# Installation Manual for PayUnity OpenCart

This manual describes the installation and usage of the PayUnity extension for OpenCart.

**Release Date:** Mon, 07 Aug 2023 14:01:06 +0200
**Version:** 6.0.46

wallee AG
General-Guisan-Strasse 47
CH-8400 Winterthur

E-Mail: info@sellxed.com

Telefon:
CH: +41 (0)44 505 13 60
DE: +49 (0)40 2999 60117
UK: +44 (0)161 850 6890
US: +1 (0)205 557 5869

# Table of Contents

# 1 Introduction

This manual explains the installation, configuration and usage of the payment module for OpenCart and PayUnity.

Before beginning with the installation, please make sure that you are in possession of all necessary data:

- User name and password for the login to the backend of PayUnity
- OpenCart payment module by sellxed.com/shop
- Access data to your server and shop

In case you don't yet have a contract with PayUnity, you can acquire it directly through us.

**Note that you must use at least PHP version 5.6 for our plugins. PHP 8 or higher is currently not supported.**

## 1.1 Installation Procedure

In this document you will find all important information for the installation of the module. It is important that you strictly follow the checklist. Only by doing so a secure usage in correspondence with all security regulations can be guaranteed.

1. Configuration of the test environment by means of the integration data from PayUnity. These can be found on the test platform under https://test.payunity.com/bip/login
2. Configuration of the basic settings of the payment module
3. Configuration of the payment methods
4. Carrying out of a test purchase by means of the attached test data at the end of this document
5. If the test was successful, you can configure the live data in your shop. Log into the live environment with the obtained access data under: https://payunity.com/bip/login

### Installation Service

Our payment plugins should have per default the correct settings for most of our customers' preferences. That means once you have entered the required credentials in the plugin configuration to connect your account to your website, the plugin should be fully operational. Should you be willing to receive detailed information on a setting you do not know, you may contact our support team who will be able to assist you further.

Our support team is at your disposal during regular business hours at: http://www.sellxed.com/support. Furthermore, you have the option of ordering our installation service. We will make sure the plugin is installed correctly in your shop: http://www.sellxed.com/shop/de/integration-und-installation.html

### .htaccess Directory Protection

In order to test the module, any kind of directory protection or IP blocking on your server must be deactivated. This is crucial as otherwise the payment feedback of PayUnity might not get through to the shop.

## 1.2 System Requirements

In general, the plugin has the same system requirements as OpenCart. Below you can find the most important requirements of the plugin:

- PHP Version: 5.4.x or higher
- OpenSSL: Current version with support for TLS 1.2 or higher.
- fsockopen: The PHP function fsockopen must be enabled. The plugin must be able to connect to external systems over the Internet.
- PHP Functions: All common PHP functions must be enabled.

In case you are using OpenCart version 3.0.3.5 or 3.0.3.6, you must patch your store to fix the twig extension error in order for modified templates to be loaded. The following extension can be used to resolve the issue: https://www.opencart.com/index.php?route=marketplace/extension/info&extension_id=40469.

# 2 Configuration

## 2.1 Basic configuration of the main module

The access data for the test environment (https://test.payunity.com/bip/login) is provided to you by PayUnity.

Navigate to the main module in your Shop and fill in the following access data:

- Authorization Bearer Token (Access Token)
- Entity ID (CHANNEL ID )
- The User ID and Password are depreacted and not required anymore, use the new Authorization Bearer Token instead.
- You no longer need the parameter **Security Sender and Security and Hash Secret** for the integration. You can simply ignore this entry.
- The boxes for "Security Hash" and "Security Hash Secret" can be left empty unless your Payment Service Provider instructs you to do otherwise

