

PMS.NEXOSALExx Quick User Guide

by philippemp31@outlook.com (04/02/2022)

Table des matières

1. What is PMS.NEXOSALE.....	1
2. Prepare settings.....	2
2.1. Minimum settings.....	2
2.2. Advanced settings.....	3
3. Make a purchase.....	5
3.1. Stop processing.....	6
4. Any other service.....	8
5. Implement PMS.NEXOSALE.....	8
5.1. In a .NET project.....	8
5.1.1. VB.....	8
5.1.2. C#.....	9
5.2. Using a non .NET language.....	11
5.2.1. Delphi.....	11

1. What is PMS.NEXOSALE

PMS.NEXOSALExx is a software based on PMS.NEXOxx libraries.

PMS.NEXOxx libraries provide an implementation of nexo Retailer embedding all messages and the ability to use them. PMS.NEXOxx currently exists in 2 versions PMS.NEXO30 (nexo Retailer v3.0) and PMS.NEXO31 (nexo Retailer v3.1). PMS.NEXOSALExx also exists based on these libraries as PMS.NEXOSALE30 for nexo Retailer v3.0 and PMS.NEXOSALE31 for nexo Retailer v3.1. Before using either PMS.NEXOxx or PMS.NEXOSALExx libraries, refer to your POI supported nexo version.

PMS.NEXOSALExx apart from embedding PMS.NEXOxx and all its functionalities, also provides a complete and easy to use nexo Retailer implementation directly inside a sale system. Integration includes:

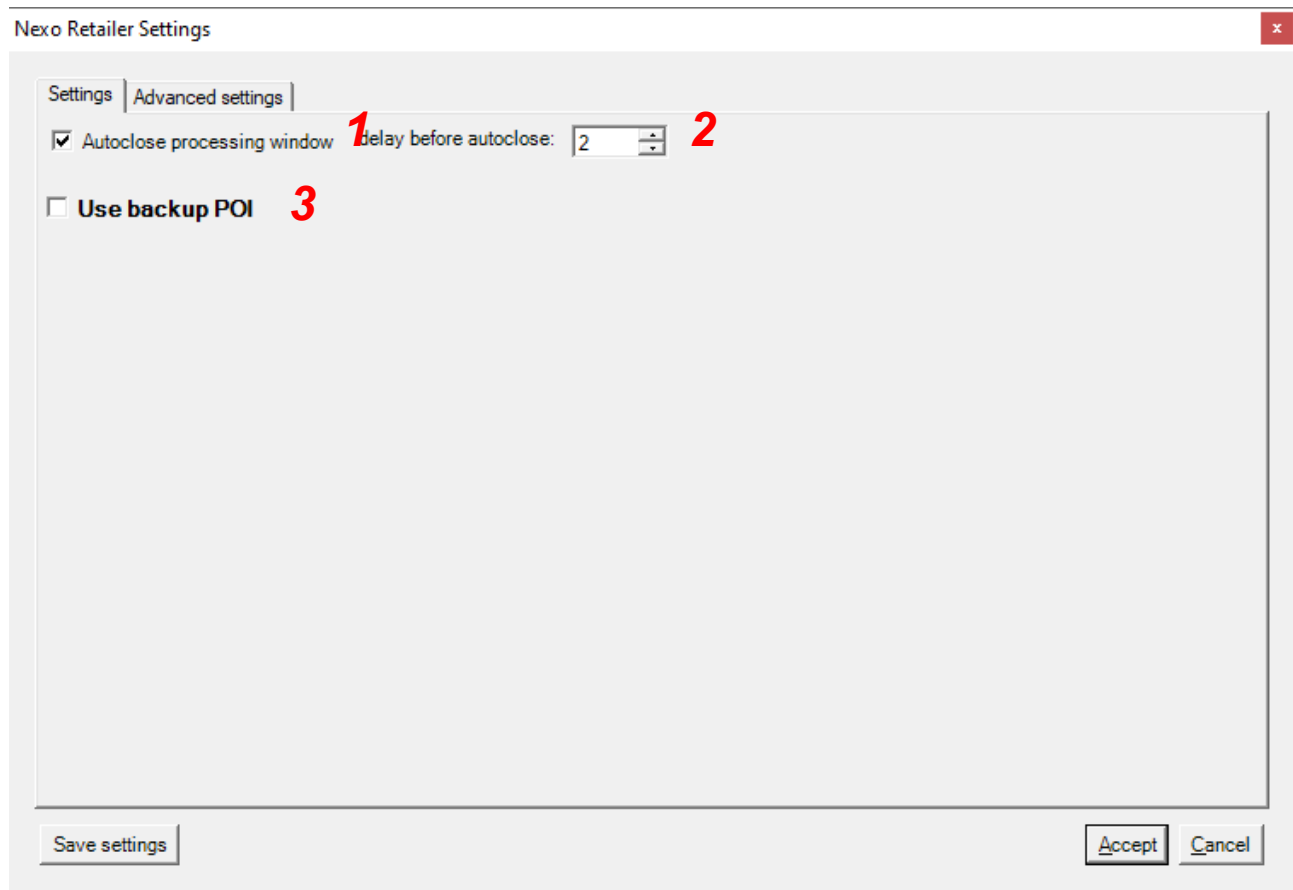
- Full settings management including:
 - Specify 2 different POI (primary and backup) with the ability to switch from one to the other
 - Specify all mandatory data to log to a POI
 - Activity tracing
- Full nexo Retailer message processing including:
 - Login & Logout
 - Payment, refund and reversal
 - Reconciliation,...

