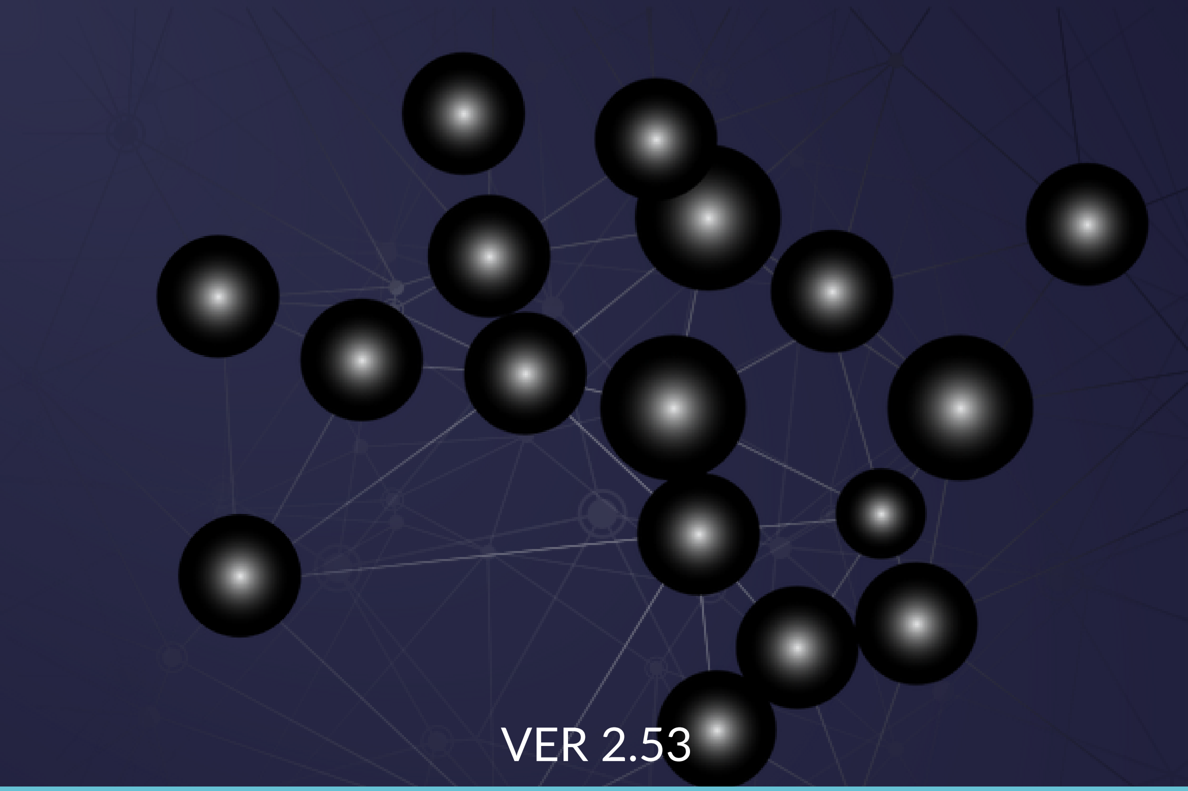
XE-CONNECT

Quick Start Guide for Windows



VER 2.53

ONTENT

Installation Instructions

1) Extract the XEConnect zip file into the root of your C: drive. Once you have extracted the folder, the file path should look like C:\XEConnect.

2) Locate xeposExternal.exe in this folder and ensure it is running in the background.

3)When xeposExternal.exe is running you should see an “XE” logo in the bottom right of your taskbar in the system tray, or in the overflow menu.

4) xeposExternal.exe should be set to run every time your software opens, or alternatively you can set it to run on Windows start-up.

XEPAY XE-CONNECT™

Adding **EMV** transaction processing to your POS system is easy with the pre-certified **XE-CONNECT™.**

The pre-certified **XE-CONNECT™** installs along side your software application to add **EMV** transaction processing to your POS system.

XE-CONNECT™ facilitates all transactional communication with the **EVO** Payments International global processing platforms and approved hardware devices to isolate payment data and keep it separate from the software application.

XE-CONNECT™ is designed to support multiple terminal manufacturers while retaining a common API. At startup, **XE-CONNECT™** detects the supported terminal manufacturer(s)/models for processing Authorize, Authorize & Capture, Return and Void transactions.

How It Works

1. Create transaction data objects in your POS software.

2. Pass the transaction data to **XE-CONNECT™**.

3. **XE-CONNECT™** initiates terminal commands and gathers tender/EMV data to send to the EVO Snap* Platform/ E-service.

4. The **EVO Snap* Platform / E-service** returns a response to **XE-CONNECT™** with receipt details.

- **XE-CONNECT™ (ver. 2.53)** works with both **EVO** Payments International global processing platforms **(snap* commerce driver (ver. 2.33.0.9) & E-Service)** which we will explain how to integrate both services into your POS software while using the same API's for both platforms.

