nexo Retailer Listener

User guide

References

Reference	*Editor*	*Date or version*
1. Sale to POI specifications – Retailer protocol	Nexo	V3.0

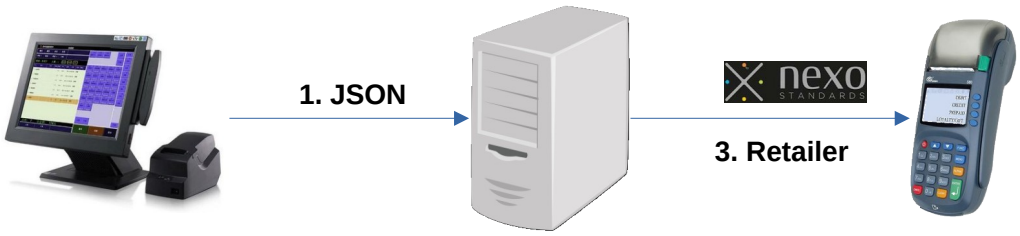
1. Introduction

The Listener is a service provided along with the nexo classes which can be found on Github at https://github.com/pmespace/PMS.NEXO.

It is an executable software which can run on any Windows, Linux[1], mac OS[2] computer supporting .NET 5.0 libraries.

Its role is to act as a gateway between any POS wishing to support nexo Retailer and any POI supporting nexo Retailer, avoiding for the POS to implement nexo Retailer.

The system works as follows :



Where:

1. **JSON**

   It is the interface between the POS and the Listener.

   That interface is simple JSON file sent over a socket opened by the Listener; the POS must open that socket and send the JSON file instructing what kind of service must be addressed, using nexo Retailer, to the POI (terminal)

   The JSON file format is described in §2.2.1 and §2.2.3.

1 *Tested under Linux Ubuntu, Linux Mint (based on Ubuntu).*

2 *The software has not been tested under mac OS so far.*

2. **Listener**

   The Listener is an executable piece of software which opens a socket to receives commands from various POS and returns the result to the calling process. Commands are then outgoing (sent to the POI) and incoming (received from the POI). A description of both these commands can be found in §2.2.1 and §2.2.3.

3. **nexo Retailer**

   The nexo Retailer protocol itself, generated and understood by the Listener to send orders to the POI and receive results. A full description of this protocol can be found in [Sale to POI specifications – Retailer protocol].

The advantages of the Listener are therefore:

- To provide an easy abstraction of nexo Retailer without having to manipulate all its data and structures.
- The ability to deploy the Listener on a the local computer running the POS software or on a server allowing to centralise its processing and serving multiple POS.
- The ability to easily modify the data to send or receive to and from the POI.

The way to use the Listener is as follows:

Listener is started on a computer

1. The Listener uses a JSON settings file. This file is described in §2.1.
2. The POS identifies the service it wants to operates on the POI (Login, Payment,…), retrieves the data it wants to set inside the request message, identifies which POI it wants to address and creates the JSON file to send over to the Listener.
3. The POS opens the socket created by the Listener and sends the command (under a JSON file) and waits for the answer.
4. The Listener receives the command, creates a nexo Retailer service as requested by the POS, sets the various data inside and sends it to the POI.
5. The POI receives the request, processes it and returns the result to the caller (the Listener).
6. The Listener receives the answer from the POI; it fetches the data the POS expects to receive and puts them inside a reply command.
7. The reply command is sent back, as a JSON file, over the socket to the POS.
8. The POS receives the JSON file and analyses it to know whether the operation was successful or not, eventually identify if data passed to be set have been set or not, eventually receives data from the reply from the POI, it they were present inside the reply sent by the POI.

2. Interfaces

These interfaces are all JSON files (or structures) used by the Listener application.

2.1. Listener settings

Data	*Type*	*Description*
Port	Integer	The port the Listener will open on the computer to receive commands from a POS

Data	Type	Description
AllowedIP	List of strings	A list TCP/IP addresses the Listener will accept commands from. If the Listener is to be used from multiple POS all computers IP (v4) trying to reach it will have to be defined here. The local address is defined by default and does not need to be added, that means the computer on which the Listener is installed can always send it commands without having to update the settings file.
AllowedServices	List of strings	A list of nexo Retailer services the Listener will accept to support and send a request to a POI. The POI does not support all nexo Retailer services, where the Listener does. That list allows to follow the evolutions of the POI to declare which services are supported or not. A service is declared using its name as indicated in [Sale to POI specifications – Retailer protocol]. The following services are although automatically defined and do not need to be defined (though they can): • Login • Logout • Payment (allowing a normal payment and a refund)

Example

```
{
  "Port": 29134,


  "AllowedIP": ["192.168.1.12", "165.76.99.21"],
  "AllowedServices": ["Reversal", "Reconciliation"]
}
```

2.2. Listener commands

2.2.1. Request

A request is sent by the POS to the Listener to request the activation of a nexo Retailer service.