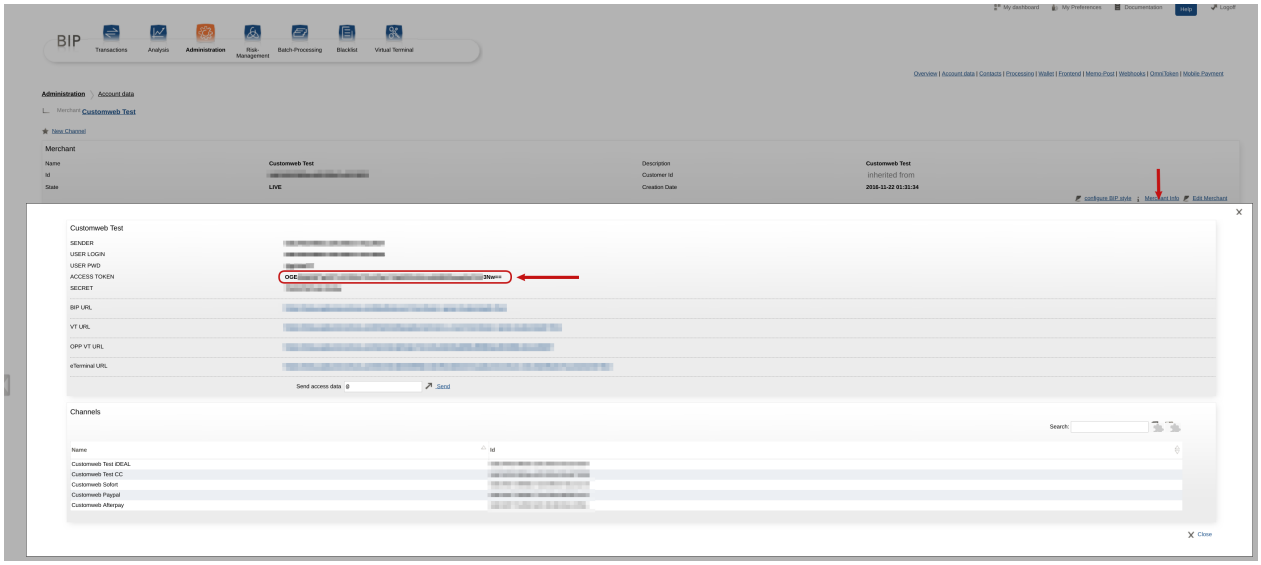


**Figure 2.1:** The Authorization Bearer Token in the PayUnity backend.

You also have more entry options. For example the **Custom Parameters**. In principle you leave these blank unlessPayUnity requires you to do otherwise.

## 2.2 Configuration of the Payment Methods

In order to activate the payment methods and to change other payment specific settings, navigate to the settings. (A description as to where to find these settings is listed here hier)

In the configuration of the payment methods you can define which Entity ID (Channel ID) should be used for which amount. This allows you to use a non-3D Secure Channel in case of small figures. In order for you to be able to set up and get the corresponding contract please contactPayUnity.

| | | |
|---|---|---|
| Channel Conditions | | [STORE VIEW] |
| | ▲ Beside the default channel ID a set of channel IDs depending on the order total amount can be defined. Each line must contain a lower amount, an upper amount and the applicable Channel ID. The format is as follow 'lower amount;upper amount;Channel ID' as shown in the following example: '10.00;200.00;234131132131312313 53'. The upper boundaries are not included. If you leave this field empty or if an amount is outside any range the default channel id is applied. If multiple conditions match the last one is taken. | |
| Entity ID (MoTo) | | [STORE VIEW] |
| | ▲ If this payment method should support recurring and moto transactions, you have to specify a second entity ID without 3D secure. | |
| COPYandPAY Style | Card | [STORE VIEW] |
| | ▲ Define the style to be used for the COPYandPAY payment form. | |
| Authorised status | Complete | [STORE VIEW] |
| | ▲ This status is set when the payment was successful and it is authorised. | |
| Uncertain status | On Hold | [STORE VIEW] |
| | ▲ You can specify the order status for new orders that have an uncertain authorisation status. | |
| Captured status | Don't change order status | [STORE VIEW] |
| | ▲ You can specify the order status for orders that are captured either directly after the order or manually in the back-end. | |
| Approved payment Status | Complete | [STORE VIEW] |
| | ▲ You can specify the order status for orders that are approved after being in an uncertain state. | |
| Denied Payment Status | Canceled PayUnity | [STORE VIEW] |
| | ▲ You can specify the order status for orders that are denied after being in an uncertain state. | |
| Authorisation Method | Server Authorization | [STORE VIEW] |
| | ▲ Select the authorisation method to use in order to process this payment method. | |

**Figure 2.1:** Configuration of the payment method using the example Magento (icon image, can differ from your version of the image).