- Exchanges with the POI and simple result management

All these functionalities are available directly within dialog windows which will take care of all user interactions, as described here after.

2. Prepare settings

2.1. Minimum settings



PMS.NEXOSALExx Minimum settings window

1. Auto close processing window

This flag allows specifying whether the transaction window (showing transaction progress) must be dismissed automatically when the transaction is finished, or not.

Checking this flag will make the window disappear automatically, unchecking it will force the merchant to manually and explicitly close the window when the transaction is finished.

2. Delay before auto close

If the Terminal port flag has been set (checked), this value indicates the delay (in seconds) between the end of the transaction and the moment the window is automatically closed.

A low value might make the window disappear too fast, not giving the opportunity to the merchant to really check the transaction result.

A high value makes the window to stay longer on the screen preventing the receipt to be printed.

The advised value is between 1 and 5 seconds (2 preferable).

3. Use backup POI

Indicates whether the Y2 system must use the backup POI instead of the nominal one.

This is useful if the nominal POI is not reachable.

Terminal connection is available in the.

2.2. Advanced settings

The advanced setup are available if requested when displaying the settings and should be . It is strongly advised to not provide advanced setup to regular users.

NEXOSale Settings [nexo retailer v3.0]

Settings **Advanced settings** Miscellaneous

SaleID: ☒ Use local IP POIID: ☒ Use terminal IP Currency: 3

Application name: 4 Manufacturer name: 5 Software version: 6

Certification code: Admin code: ☐ Use date

☒ Primary POI 9 ☐ Backup POI ☐ Save receipts

Terminal IP: 10 Terminal port: 11 12 General nexo timer: 13

☒ Print receipt ☒ Customer ☒ Merchant 14 ☐ Synchronous Payment timer:

☐ Supports refund 15 ☐ Supports reconciliation 16 ☐ Supports cancel 17 ☒ Supports auto 18 ☐ Supports checks 19 Check operations timer:

Settings file name: 20

Log file name: 21

PMS.NEXOSALExx Advanced settings window

1. SaleID

Enter any ID identifying the sale.

To easily identify the sale it is possible to let the system use the IP address making it unique on the network.

2. POIID

Enter any ID identifying the POI (the terminal).

To easily identify the terminal it is possible to let the system use the IP address making it unique on the network.

3. Currency

Choose the currency that will be used to perform the transactions.

4. Application name

Any string identifying the application on the Y2 system.

5. Manufacturer name

Any string identifying the application manufacturer on the Y2 system.

6. Software version

Any string identifying the software version on the Y2 system.

7. Certification code

Any string accepted by the terminal.

8. Admin code

In case the terminal does not respond to a teller request, a code is requested to unlock the system. That code can be entered here. If no code is provided a date will be requested for that operation.

9. Primary POI / Backup POI

Allows indicating which POI settings to display and manipulate.

10. Terminal IP

The IP of the terminal to reach to perform a transaction (depends on Primary POI / Backup POI).

11. Terminal port

This IP port to reach on the terminal (depends on Primary POI / Backup POI).

12. Test connection

Allows to test the connection to the specified POI using the indicated IP + port data.

13. General nexo timer

The timer to wait for when a non blocking operation is in progress (login, logout,...).

The advised value is between 15 seconds.

14. Print receipt

Allows indicating the Y2 system must print the transaction receipt when finished.

If checked it is possible to indicate which receipts are to be printed (merchant and/or customer).

15. Supports refund

Indicates if the terminal accepts to perform refunds.

16. Supports reconciliation

Indicates if the terminal accepts to perform reconciliation.

17. Supports abort

Indicates if the terminal accepts to perform abort.

18. Supports cancel

Indicates if the terminal accepts to perform cancellation (reversal).

19. Supports checks

Indicates if the terminal accepts to perform checks operations.

20. Settings file name

Allows indicating where the settings file is stored. It can be stored locally or on a network.

21. Log file name

Allows indicating where the log file is stored. It can be stored locally or on a network.

3. Make a purchase

1. The merchant enters or compute the amount to pay for and calls the PMS.NEXOSALExx payment procedure (the procedure for calling this functionality might vary on cash registers).
2. The amount is sent to the POI for processing.
3. Once the transaction has been initiated onto the POI the purchase processing window is displayed.

PAYMENT REQUEST FROM SALE SALE TO POI - 4 seconds

13,04 EUR

a

PAYMENT REQUEST [487]

<?xml version="1.0" encoding="utf-8"?><SaleToPOIRequest><MessageHeader MessageClass="Service" MessageCategory="Payment" MessageID="0010193F41" SaleID="127.0.0.1" POIID="127.0.0.1" /><PaymentRequest><SaleData><SaleTransactionID TransactionID="2021-01-04T04:01:30+01:00" TimeStamp="2021-01-04T04:01:30+01:00" /></SaleData><PaymentTransaction><AmountsReq Currency="EUR" RequestedAmount="13.04" /></PaymentTransaction><PaymentData /></PaymentRequest></SaleToPOIRequest>

b

Payment in progress
Please wait...

c

Cancel

Purchase processing window: purchase in progress

3.a. AMOUNT

This upper part displays the amount of the transaction along with the transaction currency.