Version Details

XE-CONNECT™ - ver 2.53

- Supported Terminals:

(Snap* commerce driver): **Ingenico ICMP via Serial USB, Ingenico iPP320/iPP350 via Serial USB**

(E-service): **Ingenico Desk 3500 & Ingenico Move 3500**

Compatibility

- XE-CONNECT™ – Windows® 7+
- Visual Studio 2010+
- .Net 4.8

Setup

(Snap* Commerce Driver / E-Service)

To get started with Snap * commerce driver & E-Service integration, **XE-CONNECT™** must be hosted locally.
For our example, the following folder has been placed in the C:/ directory.

1. Download the **XE-CONNECT™** Software and place it in the C:/ directory.

2. Extract the archive into the same directory.

3. Edit the XeposExternal.exe.config file.

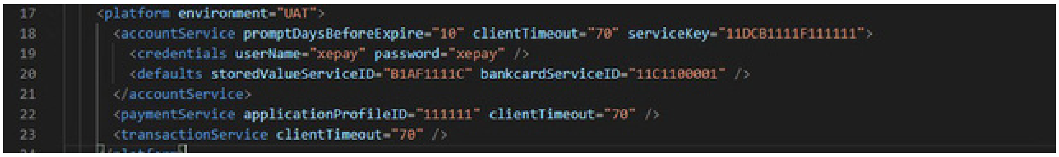
4. In the configuration file the values that need to be configured are between line 17 to 34 which are as follows

- All values are string type

CREDENTIALS

(Snap* commerce driver)

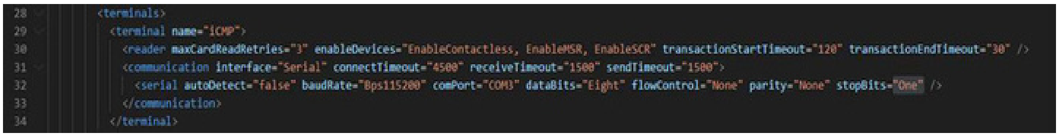
- **Platform environment:** There are 2 types of environments that can be set, UAT & PROD
- **UAT:** For test purposes (UAT card terminal required)
- **PROD:** For production purposes
- **Service Key:** provided by XEPAY (example: "11DCB1111F111111")
- **Username & Password:** Provided by XEPAY (example: username="xepay" password="xepay")
- **Store Value Service ID:** Provided by XEPAY (example: "B1AF1111C")
- **Bank Card Service ID (Terminal ID):** provided by XEPAY (example: "11C1100001")
- **Application profile ID:** Provided by XEPAY (example: "111111")



```
<platform environment="UAT">
  <accountService promptDaysBeforeExpire="10" clientTimeout="70" serviceKey="11DCB1111F111111">
    <credentials userName="xepay" password="xepay" />
    <defaults storedValueServiceID="B1AF1111C" bankcardServiceID="11C1100001" />
  </accountService>
  <paymentService applicationProfileID="111111" clientTimeout="70" />
  <transactionService clientTimeout="70" />
```

Terminal Configuration (Snap* commerce driver)

- **Terminal Name:** “icmp”
- **Serial Auto Detect:** “false”
- **Baud Rate:** "Bps115200"
- **Com Port:** device manager should be checked for Com port
- **Data Bits:** "Eight"
- **Flow Control:** "None"
- **Parity:** "None"
- **Stop Bits:** "One"



```
<terminals>
  <terminal name="iCMP">
    <reader maxCardReadRetries="3" enableDevices="EnableContactless, EnableMSR, EnableSCR" transactionStartTimeout="120" transactionEndTimeout="30" />
    <communication interface="Serial" connectTimeout="4500" receiveTimeout="1500" sendTimeout="1500">
      <serial autoDetect="false" baudRate="Bps115200" comPort="COM3" dataBits="Eight" flowControl="None" parity="None" stopBits="One" />
    </communication>
  </terminal>
```

- Ater setting up all the details listed above save the configuration and run xeposexternal.exe

Terminal Configuration

(E-service)

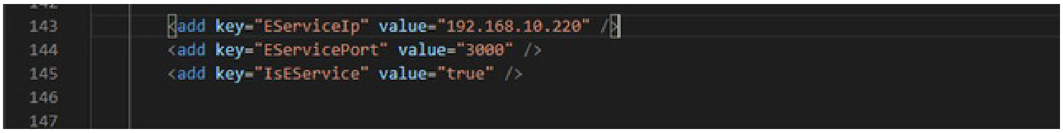
The Terminals that are integrated with the E-Service platform uses a local or public IP address and a static port number to communicate with **XE-CONNECT™.**

- EServiceIp: card terminal IP address
- EServicePort: default value is “3000”
- IsEService: the value is either “True” or “False”

