResultLauncher.launch(intent);
} else {
        //TODO payment app was not found
}
```

Cloud (Payment Middleware) call

If you want the POS to communicate with POI on different networks, you can use cloud call of our Portal.

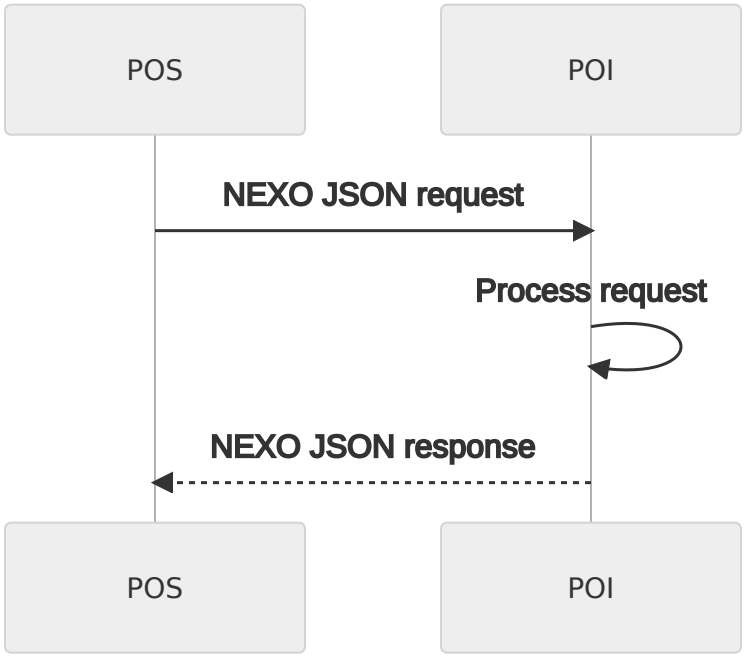
> Cloud call of POI requires Launcher app to be installed on terminal

Architecture

Communication between POS (ECR) and POI (payment terminal) is always initiated by POS. There are 2 ways of communication:

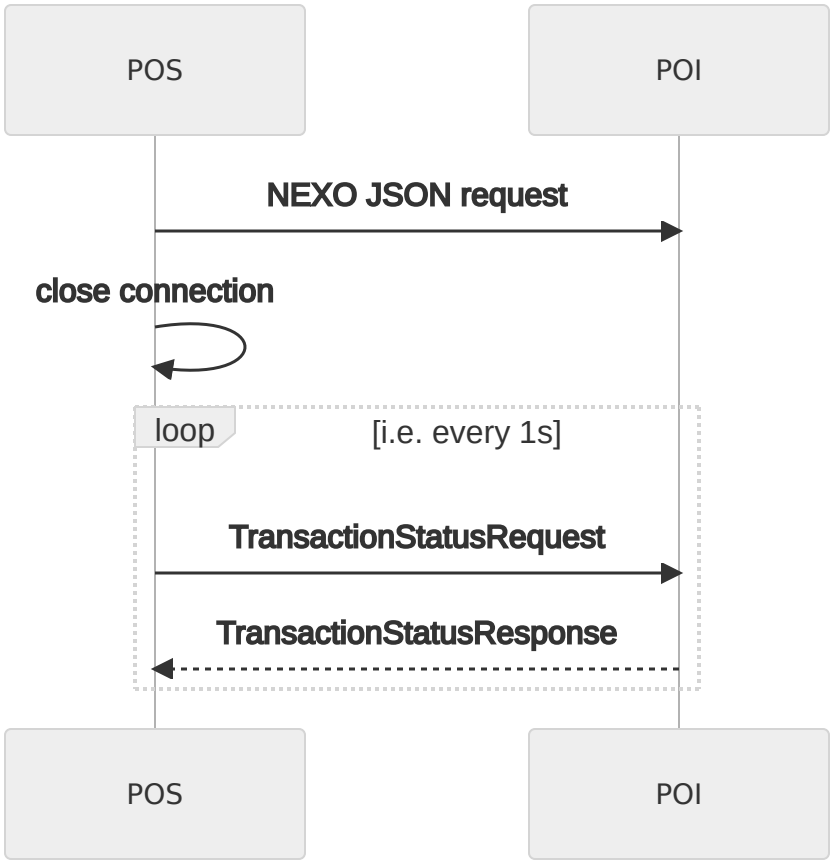
1. **Synchronous** - connection is opened until response is sent back, or connection is broken:

> For information about how to handle communication errors, please refer to the Error handling chapter.



2. **Asynchronous** - request is sent to POI and connection is closed by POS. POS is then querying the POI for the status of the transaction (via TransactionStatusRequest), until transaction is finished.

> If you want to show transaction status message to the merchant (i.e. when the POI faces customer and merchant cannot see it), the `TransactionStatusResponse` contains (starting from version 3.35.2 of Payment app) `MerchantDisplayOutput` string about status of the transaction on the POI (i.e. “Waiting for card”, “Insert PIN”, “Sending data to host”).



NEXO message structure

Header

Each NEXO request and response contains **MessageHeader** object.
Example:

```
"MessageHeader": {
        "MessageCategory": "Payment",
        "MessageClass": "Service",
        "MessageType": "Request",
        "POIID": "TID001",
        "ProtocolVersion": "3.1",
        "SaleID": "123456",
        "ServiceID": "62052376-280f-47d2-a012-729b2f814791"
}
```

IMPORTANT: Service ID must be unique for the message pair. It is recommended to use

UUID as ServiceID

All of the fields in MessageHeader are **mandatory** in both request and response.
Meaning of header fields:

- **MessageCategory**

  `format: EnumString`

  Specifies intended action to be performed in POI. Supported values are:

  - Payment - start payment/refund transaction
  - Input - get information from user (confirmation, or asking question)
  - Diagnosis - get information whether terminal functionality is ready/working (printer, card readers, communication with host, etc.)
  - Reconciliation - settlements
  - TransactionStatus - status of previous transactions
  - Reversal - payment reversal
  - Admin - reboot device remotely
  - Abort - aborting transaction in progress

- **MessageClass**

  `format: EnumString`

  Specifies initiator of operation. Supported values are:

  - Device - operation initiated by POI (currently not implemented)
  - Service - operation initiated by POS/ECR

- **MessageType**

  `format: EnumString`

  Defines, whether the message is request, or response. Supported values:

  - Request
  - Response

- **POIID**

  `format: String, length: <1, 128>`

  Identifier of POI. Generally, the merchant TID (terminal ID assigned to merchant by acquirer) is used, because multiple merchants may use the same payment app, so the

TID is used to identify the target merchant. You can find the TID in About screen of the payment app.
In most cases though, the payment app is used only by single merchant - in this case you can use device hardware/software identifier (i.e. device serial number) as POIID.

> Since version 3.35.0, the Payment app contains Content provider, which allows you to get current TID, and also list of available TIDs (if multimerchant is supported). You can find more detailed information here.

- **ProtocolVersion**
  `format: String`
  Constant value of "3.1"
- **SaleID**
  `format: String, length: <1, 128>`
  Identification of the Sale system (POS/ECR). It is used to identify, from which POS/ECR the request came from. For example hardware serial number of POS/ECR, or id of POS/ECR software installation. If you are somehow missing this information, please create SaleID yourself in a way, so that you can identify the POS/ECR in future.

> SaleID is used mainly for debugging - if you receive issue report containing request/response JSON, you can identify specific ECR/POS using SaleID

- **ServiceID**
  `format: String, length: <1, 128>`
  Identification of message pair. POI will return the same ServiceID in response as is in request.
  IMPORTANT: Service ID must be unique for the message pair. It is recommended to use UUID or GUID as ServiceID

Request

Structure of NEXO request is as follows:

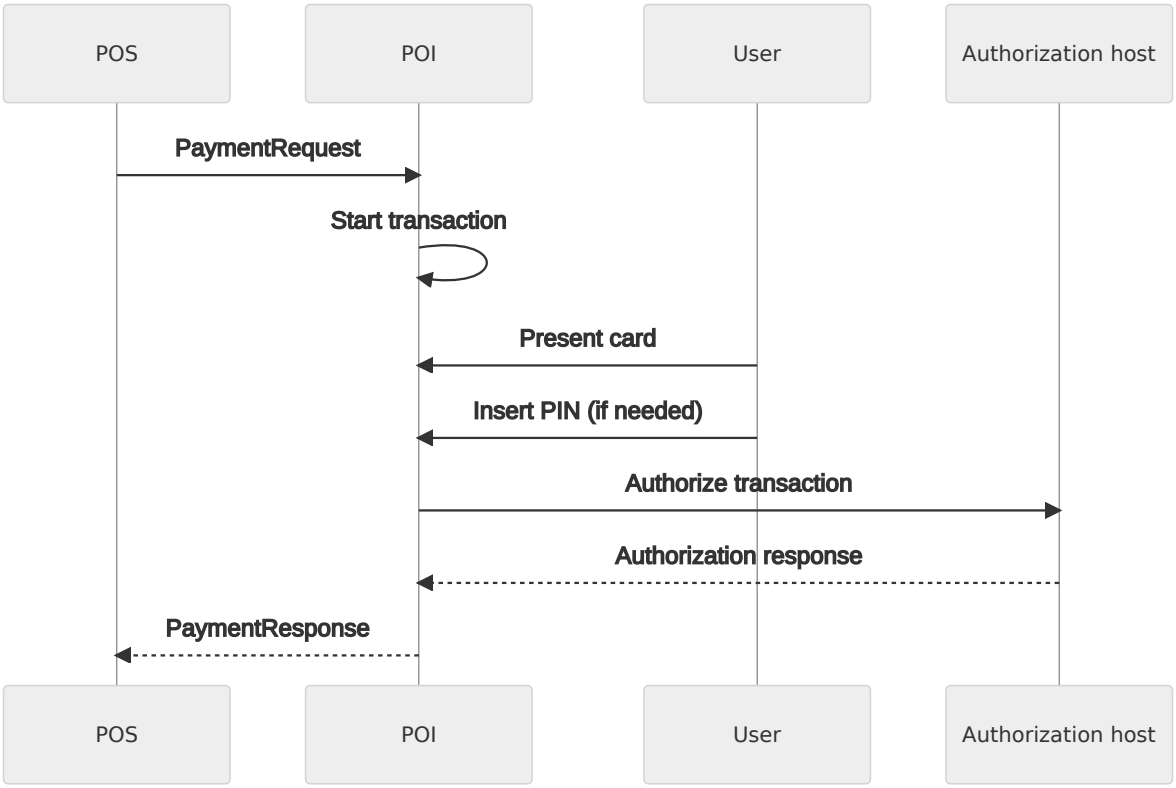
```
{
        "SaleToPOIRequest": {
                "MessageHeader": {
                        ...
```