3.b. INFORMATION

The middle part displays information about how the transaction is being performed.

3.c. MESSAGE/RESULT

The lowest part displays messages to the merchant and the result of the transaction.

4. When the transaction is completed (accepted or not) the RESULT part indicates it with a colour code and a message indicating the final result.

PAYMENT REQUEST FROM SALE SALE TO POI - 27 seconds

13,04 EUR

PAYMENT RESPONSE [524]

```
<?xml version="1.0" encoding="utf-8"?><SaleToPOIResponse><MessageHeader MessageClass="Service" MessageCategory="Payment"
Message Type="Response" ServiceID="0010193F41" SaleID="127.0.0.1" POIID="127.0.0.1" /><PaymentResponse><Response Result="Success"
/><POIData><POITransactionID TransactionID="001019406E" TimeStamp="2021-01-04T04:01:30+01:00"
/></POIData><PaymentResult><PaymentInstrumentData><CardData /></PaymentInstrumentData><AmountsResp AuthorizedAmount="13.04"
/></PaymentResult></PaymentResponse></SaleToPOIResponse>
```

Payment accepted

Close

This transaction has been accepted (RESULT is GREEN)

PAYMENT REQUEST FROM SALE SALE TO POI - 5 seconds

13,04 EUR

PAYMENT RESPONSE [1335]

```
<?xml version="1.0" encoding="utf-8"?><SaleToPOIResponse><MessageHeader MessageClass="Service" MessageCategory="Payment"
Message Type="Response" ServiceID="001030C7E8" SaleID="127.0.0.1" POIID="127.0.0.1" /><PaymentResponse><Response Result="Failure"
ErrorCondition="Cancel" /><POIData><POITransactionID TransactionID="001030C924" TimeStamp="2021-01-04T04:27:12+01:00"
/></POIData><PaymentResult><PaymentInstrumentData><CardData /></PaymentInstrumentData><AmountsResp AuthorizedAmount="13.04"
/></PaymentResult><PaymentReceipt DocumentQualifier="CustomerReceipt"><OutputContent OutputFormat="Text"><Output Text
EndOfLineFlag="false">TRANSACTION DECLINED</Output Text><Output Text EndOfLineFlag="false"></Output Text><Output Text
EndOfLineFlag="false">CLIENT RECEIPT</Output Text><Output Text EndOfLineFlag="false">AMOUNT: 13,04 EUR</Output Text><Output Text
```

Payment declined

Close

This transaction has been declined (RESULT is RED)

Please note that cancelling the transaction using the “**Cancel**” button will also decline the transaction (RESULT is RED) with a different message indicating the transaction was cancelled.

3.1.Stop processing

In case the terminal does not answer the sale system is blocked. In that case it is possible to unlock it by doing as follow:

1. Double-click on "AMOUNT" part.
2. Double-click on "INFORMATION" part.
3. Double-click on "MESSAGE/RESULT" part.

That will display a window which will allow cancelling the current processing.

Confirm cancel operation

Are you sur you want to cancel the operation ?

Enter today's date (YYYYMMDD):

Window to cancel processing in progress

The operator must then enter either (i) the Admin code If defined or (ii) the current date in the indicated format (YYYYMMDD), then validate (entering the date has no effect if an admin code has been defined).

4. If a valid admin code or date has been entered the transaction is aborted and the sale system is released.

PAYMENT REQUEST FROM SALE SALE TO POI - 5 seconds

13,04 EUR

PAYMENT RESPONSE [1335]

```
<?xml version="1.0" encoding="utf-8"?><SaleToPOIResponse><MessageHeader MessageClass="Service" MessageCategory="Payment"
Message Type="Response" ServiceID="001030C7E8" SaleID="127.0.0.1" POIID="127.0.0.1" /><PaymentResponse><Response Result="Failure"
ErrorCondition="Cancel" /><POIData><POITransactionID TransactionID="001030C924" TimeStamp="2021-01-04T04:27:12+01:00"
/></POIData><PaymentResult><PaymentInstrumentData><CardData /></PaymentInstrumentData><AmountsResp AuthorizedAmount="13.04"
/></PaymentResult><PaymentReceipt DocumentQualifier="CustomerReceipt"><OutputContent OutputFormat="Text"><Output Text
EndOfLineFlag="false">TRANSACTION DECLINED</Output Text><Output Text EndOfLineFlag="false"></Output Text><Output Text
EndOfLineFlag="false">CLIENT RECEIPT</Output Text><Output Text EndOfLineFlag="false">AMOUNT: 13,04 EUR</Output Text><Output Text
```

Payment declined

Close

Beware, aborting the transaction that way does not abort the transaction on the POI. The operator must use that option only in specific cases (terminal does not answer, no more network,...).

4. Any other service

Windows will be as for a purchase but according to the requested service (Refund, Abort,...).