> PCI: New Authorization Method **Widget**
>
> Please note that the use of the Hidden Mode comes with additional certification requirements (compare with our blog entry regarding this subject). For this reason we will no longer supply our new versions with the hidden mode. If you still use the hidden mode in you payment methods please solely use **Widget**. The support of the hidden mode will be terminated shortly.

## 2.3 Configuration Webhook

In some special cases it may happen that the payment notification sent by PayUnity can not be proceed by your OpenCart shop. We recommend to configure a so called webhook in the PayUnity backen under **Administration > Webbhook**. Please make sure that you configure the correct URL which you can find in your OpenCart shop under **Extensions > Modifications > Modules > PayUnity Base Module > Edit > More > Setup**.

### 2.3.1 Configuration Webhook by Merchant

If you have to configure the the webhook on your own, you can find the webhook URL in your OpenCart shop at **Extensions > Modifications > Modules > PayUnity Base Module > Edit > More > Setup** . Make sure you configure it in the PayUnity Backend under **Administration> Webbhook**. Furthermore you have to define a so called "secret for encryption" key. Make sure that you also save the same values in the corresponding fields, as on the Picture below.

> ## Customweb "secret for encryption" Generator
>
> Please note that only ASCII characters may be used for the generation of this key pass phrases. Please use our "secret for encryption" generator so that you do not enter invalid characters.

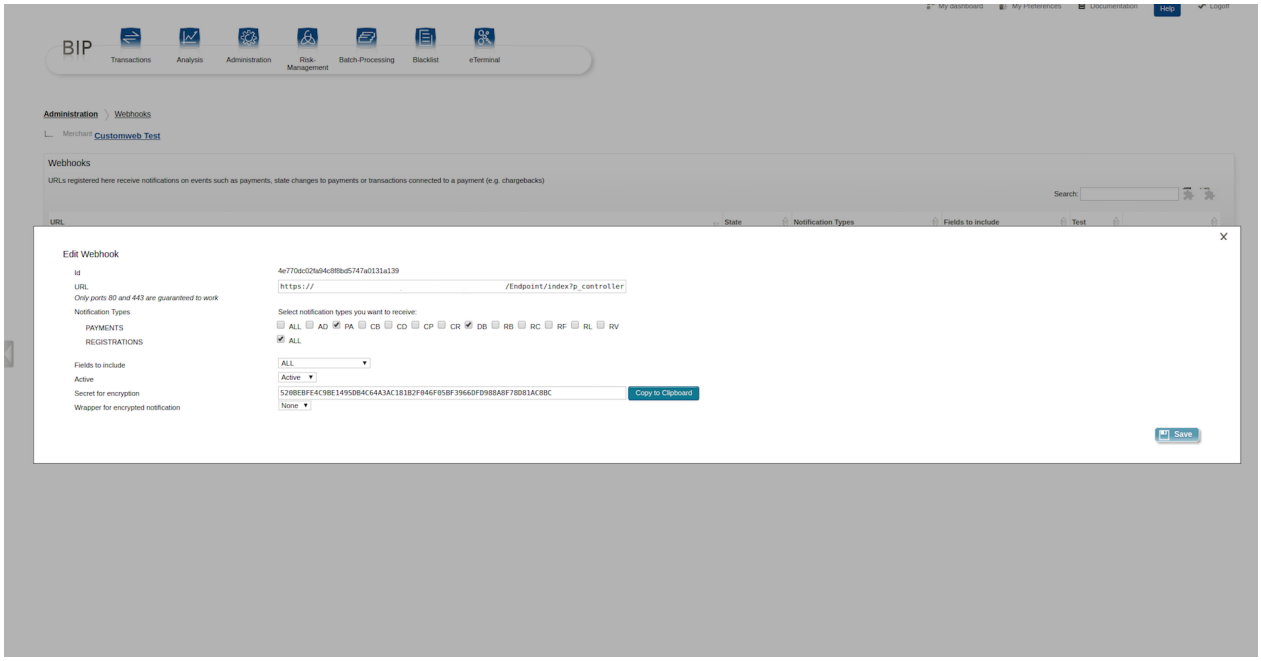


**Figure 2.1:** PayUnity Webhook configuration in the backend under **Administration > Webbhook**.

### 2.3.2 Configuration Webhook by PayUnity

In some PayUnity accounts you can not configure the webhook by yourself. Therefor you have to provide the webhook URL to PayUnity once they setup it for you, they will provide you a specific "secret for encryption" key, which you have to configure in the main module configuration of the module in your shop.