```
                }
                ...
            }
        }
```

The content of the request depends on what the MessageCategory is.

Payment (Sale/Refund)

Request is initiated by POS with intent to start payment transaction - either sale, or refund.



Request

Detailed request JSON schema is here.
Payment request must contain **PaymentRequest** object.

Sale

In **sale** payment request the object PaymentData is optional. If the PaymentData object is not part of the message, POI will treat it as a sale request. Example of payment **sale** request:

```
{
        "SaleToPOIRequest": {
                "MessageHeader": {
                        "MessageCategory": "Payment",
                        "MessageClass": "Service",
                        "MessageType": "Request",
                        "POIID": "TID001",
                        "ProtocolVersion": "3.1",
                        "SaleID": "123456",
                        "ServiceID": "62052376-280f-47d2-a012-729b2f814791"
                },
                "PaymentRequest": { /* mandatory */
                        "PaymentData": { /* PaymentData is optional in case of s
                                "PaymentType": "Normal" /* optional */
                        },
                        "PaymentTransaction": { /* mandatory */
                                "AmountsReq": { /* mandatory */
                                        "Currency": "EUR", /* mandatory */
                                        "RequestedAmount": 14.0, /* mandatory */
                                        "TipAmount": 1.0 /* optional */
                                }
                        },
                        "SaleData": { /* mandatory */
                                "SaleTransactionID": { /* mandatory */
                                        "TimeStamp": "2021-11-23T15:53:08.303+01
                                        "TransactionID": "8c9c7cbf-abdd-447d-ac7
                                }
                        }
                }
        }
}
```

Refund

In **refund** payment request the object PaymentData is **mandatory**, otherwise POI will treat it as a sale request. Example of payment **refund** request:

```
{
        "SaleToPOIRequest": {
                "MessageHeader": {
                        "MessageCategory": "Payment",
                        "MessageClass": "Service",
                        "MessageType": "Request",
                        "POIID": "TID001",
                        "ProtocolVersion": "3.1",
                        "SaleID": "123456",
                        "ServiceID": "62052376-280f-47d2-a012-729b2f814791"
                },
                "PaymentRequest": { /* mandatory */
                        "PaymentData": { /* mandatory */
                                "PaymentType": "Refund" /* mandatory */
                        },
                        "PaymentTransaction": { /* mandatory */
                                "AmountsReq": { /* mandatory */
                                        "Currency": "EUR", /* mandatory */
                                        "RequestedAmount": 14.0 /* mandatory */
                                }
                        },
                        "SaleData": { /* mandatory */
                                "SaleTransactionID": { /* mandatory */
                                        "TimeStamp": "2021-11-23T15:53:08.303+01
                                        "TransactionID": "8c9c7cbf-abdd-447d-ac7
                                }
                        }
                }
        }
}
```

Response

Detailed response JSON schema is here.
Response contains result of transaction.
The transaction is considered successful only when the parameter **PaymentResponse.Response.Result** has value **“Success”**, otherwise, the transaction failed.

Brief description of structure of PaymentResponse (for more detailed information, please refer to the JSON schema here):

- PaymentResult - contains information about amounts, card and payment data
  - AmountsResp
    - AuthorizedAmount - amount authorized by POI
    - Currency - currency of transaction
  - PaymentInstrumentData
    - CardData
      - EntryMode - enumeration. Supported values: `MagStripe`, `ContactlessMSR`, `ContactlessEMV`, `ICC`, `Keyed`, `RFID`
      - MaskedPan - masked card number. Only first 2 and last 4 digits are visible, all other digits are shown as `#`
  - PaymentType - `Normal` of `Refund`, depends on the type of payment in request
  - Proprietary tags - additional information about transaction
    - Aid - card application identifier (for more information see this article)
    - ApplicationName - card application name
    - AuthorizationCode - authorization code from authorization host
    - Brand - brand of the card
    - CardPresentationMethod - specifies card input type
    - ReceiptDate - date printed on receipt
    - SequenceNumber - sequence number of transaction authorization host
    - SignatureRequired - whether the merchant should require customer to sign the merchant receipt
    - > **Attention**: when signature verification **unsuccessful** (i.e., merchant cannot verify the signature is valid), the POS is responsible to call reversal on POI

        - TerminalID - merchant terminal identification
- POIData - information about transaction ID in POI and time of transaction in POI
- Response - contains information whether the transaction was successful
    - Result - contains information whether the transaction was successful (either Success or Failure)
    - ErrorCondition - enumeration value identifying error that occurred. For more information please refer to the Error handling chapter
    - AdditionalResponse - string explanation of error (for logging purpose only - do not show this value to customer)
- SaleData - contains SaleTransactionID sent in the request
- PaymentReceipt - contains customer (CustomerReceipt) and merchant (CashierReceipt) receipt with line by line text to be printed

Response example:


```
{
        "SaleToPOIResponse": {
                "MessageHeader": {
                        "MessageCategory": "Payment",
                        "MessageClass": "Service",
                        "MessageType": "Response",
                        "POIID": "TID001",
                        "ProtocolVersion": "3.1",
                        "SaleID": "123456",
                        "ServiceID": "62052376-280f-47d2-a012-729b2f814791"
                },
                "PaymentResponse": {
                        "PaymentResult": {
                                "AmountsResp": {
                                        "AuthorizedAmount": 14.0,
                                        "Currency": "EUR"
                                },
                                "PaymentInstrumentData": {
                                        "CardData": {
                                                "EntryMode": "ContactlessEMV",
                                                "MaskedPan": "45##-####-####-612
                                        },
                                        "PaymentInstrumentType": "Card"
                                },
```

```
                "PaymentType": "Normal",
                "ProprietaryTags": {
                        "Aid": "A0000000031010",
                        "ApplicationName": "Visa Debit",
                        "AuthorizationCode": "123456",
                        "Brand": "VISA",
                        "CardPresentationMethod": "CLESS",
                        "ReceiptDate": "2021-11-23T15:53:28.110+
                        "SequenceNumber": "1234567890",
                        "SignatureRequired": false,
                        "TerminalId": "TID001"
                }
        },
        "POIData": {
                "POITransactionID": {
                        "TimeStamp": "2021-11-23T15:53:28.110+01
                        "TransactionID": "99523506-e118-4dc0-913
                }
        },
        "Response": {
                "Result": "Success"
        },
        "SaleData": {
                "SaleTransactionID": {
                        "TimeStamp": "2021-11-23T15:53:08.303+01
                        "TransactionID": "8c9c7cbf-abdd-447d-ac7
                }
        },
        "PaymentReceipt": [
                {
                        "DocumentQualifier": "CustomerReceipt",
                        "OutputContent": {
                                "OutputFormat": "Text",
                                "OutputText": [
                                        {
                                                "Text": "-------
                                        },
                                        {
                                                "Text": "24.11.2
                                        },
                                        {
                                                "Text": "Receipt
```

```
        },
        {
            "Text": "-------
        },
        {
            "Text": ""
        },
        {
            "Text": ""
        },
        {
            "Text": ""
        },
        {
            "Text": ""
        },
        {
            "Text": "IČO:"
        },
        {
            "Text": "-------
        },
        {
            "Text": "TID:TID
        },
        {
            "Text": "Karta:4
        },
        {
            "Text": "
        },
        {
            "Text": "AID: A0
        },
        {
            "Text": "Visa De
        },
        {
            "Text": " "
        },
        {
            "Text": " "
```

```
                                        },
                                        {
                                                "Text": "Predaj"
                                        },
                                        {
                                                "Text": "14.00 E
                                        },
                                        {
                                                "Text": " "
                                        },
                                        {
                                                "Text": " "
                                        },
                                        {
                                                "Text": "Autoriz
                                        },
                                        {
                                                "Text": "Sekvenč
                                        },
                                        {
                                                "Text": " "
                                        },
                                        {
                                                "Text": "-------
                                        },
                                        {
                                                "Text": " "
                                        },
                                        {
                                                "Text": " "
                                        }
                                ]
                        },
                        "RequiredSignatureFlag": false
                },
                {
                        "DocumentQualifier": "CashierReceipt",
                        "OutputContent": {
                                "OutputFormat": "Text",
                                "OutputText": [
                                        {
                                                "Text": "-------
```

```
        },
        {
                "Text": "24.11.2
        },
        {
                "Text": "Receipt
        },
        {
                "Text": "-------
        },
        {
                "Text": ""
        },
        {
                "Text": ""
        },
        {
                "Text": ""
        },
        {
                "Text": ""
        },
        {
                "Text": "IČO:"
        },
        {
                "Text": "-------
        },
        {
                "Text": "TID:TID
        },
        {
                "Text": "Karta:4
        },
        {
                "Text": "
        },
        {
                "Text": "AID: A0
        },
        {
                "Text": "Visa De
```

```
        },
        {
                "Text": " "
        },
        {
                "Text": " "
        },
        {
                "Text": "Predaj"
        },
        {
                "Text": "14.00 E
        },
        {
                "Text": " "
        },
        {
                "Text": " "
        },
        {
                "Text": "Autoriz
        },
        {
                "Text": "Sekvenč
        },
        {
                "Text": " "
        },
        {
                "Text": "-------
        },
        {
                "Text": " "
        },
        {
                "Text": " "
        },
        {
                "Text": "Kópia"
        },
        {
                "Text": " "
```