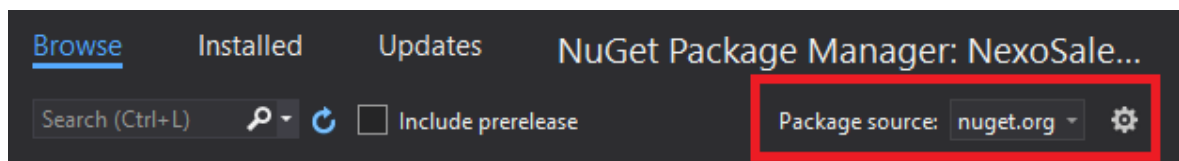
5. Implement PMS.NEXOSALE

5.1. In a .NET project

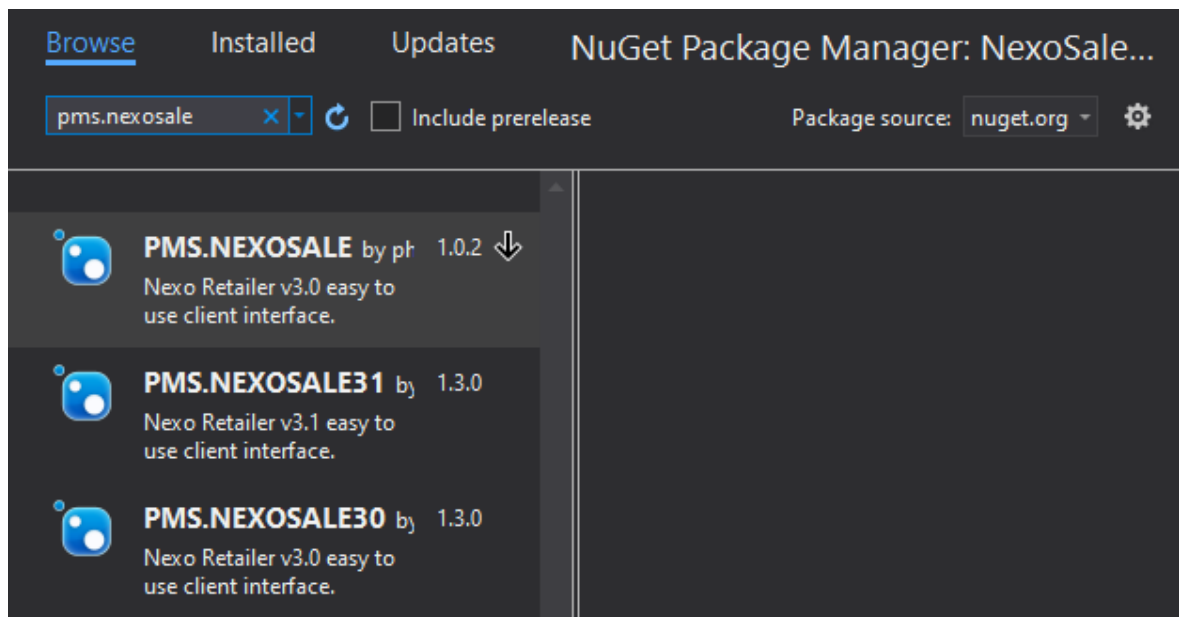
PMS.NEXOSALExx is available for downloading in <https://www.nuget.org>.

To import it inside your own .NET project, inside Visual Studio:

1. Right click on your project and select "Manage nuget packages..." and make sure to use "nuget.org" as a source.



2. In the Browse pane, enter "pms.nexosale" in the search widget



That will display all PMS.NEXOSALExx packages available.

Forget PMS.NEXOSALE which has been deprecated and select the one you want to use (PMS.NEXOSALE30 for nexo Retailer v3.0, PMS.NEXOSALE31 for nexo Retailer v3.1).

5.1.1. VB

- Create the object

```
Imports NEXOSALE
Private Nxo As New NEXOSALE.NEXOSALE
```

- Display the settings window

```
Nxo.DisplaySettings(trueOrfalse)
```


with trueOrfalse is:

- true: display Advanced settings
 - false: Advanced settings are not made available
- Display the processing window

```
Public Enum Action
```

```
_none  
_begin  
Login  
Logout  
Payment  
_base  
Refund  
Reversal  
Reconciliation  
Abort  
_checks  
ReadCheck  
PrintCheck
```

```
_end
```

```
End Enum
```

```
Dim operation as Action = Action.Payment
```

```
Dim result As ActionResult = Nxo.DisplayProcessing(operation)
```

```
lblResult.Text = result.ToString
```

```
lblBrand.Text = Nxo.Brand
```

```
Select Case result
```

```
Case ActionResult.success
```

```
    Select Case ComboBox1.SelectedItem
```

```
        Case Action.Payment
```

```
            Nxo.OriginalPOITransactionID = Nxo.POITransactionID
```

```
            Nxo.OriginalPOITransactionTimestamp = Nxo.POITransactionTimestamp
```

```
        Case Action.Refund
```

```
            Nxo.OriginalPOITransactionID = "AAA" 'Nothing
```

```
            Nxo.OriginalPOITransactionTimestamp = Nothing
```

```
        Case Action.Reversal
```

```
            Nxo.OriginalPOITransactionID = Nothing
```

```
            Nxo.OriginalPOITransactionTimestamp = Nothing
```

```
        Case Action.Reconciliation
```

```
    End Select
```

```
End Select
```

5.1.2. C#

- Create the object

```
using NEXOSALE;
```

```
NEXOSALE Nxo = new NEXOSALE();
```