Data	Type	Description
IP	String	The TCP/IP address of the POI to send the nexo service request to.
Port	Integer	The port to reach on the POI.
Service	String	The service to activate on the POS. The name of the service is as found in [Sale to POI specifications – Retailer protocol]. Example: “Login”, “Reconciliation”,...
SaleID	String	The name of the POS (sale) as given by the POS. That data is mandatory and will feed the MessageHeader.SaleID data in the nexo Retailer message. It is the name under which the POI knows the POS. Any value is valid but must remain consistent between a Login/Logout exchange. For more information refer to [Sale to POI specifications – Retailer protocol].

Data	*Type*	*Description*
POIID	String	The name of the POI (POIID) as given by the POS. That data is mandatory and will feed the MessageHeader.POIID data in the nexo Retailer message. It is the name under which the POS knows the POI. Any value is valid but must remain consistent between a Login/Logout exchange. For more information refer to [Sale to POI specifications – Retailer protocol].
ElementsToSend	Dictionary of Data Element	These are all the data elements the POS wishes to set inside the nexo Retailer service message. For a full description refer to §2.2.4.
ElementsToReturn	Dictionary of Data Element	These are all the data elements the POS wishes to set inside the nexo Retailer service message. For a full description refer to §2.2.4.
AutoLoginLogout	Boolean	Indicates whether a Login/Logout nexo service must be sent to the POI before and after the requested service (not done if the requested service is Login or Logout). This allows avoiding to send Login and Logout commands around a service.
PaymentType	String	***[Input only]*** Type of payment to perform (Normal will be considered by default). Set this data to “Refund” to perform a refund (if supported by the POI). For more information refer to [Sale to POI specifications – Retailer protocol].
RequestedAmount	Double	***[Payment/Refund service – Input only]*** Amount to use/authorise (pay or refund) for a payment or refund transaction, unused in all other cases.
AuthorizedAmount	Double	***[Payment/Refund service – Output only]*** Amount accepted/authorises for a payment or refund transaction, unused in all other cases.
POITransaction	Transaction ID	***[Payment service – Output only]*** This is the reference given by the POI to the payment transaction (successful or not). ***[Refund & Reversal[3] service – Input]*** The original ID of a payment transaction, retrieved from a previous Payment operation. ***[Refund & Reversal[3] service – Output]*** This is the reference given by the POI to the refund or reversal transaction (successful or not). Unused in all other cases.

Example for Login (note some data can be there but won’t be used):

```
{
  "IP": "172.21.48.1",
  "Port": 2018,
  "AutoLoginLogout": false,
  "Service": "Login",
  "SaleID": "SaleID",
```

3 *Reversal is the service name to use to process a cancellation.*

```
  "POIID": "POIID",
  "PaymentType": "Normal",
  "RequestedAmount": 0.0,
  "ElementsToSend": {
    "LoginRequest.SaleSoftware.ManufacturerID": {
      "Value": "MYSELF",
      "Status": false
    }
  },
  "ElementsToReturn": {
    "LoginResponse.POISystemData.POISoftware.ManufacturerID": {
      "Status": false
    }
  }
}
```

Example for Payment:

```
{
  "IP": "192.168.0.225",
  "Port": 2018,
  "AutoLoginLogout": true,
  "Service": "Payment",
  "SaleID": "SaleID",
  "POIID": "POIID",
  "PaymentType": "Normal",
  "RequestedAmount": 1.0,
  "AuthorizedAmount": 0.0,
  "ElementsToSend": {},
  "ElementsToReturn": {}
}
```

2.2.2. Transaction ID

A record describing a transaction ID generated by the POI.

Data	*Type*	*Description*
TransactionID	String	Transaction reference of a transaction.
TimeStamp	String	Timestamp the transaction was performed.

2.2.3. Reply

A reply is sent back by the Listener to the POS to inform it of the result of the previous request.