Values:

True: XE-CONNECT™ will work in E-Service platform mode

False: XE-CONNECT™ will work in Snap *commerce drive mode



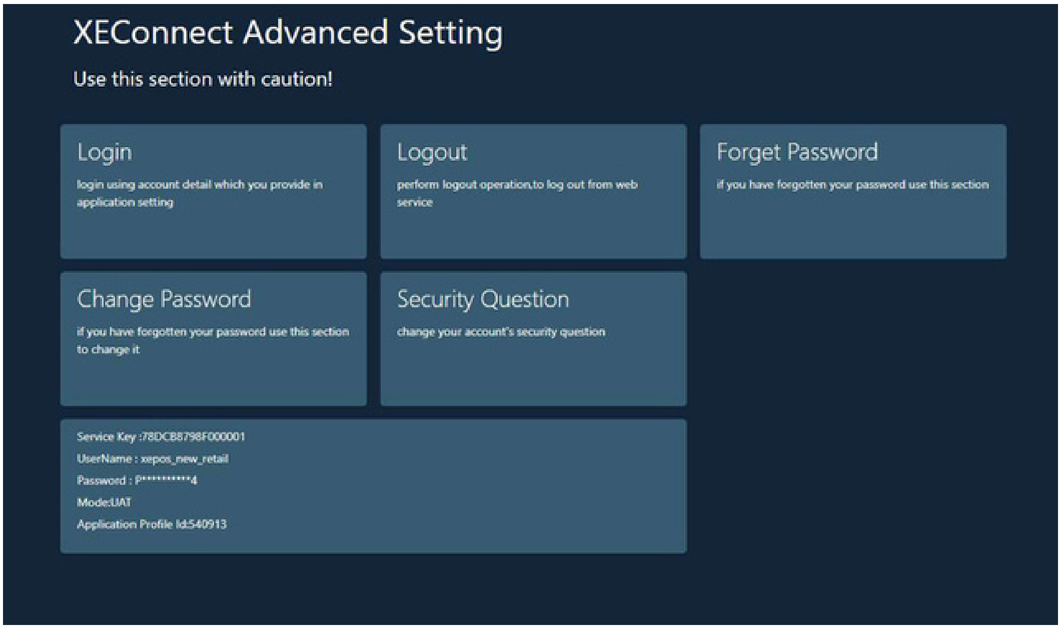
```
<add key="EServiceIp" value="192.168.10.220" />
<add key="EServicePort" value="3000" />
<add key="IsEService" value="true" />
```

Advanced Settings

(Snap* commerce driver)

1. For accessing the advanced settings to change your account information for the snap* commerce driver you could use the following link below http://localhost:5050/api/evoconfiguration

Please See Image Below



XEConnect Advanced Setting

Use this section with caution!

Login

login using account detail which you provide in application setting

Logout

perform logout operation,to log out from web service

Forget Password

if you have forgotten your password use this section

Change Password

if you have forgotten your password use this section to change it

Security Question

change your account's security question

Service Key :78DCB8798F000001
UserName : xepos_new_retail
Password : P**********4
Mode:UAT
Application Profile Id:540913

The API used for each property is as follows

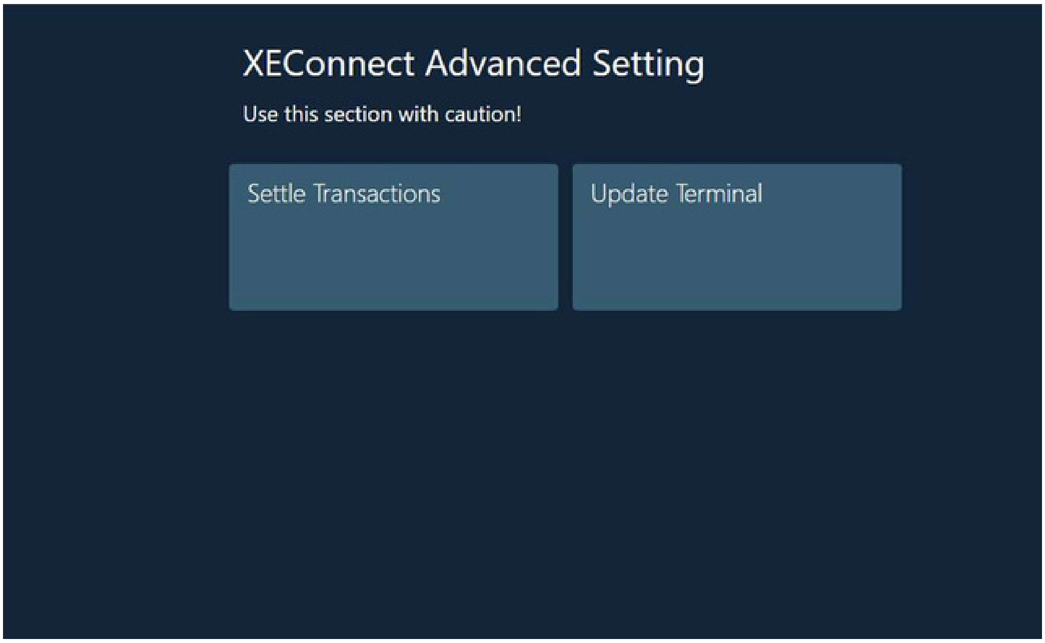
- **Login:** http://localhost:5050/api/Xepos/Evo/Login
- **Logout:** http://localhost:5050/api/Xepos/Evo/Logout\
- **Forget Password:** http://localhost:5050/api/Xepos/Evo/ForgotPassword
- **Change password:** http://localhost:5050/api/Xepos/Evo/ChangePassword
- **Security questions:** http://localhost:5050/api/Xepos/Evo/GetSecurityQuestions