- Display the settings window

```
Nxo.DisplaySettings(trueOrfalse);
```

with trueOrfalse is:

- true: display Advanced settings
 - false: Advanced settings are not made available
- Display the processing window

```

Action operation = Action.Payment;
ActionResult result = Nxo.DisplayProcessing(operation);
lblResult.Text = result.ToString();
lblBrand.Text = Nxo.Brand;
switch (result)
{
    case ActionResult.success:
        switch (operation)
        {
            case Action.Payment:
                Nxo.OriginalPOITransactionID = Nxo.POITransactionID;
                Nxo.OriginalPOITransactionTimestamp = Nxo.POITransactionTimestamp;
                break;
            case Action.Refund:
                Nxo.OriginalPOITransactionID = "AAA";
                Nxo.OriginalPOITransactionTimestamp = null;
                break;
            case Action.Reversal:
                Nxo.OriginalPOITransactionID = null;
                Nxo.OriginalPOITransactionTimestamp = null;
                break;
            case Action.Reconciliation:
                break;
        }
        break;
}
}

```

5.2. Using a non .NET language

The component is available as a COM object for non .NET languages (Delphi,...).

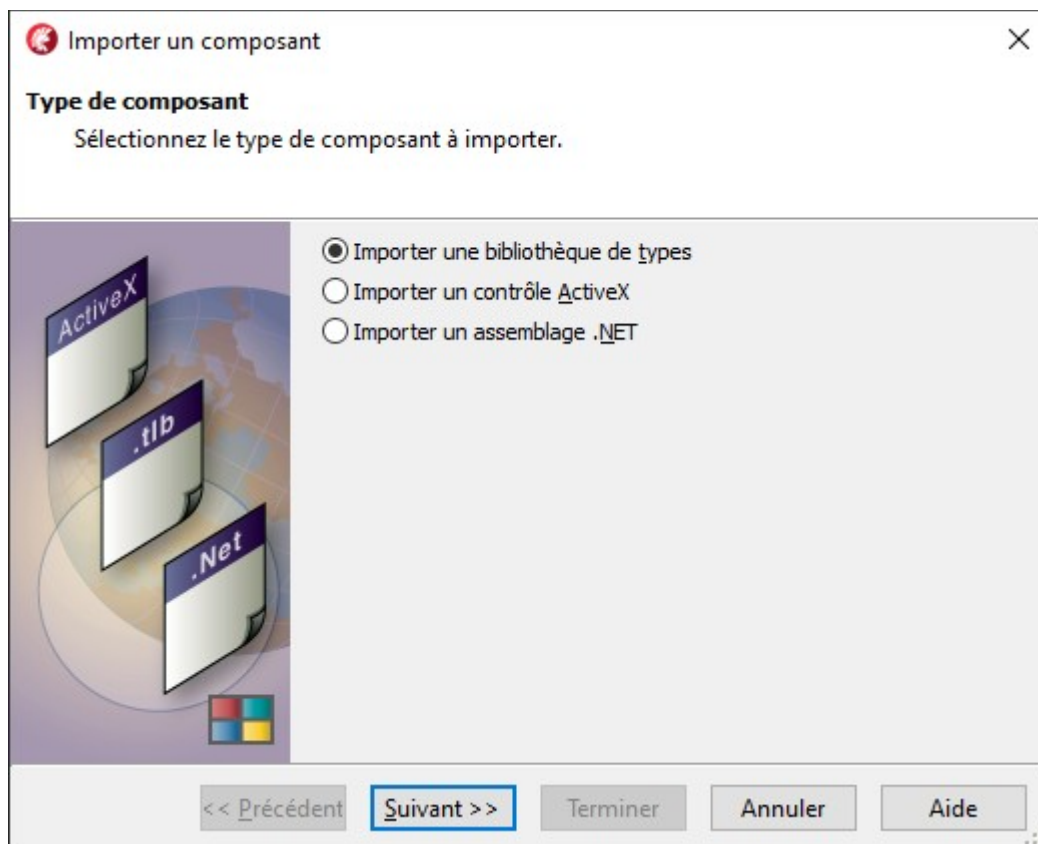
To use in Delphi the component must be registered on the computer (PMS.COMMON.dll, PMS.NEXOxx.dll & PMS.NEXOSALExx.dll) using the following command:

```
Regasm.exe <dll to register> /codebase
```