```
                                        },
                                        {
                                                "Text": " "
                                        }
                                ]
                        },
                        "RequiredSignatureFlag": false
                }
        ]
    }
  }
}
```

Reversal

Request is initiated by POS with intent to start reversal of previous payment transaction.

Request

Detailed request JSON schema is here.

Brief description of structure of ReversalRequest (for more detailed information, please refer to the JSON schema here):

- OriginalPOITransaction
  - POITransactionID
    - TransactionID - contains transaction ID of previous transaction. It can be found in PaymentResponse:
      `PaymentResponse.POIData.POITransactionID.TransactionID`
    - ReversalReason - enumeration, specifies the reason of reversal. Supported values are
      `CustCancel` - customer requested reversal
      `MerchantCancel` - merchant requested reversal
      `Malfunction` - suspected malfunction
      `Unable2Compl` - card acceptor device unable to complete transaction

Example:


```
{
        "SaleToPOIRequest": {
                "MessageHeader": {
                        "MessageCategory": "Reversal",
                        "MessageClass": "Service",
                        "MessageType": "Request",
                        "POIID": "TID001",
                        "ProtocolVersion": "3.1",
                        "SaleID": "123456",
                        "ServiceID": "338c63c5-74b2-4f13-a0d3-99fe81452d0a"
                },
                "ReversalRequest": {
                        "OriginalPOITransaction": {
                                "POITransactionID": {
                                        "TransactionID": "99523506-e118-4dc0-913
                                }
                        },
                        "ReversalReason": "CustCancel"
                }
        }
}
```


Response

Detailed response JSON schema is here.

The reversal is considered successful only in case the `ReversalResponse.Response.Result` value is “Success”

Example:


```
{
    "SaleToPOIResponse": {
        "MessageHeader": {
            "MessageCategory": "Reversal",
            "MessageClass": "Service",
            "MessageType": "Response",
            "POIID": "TID001",
```

```
            "ProtocolVersion": "3.1",
            "SaleID": "123456",
            "ServiceID": "338c63c5-74b2-4f13-a0d3-99fe81452d0a"
        },
        "ReversalResponse": {
            "POIData": {
                "POITransactionID": {
                    "TimeStamp": "2021-11-23T16:28:36.910Z",
                    "TransactionID": "a460791c-a002-4843-a3ca-12e046221eb4"
                }
            },
            "Response": {
                "Result": "Success"
            }
        }
    }
}
```

Transaction Status

Request is initiated by POS with intent to retrieve status of previous or ongoing transaction.

Request

Detailed request JSON schema is here.
To retrieve information about transaction, you can use either

- **SaleID** of transaction (from `PaymentRequest.SaleData.SaleTransactionID.TransactionID`)
- or **ServiceID** of transaction (from `MessageHeader.ServiceID`)

Example:

```
{
    "SaleToPOIRequest": {
        "MessageHeader": {
            "MessageCategory": "TransactionStatus",
```

```
            "MessageClass": "Service",
            "MessageType": "Request",
            "POIID": "TID001",
            "ProtocolVersion": "3.1",
            "SaleID": "123456",
            "ServiceID": "22de8895-6271-480b-8735-a6ccd171951d"
        },
        "TransactionStatusRequest": {
            "MessageReference": {
                "SaleID": "3d0e8d9a-9a39-4020-89b9-1654d1d913b4"
            }
        }
    }
}
```

Response

Detailed response JSON schema is here.

If the transaction on which the request refers is still **ongoing**, the POI will respond with `ErrorCondition IN_PROGRESS`:

```
{
        "SaleToPOIResponse": {
                "MessageHeader": {
                        "MessageCategory": "TransactionStatus",
                        "MessageClass": "Service",
                        "MessageType": "Response",
                        "POIID": "123456",
                        "ProtocolVersion": "3.1",
                        "SaleID": "ECR123456",
                        "ServiceID": "3598d318-46f2-40df-bb83-713c57301fa7"
                },
                "TransactionStatusResponse": {
                        "Response": {
                                "AdditionalResponse": "Transaction in progress",
                                "ErrorCondition": "IN_PROGRESS",
                                /* MerchantDisplayOutput added in version 3.35.2
                                "MerchantDisplayOutput": "Odosielanie dát",
                                "Result": "Failure"
                        }
```

```
                }
        }
}
```

If transaction is **not found**, the POI will respond with `ErrorCondition` `INVALID_TRN_REFERENCE`:


```
{
        "SaleToPOIResponse": {
                "MessageHeader": {
                        "MessageCategory": "TransactionStatus",
                        "MessageClass": "Service",
                        "MessageType": "Response",
                        "POIID": "TID001",
                        "ProtocolVersion": "3.1",
                        "SaleID": "123456",
                        "ServiceID": "f1573d2d-7b44-438b-aac9-90c3408685e8"
                },
                "TransactionStatusResponse": {
                        "Response": {
                                "AdditionalResponse": "Invalid transaction refer
                                "ErrorCondition": "INVALID_TRN_REFERENCE",
                                "Result": "Failure"
                        }
                }
        }
}
```


If the the transaction is **found**, the POI will respond with success, and the message contains RepeatedMessageResponse:


```
{
    "SaleToPOIResponse": {
        "MessageHeader": {
            "MessageCategory": "TransactionStatus",
            "MessageClass": "Service",
            "MessageType": "Response",
            "POIID": "TID001",
            "ProtocolVersion": "3.1",
            "SaleID": "123456",
```