Advanced settings

(E-service)

1. For accessing the advanced settings to update or settle the terminal you could use the following link below http://localhost:5050/api/eserviceConfiguration

Please See Image Below



XEConnect Advanced Setting

Use this section with caution!

The API used for each property is as follows

- **Settle Transactions:** http://localhost:5050/api/Xepos/Evo/DoReconciliation
- **Update Terminal:** http://localhost:5050/api/Xepos/Evo/UpdateTerminal

Integration & Transaction Processing

Two transaction sets can be processed using **XE-CONNECT™.**

Terminal Required Transactions

- Authorize (Snap* commerce driver)
- Authorize and Capture (Snap* commerce driver & E-Service)
- Return unlinked (E-service)

No Terminal Required Transactions

- Capture (Snap* commerce driver)
- Undo (Void) (Snap* commerce driver & E-Service)
- Return by ID (Refund) (Snap* commerce driver)

For processing a transaction & communicating with **XE-CONNECT™** there are 2 types of method used, GET & POST.

Authorize

(Snap* Commerce Driver)

The Authorize function is processed by a POST method & the API & values needed to proceed this transaction is as follows

Values:

```
{
  "Amount": 0,
  "OrderNo": "string",
  "TipAmount": 0,
  "CashbackAmount": 0,
  "EmployeeId": "string",
  "RefrenceNo": "string",
  "LaneId": "string"
}
```

Authorize and Capture

(Snap* Commerce Driver & E-Service)

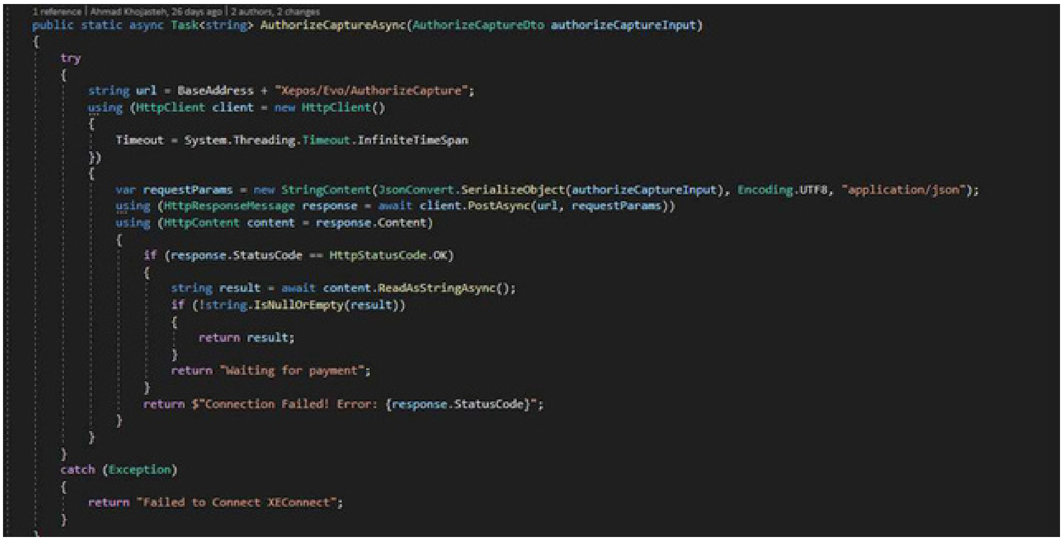
this function is processed by a POST method & the API & values needed to proceed the Authorize and capture transaction is as follows

API: http://localhost:5050/api/Xepos/Evo/AuthorizeCapture

Values:

```
{
  "Amount": 0,
  "OrderNo": "string",
  "TipAmount": 0,
  "CashbackAmount": 0,
  "EmployeeId": "string",
  "RefrenceNo": "string",
  "LaneId": "string"
}
```

C# Sample Code:



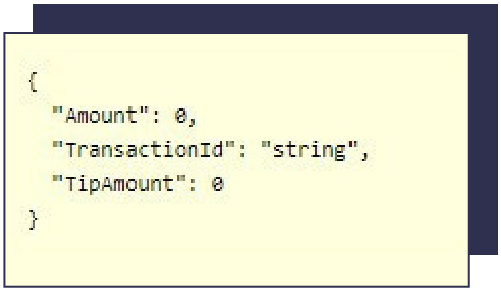
```
1 reference | Ahmad Khojasteh, 26 days ago | 2 authors, 2 changes
public static async Task<string> AuthorizeCaptureAsync(AuthorizeCaptureDto authorizeCaptureInput)
{
    try
    {
        string url = BaseAddress + "Xepos/Evo/AuthorizeCapture";
        using (HttpClient client = new HttpClient()
        {
            Timeout = System.Threading.Timeout.InfiniteTimeSpan
        })
        {
            var requestParams = new StringContent(JsonConvert.SerializeObject(authorizeCaptureInput), Encoding.UTF8, "application/json");
            using (HttpResponseMessage response = await client.PostAsync(url, requestParams))
            using (HttpContent content = response.Content)
            {
                if (response.StatusCode == HttpStatusCode.OK)
                {
                    string result = await content.ReadAsStringAsync();
                    if (!string.IsNullOrEmpty(result))
                    {
                        return result;
                    }
                    return "Waiting for payment";
                }
                return $"Connection Failed! Error: {response.StatusCode}";
            }
        }
    }
    catch (Exception)
    {
        return "Failed to Connect XEConnect";
    }
}
```

Capture

(Snap* Commerce Driver)

This function is processed by a POST method & the API & values needed to proceed the Authorize and capture transaction is as follows
API: http://localhost:5050/api/Xepos/Evo/Capture

Values:



```
{
  "Amount": 0,
  "TransactionId": "string",
  "TipAmount": 0
}
```

TranactionId: The value is provided from the response of the authorize transaction

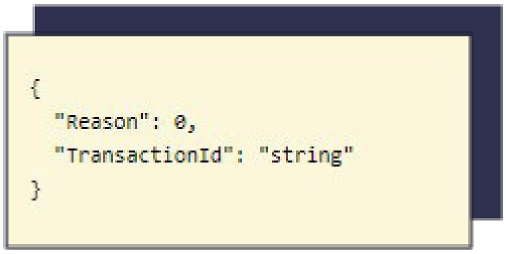
Undo (Void)

(Snap* Commerce Driver & E-Service)

The Undo function is processed by a POST method & the API & values needed to proceed the Authorize and capture transaction is as follows

API: http://localhost:5050/api/Xepos/Evo/Void

Values:

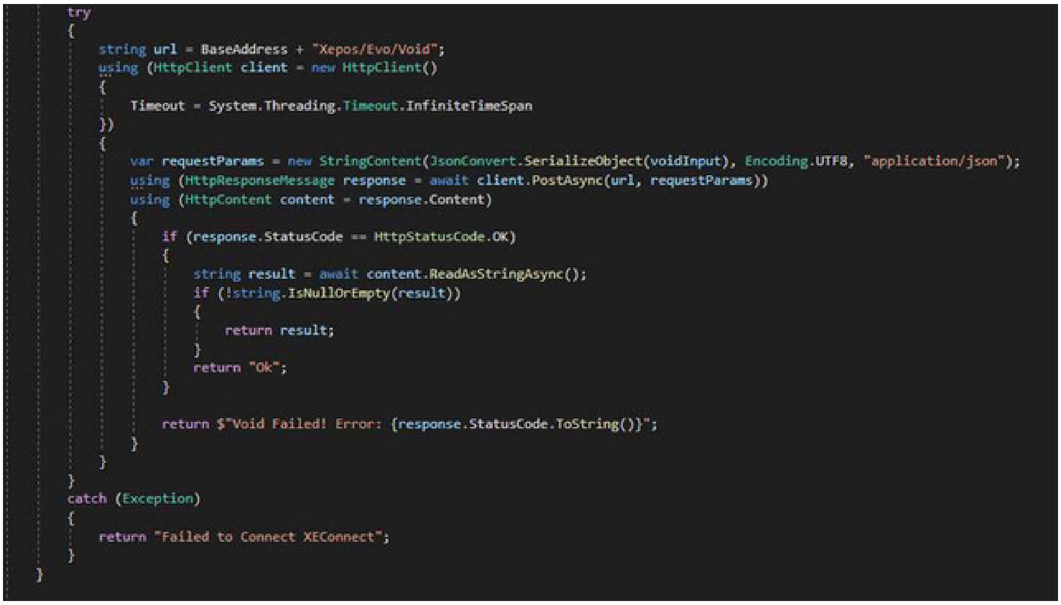


```
{
  "Reason": 0,
  "TransactionId": "string"
}
```