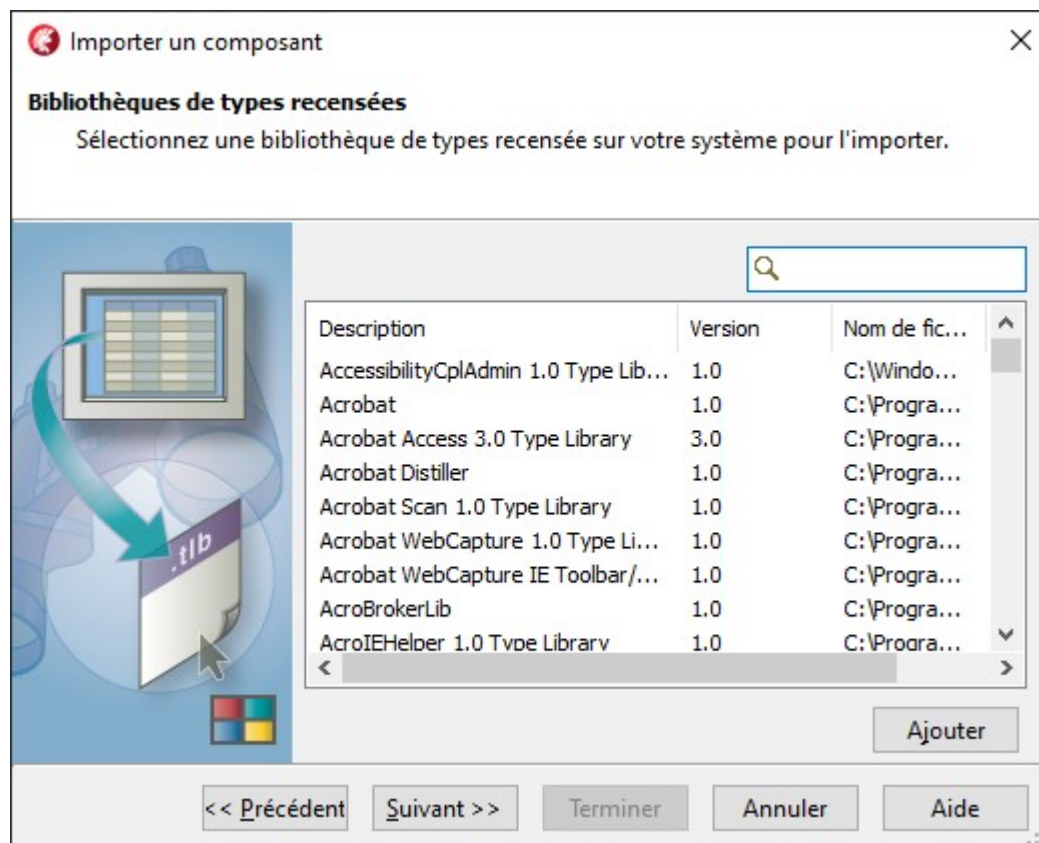
5.2.1. Delphi

To use the component inside Delphi:

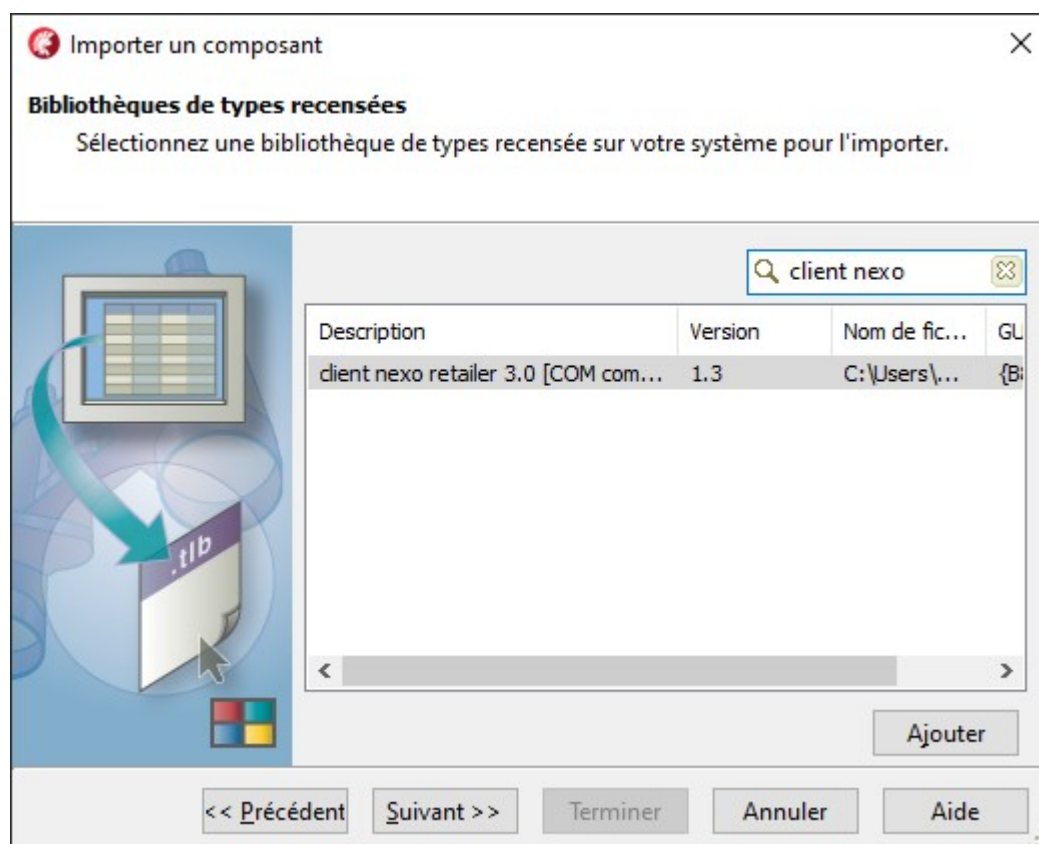
1. In the menu, select: “Component→Import component”, then choose “Import a type library” and click “Next”