```
        "ServiceID": "22de8895-6271-480b-8735-a6ccd171951d"
    },
    "TransactionStatusResponse": {
        "RepeatedMessageResponse": {
            "MessageHeader": {
                "MessageCategory": "Payment",
                "MessageClass": "Service",
                "MessageType": "Response",
                "POIID": "TID001",
                "ProtocolVersion": "3.1",
                "SaleID": "123456",
                "ServiceID": "d2243162-65aa-42c6-a6e1-3ae076bb5218"
            },
            "RepeatedResponseMessageBody": {
                "PaymentResponse": {
                    "POIData": {
                        "POITransactionID": {
                            "TimeStamp": "2021-11-24T12:13:46.811Z",
                            "TransactionID": "0fdfad94-73d8-4a7f-8035-412fe2
                        }
                    },
                    "PaymentResult": {
                        "AmountsResp": {
                            "AuthorizedAmount": 5.0,
                            "Currency": "EUR"
                        },
                        "PaymentInstrumentData": {
                            "CardData": {
                                "EntryMode": "Contactless",
                                "MaskedPan": "44##-####-####-6218"
                            },
                            "PaymentInstrumentType": "Card"
                        },
                        "PaymentType": "Normal",
                        "ProprietaryTags": {
                            "Aid": "A0000000032010",
                            "ApplicationName": "VISA Prepaid",
                            "AuthorizationCode": "123456",
                            "Brand": "VISA",
                            "CardPresentationMethod": "CLESS",
                            "ReceiptDate": "2021-11-24T12:13:46.811Z",
                            "SequenceNumber": "1234567890",
```

```
                        "SignatureRequired": false,
                        "TerminalId": "TID001"
                    }
                },
                "Response": {
                    "Result": "Success"
                },
                "SaleData": {
                    "SaleTransactionID": {
                        "TimeStamp": "2021-11-24T12:13:46.362+0100",
                        "TransactionID": "3d0e8d9a-9a39-4020-89b9-1654d1
                    }
                }
            }
        }
    },
    "Response": {
        "Result": "Success"
    }
}
}
```


Reconciliation (settlement)

Request is initiated by POS with intent to start reconciliation (settlement).

Request

Detailed request JSON schema is here.
There are 2 supported types of reconciliation:

- SaleReconciliation - X report (overview settlement - the counters are not reset)
- AcquirerReconciliation - Z report (daily settlement - counters **are reset**)

Example:

```
{
    "SaleToPOIRequest": {
        "MessageHeader": {
            "MessageCategory": "Reconciliation",
            "MessageClass": "Service",
            "MessageType": "Request",
            "POIID": "TID001",
            "ProtocolVersion": "3.1",
            "SaleID": "123456",
            "ServiceID": "3830102f-86a0-4b96-b194-66c55bcfcdca"
        },
        "ReconciliationRequest": {
            "ReconciliationType": "SaleReconciliation"
        }
    }
}
```


Response

Detailed response JSON schema is here.

The response contains information about all the counters on the POI.
`PaymentInstrumentType` is always `Card`

> Section with **Tips** (ReconciliationResponse.TransactionTotals.PaymentTotals with TransactionType 'Tip') added in version `4.10.0` of the payment app

Example:


```
{
    "SaleToPOIResponse": {
        "MessageHeader": {
            "MessageCategory": "Reconciliation",
            "MessageClass": "Service",
            "MessageType": "Response",
            "POIID": "TID001",
            "ProtocolVersion": "3.1",
            "SaleID": "123456",
            "ServiceID": "3830102f-86a0-4b96-b194-66c55bcfcdca"
```

```
},
"ReconciliationResponse": {
    "POIReconciliationID": 1,
    "ReconciliationType": "SaleReconciliation",
    "Response": {
        "Result": "Success"
    },
    "TransactionTotals": [
        {
            "PaymentCurrency": "EUR",
            "PaymentInstrumentType": "Card",
            "PaymentTotals": [
                {
                    "TransactionAmount": 6.0,
                    "TransactionCount": 1,
                    "TransactionType": "Debit"
                },
                {
                    "TransactionAmount": 0.0,
                    "TransactionCount": 0,
                    "TransactionType": "Credit" //Refund transactions
                },
                {
                    "TransactionAmount": 0.0,
                    "TransactionCount": 0,
                    "TransactionType": "OneTimeReservation" //Preauthori
                },
                {
                    "TransactionAmount": 0.0,
                    "TransactionCount": 0,
                    "TransactionType": "CompletedReservation" //Preautho
                },
                {
                    "TransactionAmount": 0.0,
                    "TransactionCount": 0,
                    "TransactionType": "Failed"
                },
                { //added in version 4.10.0 of the payment app
                    "TransactionAmount": 0.0, //Amount of transaction wi
                    "TransactionCount": 0, //Number of transactions with
                    "TransactionType": "Tip" //Transactions with tip
                }
```

```
                    ]
                }
            ]
        }
    }
}
```

Diagnosis

Request is initiated by POS with intent to get diagnostic data about state of POI.

Request

Detailed request JSON schema is here.

In the request, you can set whether the POI should check connection to the authorization host in field `HostDiagnosisFlag` .

Example:

```
{
    "SaleToPOIRequest": {
        "MessageHeader": {
            "MessageCategory": "Diagnosis",
            "MessageClass": "Service",
            "MessageType": "Request",
            "POIID": "TID001",
            "ProtocolVersion": "3.1",
            "SaleID": "123456",
            "ServiceID": "168c5a30-3c76-4a1d-8cda-114c23d1709d"
        },
        "DiagnosisRequest": {
            "HostDiagnosisFlag": true
        }
    }
}
```

Response

Detailed response JSON schema is here.

The response contains following information:

- DiagnosisResponse
  - `Response.Result` - whether the diagnosis ended successfully
  - `HostStatus.IsReachableFlag` - whether authorization host is reachable
  - POIStatus
    - CardReaderOKFlag - whether the card readers are OK
    - CommunicationOKFlag - whether there is connection to the internet
    - GlobalStatus - enumeration, status of the POI. Possible values are:
      `OK` - The POI is ready to receive and process a request
      `Busy` - The POI Terminal cannot process a request because another processing is in
      progress
      `Maintenance` - The POI is in maintenance processing
      `Unreachable` - The POI is unreachable or not responding
    - PEDOKFlag - Indicates if the PED is working and usable
    - PoiReady - wheter POI is ready to receive request
    - PrinterOKFlag - whether the printer is working

Example:


```
{
    "SaleToPOIResponse": {
        "MessageHeader": {
            "MessageCategory": "Diagnosis",
            "MessageClass": "Service",
            "MessageType": "Response",
            "POIID": "TID001",
            "ProtocolVersion": "3.1",
            "SaleID": "123456",
            "ServiceID": "168c5a30-3c76-4a1d-8cda-114c23d1709d"
        },
        "DiagnosisResponse": {
```

```
            "HostStatus": {
                "IsReachableFlag": true
            },
            "POIStatus": {
                "CardReaderOKFlag": true,
                "CommunicationOKFlag": true,
                "GlobalStatus": "OK",
                "PEDOKFlag": true,
                "PoiReady": true,
                "PrinterOKFlag": true
            },
            "Response": {
                "Result": "Success"
            }
        }
    }
}
```


Administration request

Request is initiated by POS with intent to reboot the device.

Request

Detailed request JSON schema is here.

Example:


```
{
        "SaleToPOIRequest": {
                "MessageHeader": {
                        "SaleID": "123456",
                        "MessageClass": "Service",
                        "POIID": "TID001",
                        "ServiceID": "168c5a30-3c76-4a1d-8cda-114c23d1709d",
                        "MessageType": "Request",
                        "ProtocolVersion": "1.0",
                        "MessageCategory": "Admin"
```

```
                },
                "AdminRequest": {
                        "ServiceIdentification": "Reboot"
                }
        }
}
```

Response

The response is not specified. In case of successful request, the terminal will start rebooting.

Error handling

This chaper describes, how the error states should be handled.

Application errors

In general the NEXO request can end successfully or with an error. If request was successful, the NEXO response will contain “Result”: “Success” in Response object.

In case of failure, the NEXO response will contain “Result”: “Failure” in Response object. In this case, the response will also contain:

- **AdditionalResponse** – description why the request failed (only for logging purposes)
- **ErrorCondition** – enumeration of values, that caller may use to react to failure

Example:

```
{
        "SaleToPOIResponse": {
                "MessageHeader": {
                        "MessageCategory": "Payment",
                        "MessageClass": "Service",
                        "MessageType": "Response",
                        "POIID": "TID001",
                        "ProtocolVersion": "3.1",
```

```
                "SaleID": "123456",
                "ServiceID": "168c5a30-3c76-4a1d-8cda-114c23d1709d"
        },
        "PaymentResponse": {
                "Response": {
                        "AdditionalResponse": "Requested Transaction ID
                        "ErrorCondition": "TRN_ALREADY_EXISTS",
                        "Result": "Failure"
                }
        }
    }
}
```


Invalid JSON

If the request message is not a valid JSON, the POI will respond with the following

HTTP body:

[“Bad JSON:[line]: [message]”] **where [line] and [message] will contain the details of the error.**

Invalid NEXO request

If the request message is valid JSON but not a valid NEXO request, a response generated according to NEXO specification.

Example:


```
{
    "SaleToPOIResponse": {
        "MessageHeader": {
                "SaleID": "123456",
                "MessageClass": "Service",
                "POIID": "TID001",
                "ServiceID": "168c5a30-3c76-4a1d-8cda-114c23d1709d",
                "MessageType": "Response",
                "ProtocolVersion": "3.1",
                "MessageCategory": "Payment"
        },
```

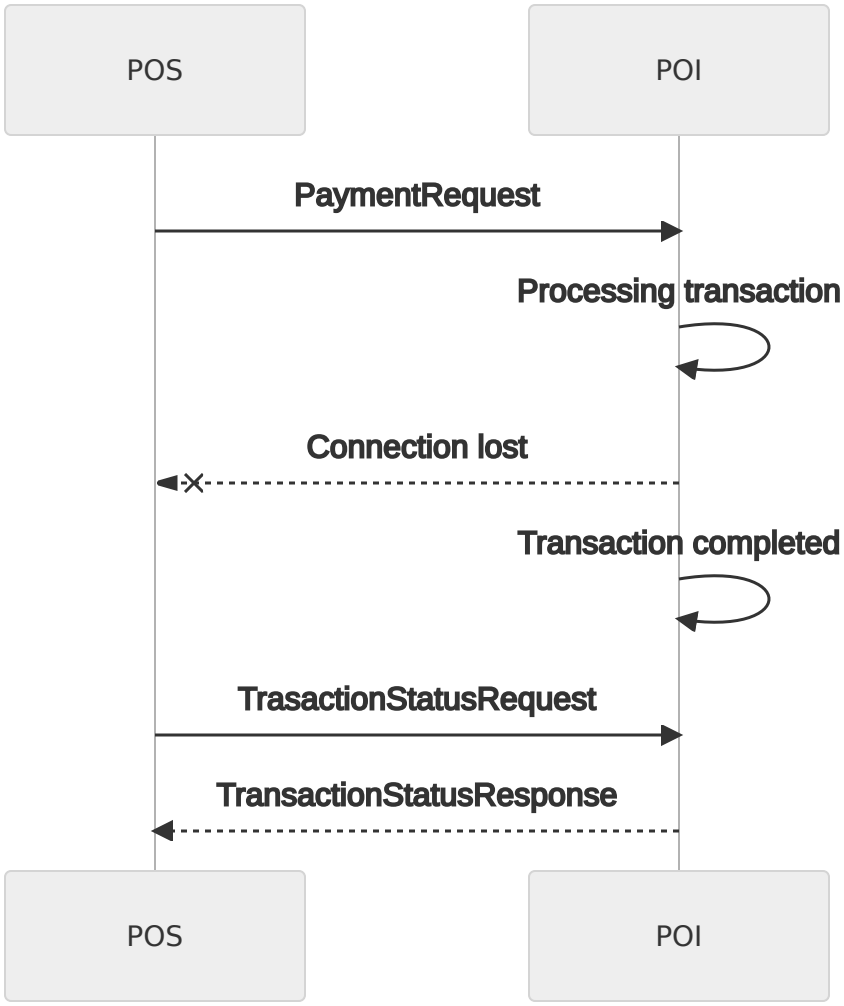
```
                "PaymentResponse": {
                        "Response": {
                                "Result": "Failure",
                                "ErrorCondition": "MESSAGE_FORMAT_ERROR",
                                "AdditionalResponse": "At top level,field SaleTO
                        }
                }
        }
}
```


Connection errors

During transaction processing, the connection between POS and POI could be lost at any time.

IMPORTANT: If the connection between POS and POI is lost without response reaching POS, the POS is responsible to call `TransactionStatus` request to obtain the result of the transaction

Example:



ErrorCondition enumeration

In the following table is a list of possible values, which caller may expect.

Value	Description
MESSAGE_FORMAT_ERROR	Request message has invalid format. For example, some mandatory fields of request are missing.
DEVICE_OUT_OF_ORDER	Device is not ready to process request. For example due to maintenance.
UNAVAILABLE_DEVICE	The hardware is not available
NOT_ALLOWED	A service request is sent during a

Value	Description
	Service dialogue. A combination of services not possible to provide. During the CardReaderInit message processing, the user has entered a card which has to be protected by the POI, and cannot be processed with this device request from the external, and then the Sale System.
UNAVAILABLE_SERVICE	The service is not available (not implemented, not configured, protocol version too old…)
LOGGED_OUT	Not logged in
BUSY	The system is busy, try later
ABORTED	The Initiator of the request has sent an Abort message request, which was accepted and processed.
CANCELLED	The user has aborted the transaction on the PED keyboard, for instance during PIN entering.
IN_PROGRESS	The transaction is still in progress and then the command cannot be processed
INSERTED_CARD	If the Input Device request a NotifyCardInputFlag and the Customer enters a card in the card reader without answers to the Input command, the POI abort the Input command processing, and answer a dedicated ErrorCondition value in the

Value	Description
	Input response message.
UNREACHABLE_HOST	Acquirer or any host is unreachable or has not answered to an online request, so is considered as temporary unavailable. Depending on the Sale context, the request could be repeated (to be compared with “Refusal”).
REFUSAL	The transaction is refused by the host or the rules associated to the card, and cannot be repeated.
INVALID_CARD	The card entered by the Customer cannot be processed by the POI because this card is not configured in the system
PAYMENT_RESTRICTION	Some sale items are not payable by the card proposed by the Customer.
WRONG_PIN	The user has entered the PIN on the PED keyboard and the verification fails.
CARD_REMOVED_PREMATURELY	User removed card before card operation finished.
CARD_READ_TIMEOUT	Timeout for card reading expired.
LICENSE_EXPIRED	License of application is expired.
CURRENCY_INVALID	Unknown currency code in request.
CURRENCY_DOES_NOT_MATCH_MERCHANT	Merchant does not support requested currency.

Value	Description
TRN_ALREADY_EXISTS	Transaction with specified TransactionID already exists.
REVERSAL_NO_REVERSIBLE_TRN	No reversible transaction was found.
NO_TRN_FOUND	The transaction is not found (e.g. for a reversal or a repeat)
INVALID_TRN_REFERENCE	Message reference for transaction status operation is missing

Testing NEXO errors

Starting from version 3.36.0, you can simulate various response codes, when the app is in the **MOCK mode**. For more info see this link.