Reason: This value should always be set as 0

TransactionId: The value is provided from the response of the authorize and authorize & capture transactions

C# Sample Code:



```
    try
    {
        string url = BaseAddress + "Xepos/Evo/Void";
        using (HttpClient client = new HttpClient()
        {
            Timeout = System.Threading.Timeout.InfiniteTimeSpan
        })
        {
            var requestParams = new StringContent(JsonConvert.SerializeObject(voidInput), Encoding.UTF8, "application/json");
            using (HttpResponseMessage response = await client.PostAsync(url, requestParams))
            using (HttpContent content = response.Content)
            {
                if (response.StatusCode == HttpStatusCode.OK)
                {
                    string result = await content.ReadAsStringAsync();
                    if (!string.IsNullOrEmpty(result))
                    {
                        return result;
                    }
                    return "Ok";
                }

                return $"Void Failed! Error: {response.StatusCode.ToString()}";
            }
        }
    }
    catch (Exception)
    {
        return "Failed to Connect XEConnect";
    }
}
```

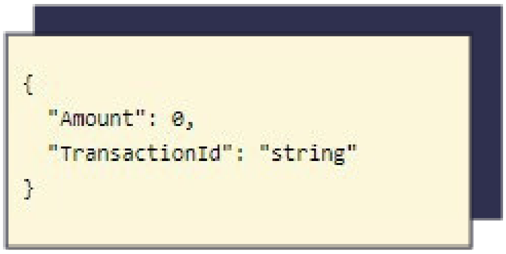
Return by I (Refund)

(Snap* Commerce Driver)

The Refund function is processed by a POST method & the API & values needed to proceed the Authorize and capture transaction is as follows

API: http://localhost:5050/api/Xepos/Evo/Refund

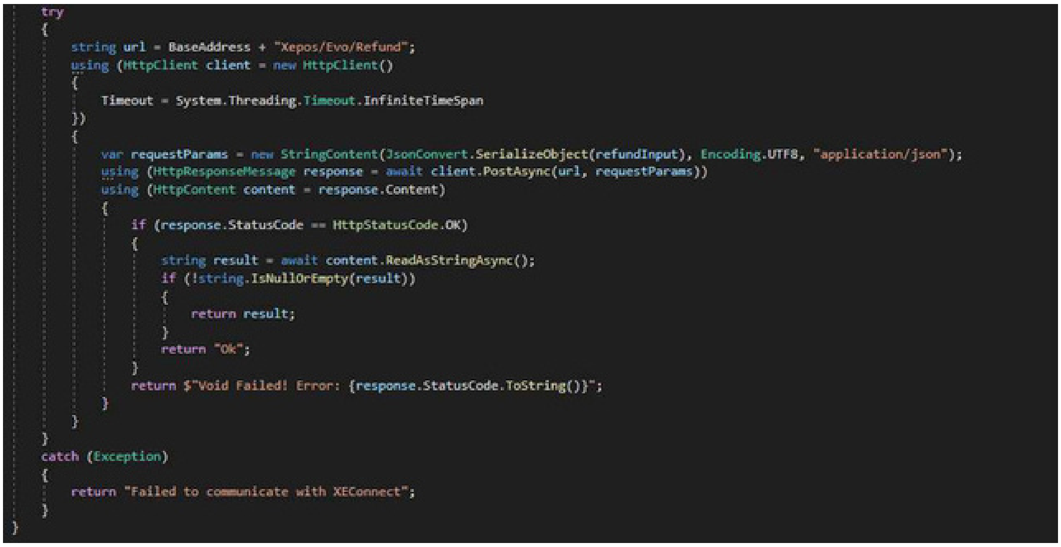
Values:



```
{
  "Amount": 0,
  "TransactionId": "string"
}
```

TransactionId: The value is provided from the response of the authorize and authorize & capture transactions

C# Sample Code:



```
    try
    {
        string url = BaseAddress + "Xepos/Evo/Refund";
        using (HttpClient client = new HttpClient()
        {
            Timeout = System.Threading.Timeout.InfiniteTimeSpan
        })
        {
            var requestParams = new StringContent(JsonConvert.SerializeObject(refundInput), Encoding.UTF8, "application/json");
            using (HttpResponseMessage response = await client.PostAsync(url, requestParams))
            using (HttpContent content = response.Content)
            {
                if (response.StatusCode == HttpStatusCode.OK)
                {
                    string result = await content.ReadAsStringAsync();
                    if (!string.IsNullOrEmpty(result))
                    {
                        return result;
                    }
                    return "Ok";
                }
                return $"Void Failed! Error: {response.StatusCode.ToString()}";
            }
        }
    }
    catch (Exception)
    {
        return "Failed to communicate with XEConnect";
    }
}
```

Return Unlinked (Refund)

(E-Service)