2. Select the right type library to import:

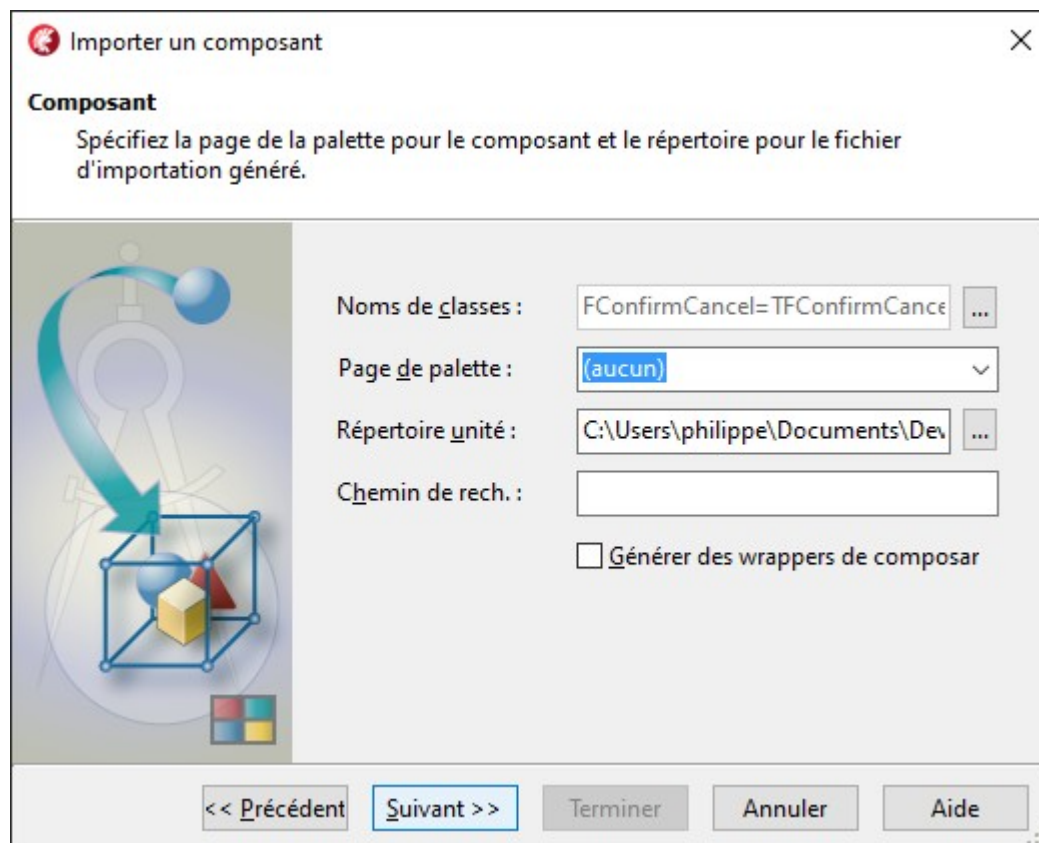


Type "client nexo retailer"



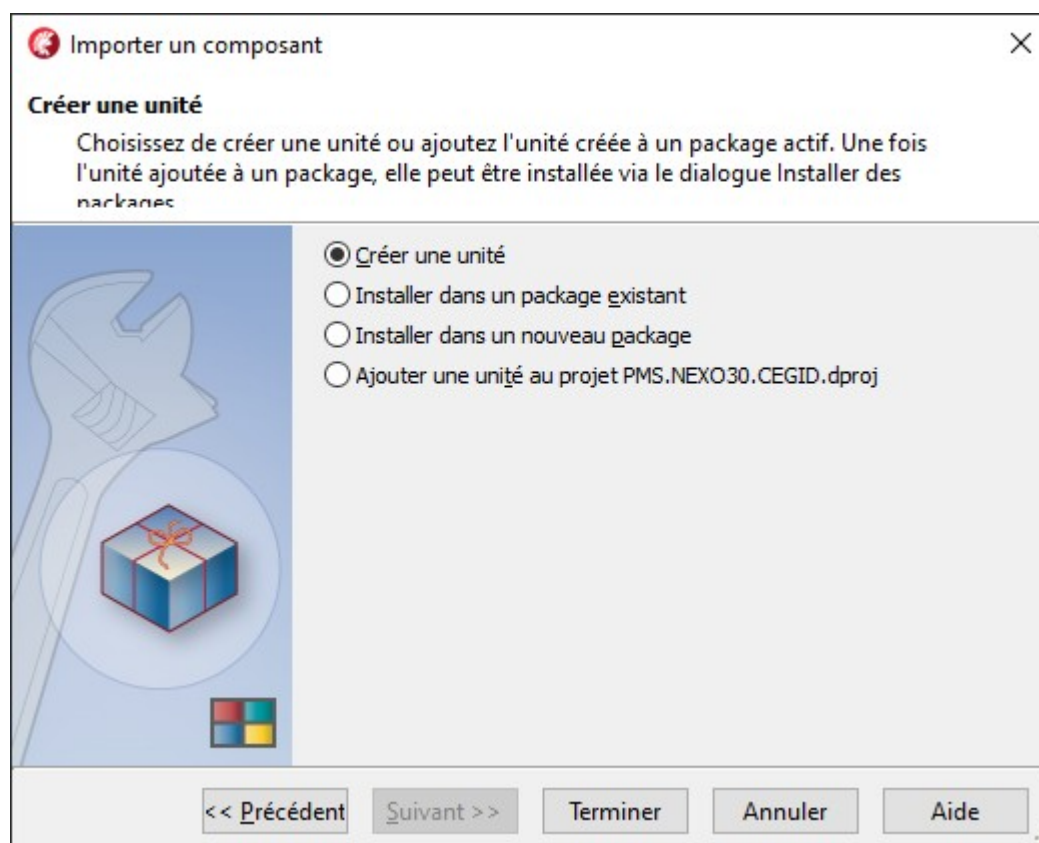
The nexo Retailer component is displayed, select it and click on "Next"