<table>
<tr><th>Data</th><th>Type</th><th>Description</th></tr>
<tr><td>ReplyStatus</td><td>Integer</td><td>A global status indicating how the request was processed.
<table>
<tr><th>Value</th><th>Description</th></tr>
<tr><td>0</td><td>The nexo service was successfully completed
(Response.Result = Success)</td></tr>
<tr><td>1</td><td>The nexo service failed to complete
(Response.Result = Failure)</td></tr>
<tr><td>2</td><td>The nexo service failed to complete
(Response.Result = Partial)</td></tr>
<tr><td colspan="2">All following error codes imply the nexo service wasn’t processed</td></tr>
<tr><td>3</td><td>No special error was detected.</td></tr>
<tr><td>4</td><td>Request service is not supported by the Listener</td></tr>
<tr><td>5</td><td>A mandatory object is not present</td></tr>
<tr><td>6</td><td>The request is invalid and can’t be processed</td></tr>
<tr><td>7</td><td>The service is invalid (it doesn’t exist)</td></tr>
<tr><td>8</td><td>Invalid POIID</td></tr>
<tr><td>9</td><td>Invalid SaleID</td></tr>
<tr><td>10</td><td>A value is invalid</td></tr>
<tr><td>11</td><td>Failed to connect to the POI; check the IP address and port number</td></tr>
<tr><td>12</td><td>Error during the exchange; check log file</td></tr>
<tr><td>13</td><td>Timeout during the exchange; the reply has not been received in time</td></tr>
<tr><td>14</td><td>Cancelled; the operation was cancelled by the user</td></tr>
<tr><td>15</td><td>Unknown error</td></tr>
</table></td></tr>
<tr><td>Label</td><td>String</td><td>The status in a text format</td></tr>
<tr><td>Message</td><td>String</td><td>A message describing the final result</td></tr>
<tr><td>NexoError</td><td>String</td><td>The contain of the Reponse.ErrorCondition data element inside the nexo Retailer service reply.
For more information refer to [Sale to POI specifications – Retailer protocol].</td></tr>
<tr><td>NexoInformation</td><td>String</td><td>The contain of the Reponse.AdditionalResponse data element inside the nexo Retailer service reply.
For more information refer to [Sale to POI specifications – Retailer protocol].</td></tr>
<tr><td>Request</td><td>The Request command</td><td>This is the request command as received by the Listener augmented with information about ElementsToSend and ElementsToReturn processing.
For a full description refer to §2.2.4.</td></tr>
</table>

Example for Login (note some data can be there but won’t be used):

```
{
  "Status": 13,
  "Label": "Success",
```

```
  "Message": "SUCCESS processing Login service",
  "NexoError": "2147483647",
  "Request": {
    "IP": "192.168.0.111",
    "Port": 2018,
    "Service": "Login",
    "SaleID": "SaleID",
    "POIID": "POIID",
    "ElementsToSend": {
      "LoginRequest.SaleSoftware.ManufacturerID": {
        "Value": "MYSELF",
        "Status": true
      }
    },
    "ElementsToReturn": {
      "LoginResponse.POISystemData.POISoftware.ManufacturerID": {
        "Value": "PMS.NEXO30",
        "Status": true
      }
    }
  }
}
```


Example for Payment:


```
{
  "Status": 1,
  "Label": "Failure",
  "Message": "FAILURE processing Payment service",
  "NexoError": "Refusal",
  "Request": {
    "IP": "192.168.0.225",
    "Port": 2018,
    "AutoLoginLogout": true,
    "Service": "Payment",
    "SaleID": "SaleID",
    "POIID": "POIID",
```

```
    "PaymentType": "Normal",
    "RequestedAmount": 1.0,
    "AuthorizedAmount": 0.0,
    "POITransaction": {
      "TransactionID": "00000063",
      "TimeStamp": "2021-11-10T09:44:56+01:00"
    },
    "ElementsToSend": {},
    "ElementsToReturn": {}
  }
}
```

2.2.4. Data Element

A data element is a data that will be exchanged between the POS and the Listener.

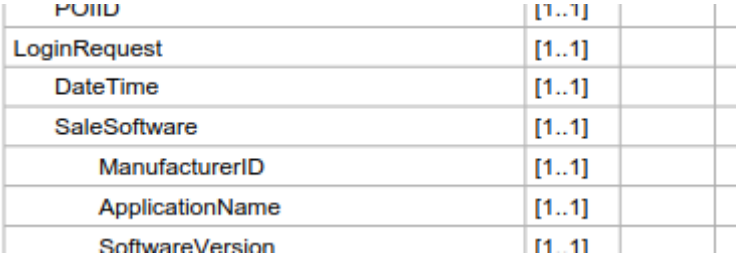
Each data element is composed of:

- **A key**

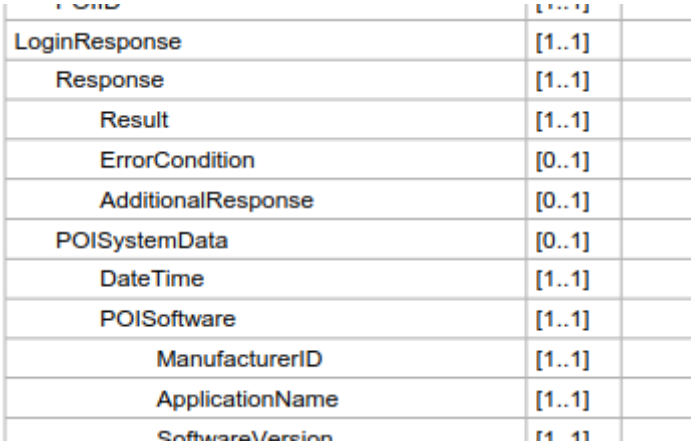
  The key is a string describing the path to the data to either set in the request sent to the POI or retrieve from the reply received from the POI.