This process is like the Refund by ID and uses the same API but in E-Service mode the refund is handled at the terminal side and transactionID is optional

API: http://localhost:5050/api/Xepos/Evo/Refund

Transaction Info:

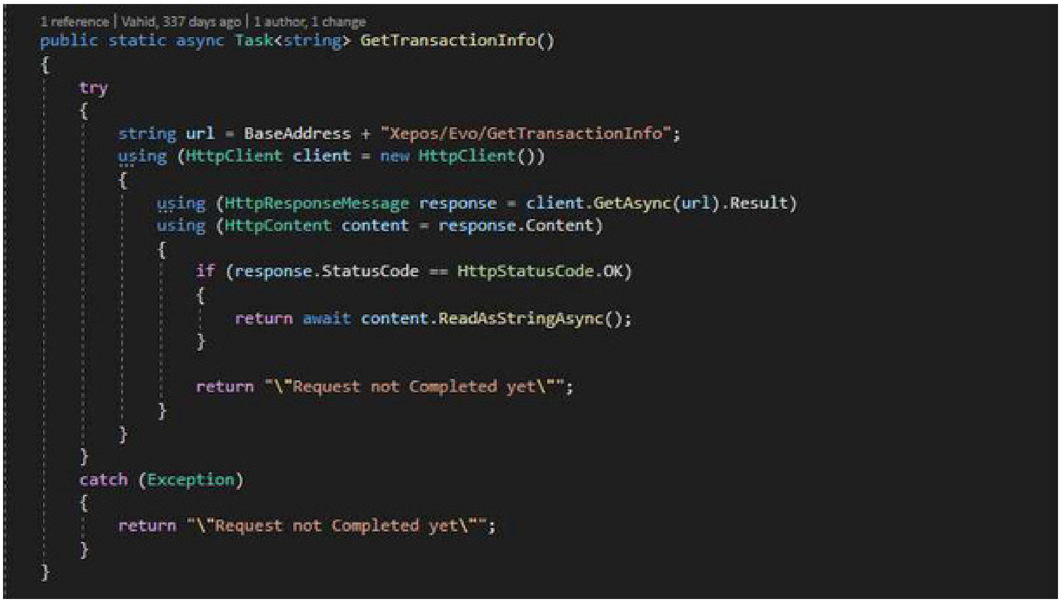
This function is processed by a GET method and should be called after each of the above functions to capture and store the response from EVO servers

API: http://localhost:5050/api/Xepos/Evo/GetTransactionInfo

- This API should be set in a loop for every 1 second, the response that you will receive is ‘Request not Completed yet’ this loop should continue until you don’t get this message anymore, the response that you will get from the server is either approved, cancelled or declined with the values listed below.

- The response of each transaction whether its approved, declined or canceled is also store in a file response.txt in the same directory of the software

C# Sample Code:



```
1 reference | Vahid, 337 days ago | 1 author, 1 change
public static async Task<string> GetTransactionInfo()
{
    try
    {
        string url = BaseAddress + "Xepos/Evo/GetTransactionInfo";
        using (HttpClient client = new HttpClient())
        {
            using (HttpResponseMessage response = client.GetAsync(url).Result)
            using (HttpContent content = response.Content)
            {
                if (response.StatusCode == HttpStatusCode.OK)
                {
                    return await content.ReadAsStringAsync();
                }

                return "\"Request not Completed yet\"";
            }
        }
    }
    catch (Exception)
    {
        return "\"Request not Completed yet\"";
    }
}
```

The values returned is as follows:

- StatusCode
- Approval Code
- Amount
- Status
- ApprovalCode
- AvsResultActualResult
- AvsResultPostalCodeResult
- TransactionStatusCode
- BatchId
- CvResult
- StatusMessage
- TransactionId

Optional APIs

Using the APIs below is optional but useful to have in your software

Auto Open XE-CONNECT: To open the XE-CONNECT™ automatically the following code can be used.

ProcessStartInfo start = new ProcessStartInfo {FileName = XeConnectFilePath};

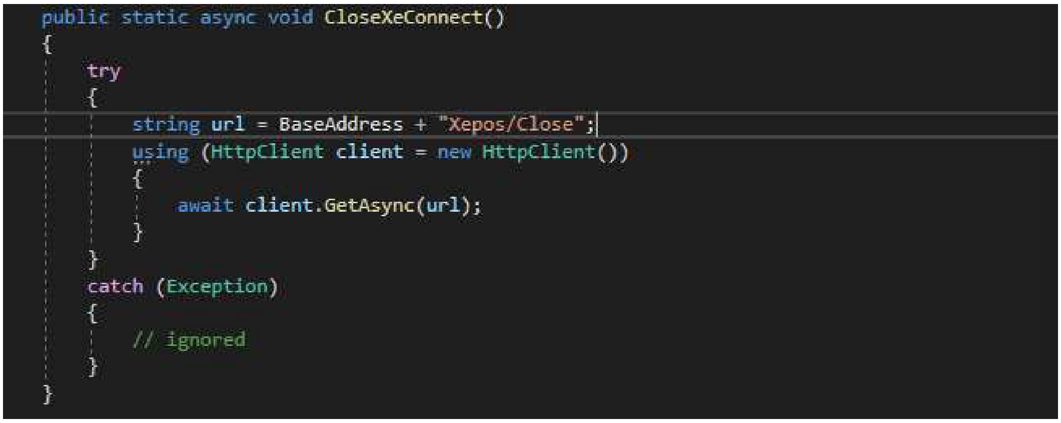
Process.Start(start);