3. Select where to put the the component and click on “Next”



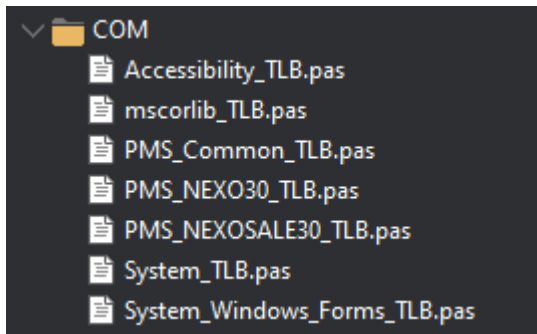
The dialog box is titled "Importer un composant" with a close button (X) in the top right corner. Below the title bar, the section "Composant" is displayed, followed by the instruction: "Spécifiez la page de la palette pour le composant et le répertoire pour le fichier d'importation généré." To the left of the input fields is a graphic showing a blue sphere with a curved arrow pointing to a 3D cube. The input fields are: "Noms de classes :" with the text "FConfirmCancel= TFCConfirmCance" and a browse button (...); "Page de palette :" with a dropdown menu showing "(aucun)"; "Répertoire unité :" with the path "C:\Users\philippe\Documents\Dev" and a browse button (...); and "Chemin de rech. :" with an empty text box. Below these fields is a checkbox labeled "Générer des wrappers de composar". At the bottom, there are five buttons: "<< Précédent", "Suivant >>", "Terminer", "Annuler", and "Aide".

4. Select “Create a unit” and click “Terminate”



The dialog box is titled "Importer un composant" with a close button (X) in the top right corner. Below the title bar, the section "Créer une unité" is displayed, followed by the instruction: "Choisissez de créer une unité ou ajoutez l'unité créée à un package actif. Une fois l'unité ajoutée à un package, elle peut être installée via le dialogue Installer des packages." To the left of the radio buttons is a graphic showing a wrench and a 3D cube with a red ribbon. The radio buttons are: "Créer une unité" (selected), "Installer dans un package existant", "Installer dans un nouveau package", and "Ajouter une unité au projet PMS.NEXO30.CEGID.dproj". At the bottom, there are five buttons: "<< Précédent", "Suivant >>", "Terminer", "Annuler", and "Aide".

5. That will create a set of units you must add to your project.



6. Now go somewhere in your app and put the following statements:

```
// create the PMS.NEXOSALE object
myNexo := CoNEXOSale.create;

// associate your main form to the object (to prevent window
// disappearing)
myNexo.MainWindow := Application.MainFormHandle;

...

// call settings window (fFullSettinigs = true to display Advanced
// settings)
myNexo.DisplaySettings(fFullSettings);

...

// make a purchase
// set amount as expected
myNexo.amount := 100; // always expressed in cents
// set some specific settings
myNexo.MerchantReferenceID := "MYREF";
myNexo.Login.RequestOperatorID := "MY NAME";

...

dlgpr := Action_payment;
dlgpres := myNexo.DisplayProcessing(dlgpr);

// test transaction result
case dlgpres of
  ActionResult_success:
    begin
      processedAmount := myNexo.Payment.ReplyAuthorizedAmount;
      txnBrand := myNexo.Brand;
      txnAuthNumber :=
myNexo.Payment.ReplyData.PaymentResult.PaymentAcquirerData.ApprovalCod
e;

      if myNexo.Payment.ReplyData.PaymentResult.OnlineFlag then
        txnMode := TXN_MODE_ONLINE
      else
        txnMode := TXN_MODE_OFFLINE;
```

```

        txnCard :=
myNexo.Payment.ReplyData.PaymentResult.PaymentInstrumentData.CardData.
MaskedPAN;
    end;
else
    begin
        processedAmount := myNexo.Payment.RequestRequestedAmount;
        case dlgpres of
            ActionResult_decline:
                begin
                    // payment has been declined
                    dlgpresText := TXN_RESULT_DECLINED;
                end;
            ActionResult_incomplete:
                begin
                    // payment is incomplete (has been stopped)
                    dlgpresText := TXN_RESULT_INCOMPLETE;
                end;
            ActionResult_cancel:
                begin
                    // cancelled by user while processing
                    dlgpresText := TXN_RESULT_CANCELLED_BY_USER;
                end;
            ActionResult_timeout:
                begin
                    // timeout waiting the answer
                    dlgpresText := TXN_RESULT_TIMEOUT;
                end;
            ActionResult_exception:
                begin
                    // an exception has occurred
                    dlgpresText := TXN_RESULT_EXCEPTION;
                end
            else
                begin
                    dlgpresText := TXN_RESULT_UNKNOWN_ERROR;
                end;
        end;
    end;
end;
end;

```

7. And you're all set !