  The path must declare every data by its name as declared inside the [Sale to POI specifications – Retailer protocol] and through which to go inside a request or reply to reach a data to set or retrieve. All intermediary data must be declared separated by a “.” (dot).

  For instance, to set the manufacturer ID inside a login request the key shall be:



  Key = **`LoginRequest.SaleSoftware.ManufacturerID`**



Key = **`LoginResponse.POISystemData.POISoftware.ManufacturerID`**

If the path does not point to a valid data, misses a node or adds an extra node, that data won't be reached and won't be set or retrieved.

The key is case insensitive.

Note: today the key doesn't allow accessing an array item.

- **A Data Element properties object**

  That object describes the value to set or retrieved and information about the success of the operation, as follows.

Data	*Type*	*Description*
Value	Object (string, numeric, boolean,...)	It can be any scalar data type (integer, boolean, numeric,…) or a string. **In ElementsToSend inside a Request** **Meaningful**, this is the value to set to the pointed data. It can be null. **In ElementsToSend inside a Reply** This is the value as specified in the Request command. **In ElementsToReturn inside a Request** Meaningless, it may be missing or set to any value which won't be used. **In ElementsToReturn inside a Reply** **Meaningful**, this is the value as found inside the reply from the POI ***only if Status is true***.
Status	Boolean	Indicates whether the data has been set or retrieved. That data is always meaningless in a request as it describes the successful result of setting or retrieving the data inside the service messages. **In ElementsToSend and ElementsToReturn inside a Request** Meaningless, it may be missing or set to any value which won't be used. **In ElementsToSend and ElementsToReturn inside a Reply** **Meaningful**, it indicates whether the data was set or retrieved successfully (true) or not (false). Any data, inside a reply message, with a false indicator is therefore meaningless to the POS.
Message	String	Tries to give information about the error is the data has not been set or retrieved. That data is always meaningless in a request as it describes and is meaning full only if the Status is false.

Example

```
"ElementsToSend": {
  "LoginRequest.SaleSoftware.ManufacturerID": {
    "Value": "MYSELF",
    "Status": false
  },
  "LoginRequest.SaleSoftware2.ManufacturerID": {
    "Value": "MYSELF",
```

```
      "Status": false
    }
  },
  "ElementsToReturn": {
    "LoginResponse.POISystemData.POISoftware.ManufacturerID": {
      "Status": false
    },
    "LoginResponse.POISystemData.POISoftware2.ManufacturerID": {
      "Status": false
    }
  }
```

Elements in green are valid (refer to [Sale to POI specifications – Retailer protocol]) while elements in red are false (the path is invalid). They will give the following response inside the Reply message:

```
    "ElementsToSend": {
      "LoginRequest.SaleSoftware.ManufacturerID": {
        "Value": "MYSELF",
        "Status": true
      },
      "LoginRequest.SaleSoftware2.ManufacturerID": {
        "Value": "MYSELF",
        "Status": false,
        "Message": "Failed settings data
LoginRequest.SaleSoftware2.ManufacturerID with MYSELF"
      }
    },
    "ElementsToReturn": {
      "LoginResponse.POISystemData.POISoftware.ManufacturerID": {
        "Value": "PMS.NEXO30",
        "Status": true
      },
      "LoginResponse.POISystemData.POISoftware2.ManufacturerID": {
        "Status": false,
        "Message": "Failed fetching data
LoginResponse.POISystemData.POISoftware2.ManufacturerID"
      }
    }
```

2.2.5. Exchanges with the Listener

Exchanging with the Listener is made using a socket, opened on start (as defined in Listener settings - §2.1).

The Listener exchanges with the outside world using Request and Reply json files, sent over the socket with the following format:

$O_1O_2O_3O_4$<JSON file>

With: $O_1O_2O_3O_4$ is a 4 bytes header giving the number of bytes inside the <JSON file>.

Coding $O_1O_2O_3O_4$ is done as follows:

- O_1 is the most significant byte, O_4 is the least significant byte
- The value is computed $(O_1)^4+(O_2)^3+(O_3)^2+(O_4)^1$
- The maximum value is $255^4+255^3+255^2+255^1 = 4294967295$