XeConnectFilePath = @"C:\localprinters\XeposExternal.exe";

Closing XE-CONNECT: This process is done by a GET method and to close the XE-CONNECT™ app through your software you can use the API below

http://localhost:5050/api/Xepos/Close

C# Sample Code:



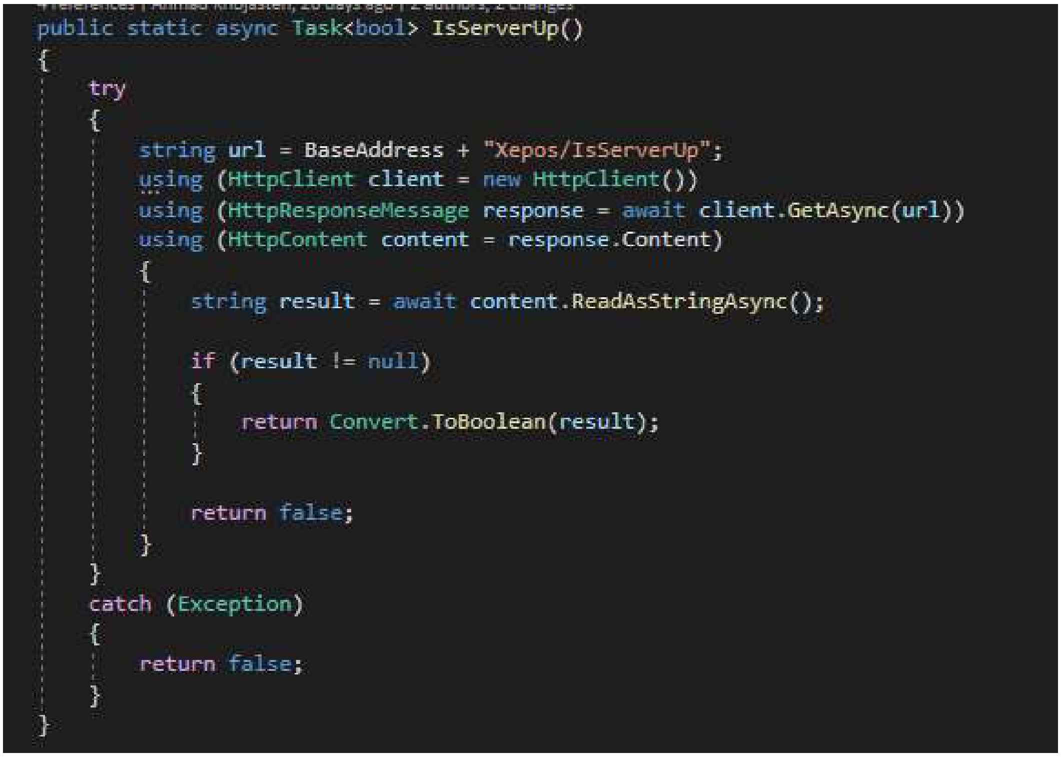
```
public static async void CloseXeConnect()
{
    try
    {
        string url = BaseAddress + "Xepos/Close";
        using (HttpClient client = new HttpClient())
        {
            await client.GetAsync(url);
        }
    }
    catch (Exception)
    {
        // ignored
    }
}
```

XE-CONNECT uptime: This process is done by a GET method and to ensure that xeposexternal is running in the background at all times this API can be used. The response is either True or False.

http://localhost:5050/api/Xepos/IsServerUp

- Its recommended to call this API before any transaction to make sure XE-CONNECT™ is running

C# Sample Code:



```
public static async Task<bool> IsServerUp()
{
    try
    {
        string url = BaseAddress + "Xepos/IsServerUp";
        using (HttpClient client = new HttpClient())
        using (HttpResponseMessage response = await client.GetAsync(url))
        using (HttpContent content = response.Content)
        {
            string result = await content.ReadAsStringAsync();

            if (result != null)
            {
                return Convert.ToBoolean(result);
            }

            return false;
        }
    }
    catch (Exception)
    {
        return false;
    }
}
```

Sample App

(Swagger):

All functions stated in the document can be tested using the following link.

http://localhost:5050/api/swagger

xepay.co.uk