## 2.4 Activation and Testing

When you have activated and configured the payment methods you can run a test with help from the manual Test data.

If the tests were successful you now switch the operation mode from test mode to "Live Mode" and replace the safety features above with your personal access data for the live platform. You will receive this data directly from your Payment Service Provider. Normally the USER ID etc. should be identical for the Live Mode.

# 3 Module Installation and Update in the OpenCart Shop

## 3.1 Installation

At this time you should already be in possession of the module. Should this not be the case, you can download the necessary files in your customer account in the sellxed shop (Menu "My Downloads Downloads"). In order to install the module in your shop, please carry out the following steps:

1. Download the plugin. The download can be found in your sellxed.com account under "My Downloads".
2. Unzip the archive you have just downloaded.
3. In the unzipped folder navigate to the folder "files"
4. For some shops there are different versions of the plugin provided. If this is the case open the folder which corresponds to your shop version.
5. Using your preferred FTP client upload **entire content** of this folder into the root directory of your shop. For some shops there is a specific folder containing the plugins. If that is the case upload the plugin into this folder. Make sure that the folders aren't replaced but merely merged.
6. If you haven't yet done so, log back into your shop.

## 3.2 Updates and Upgrades

You have direct and unlimited access to updates and upgrades during the duration of your support contract. In order to receive constant information about available updates we ask you to subscribe to our RSS feed that we publish for your module.

More information regarding the subscription of this RSS feed can be found under: http://www.sellxed.com/en/updates_upgrades.

We only recommend an update if something doesn't work in your shop, if you want to use new feature or if there is a necessary security update.

### 3.2.1 Update Checklist

We ask you to strictly comply with the checklist below when doing an update:

1. Always do a backup for your database and your files in your shop
2. Use always a test system to test the update process.
3. Wait until all the files are copied to the shop, clear the cache if there is one in your shop and then visit the configuration page of the main module so that the update process will be initialized.

> ### Do not do updates directly in the live environment
>
> Please test the update procedure first in your test shop. Our support team is able and willing to help you if you experience problems with the update process. However, if you decide to perform the update directly in your live shop there is the possibility of a downtime of the shop of more than two days depending on the availability of our support if you do not want to book our complementary support.

Depending on the version it could be that the database has to be migrated. We recommend you therefore, to perform the updates in times when the shop is not visited too frequently by your customers.

### 3.2.2 Update Instructions

Please always read the update instruction. Those instructions can be found in the changelog. If there are no special remarks, you can proceed by just overwriting the files in your system.

# 4 Module Configuration in the OpenCart Shop

The configuration consists of two steps. The first step is the configuration of the main module with all the basic settings (cf. Configuration of the Main Module). During the second step you can then carry out individual configurations for each payment method . This allows for full flexibility and perfect adaptation to your processes.

### Create backups!

Please create a backup of the main directory of your shop. In case of problems you will then always be able to return your shop to its original state.

We furthermore recommend testing the integration on a test system. Complications may arise with third party modules installed by you. In case of questions, our support is gladly at your disposal.

# 5 OpenCart 3.0 Installation - Additions

In order to guarantee smooth operations and the usage of all features please make sure that you follow the instructions below.

## 5.1 Configuration of the Main Module

You will find the settings for the main module under **"Extension > Modules > PayUnity Base Module"** . Install the module by clicking **Install**.

By clicking **Edit** you can configure the main module. Enter all data in the corresponding fields. Each option is, furthermore, explained in short info texts in the shop.

## 5.2 Configuration of the Payment Module

After having successfully configured the main module, you can find the settings for the individual payment methods in your shop under **Extensions > Payments** . Each payment method is listed individually. Install the payment methods you wish to offer to your customers. You can carry out individual settings for each payment method and thereby optimally adapt the payment methods to your existing processes. The most central options are described in more detail further below.

By clicking on **Install** the payment method is activated in your shop. Click **Edit** in order to modify the configuration of the payment method.

## 5.3 Direct Capturing of Transactions

The option "Capture" allows you to specify if you wish to debit payments directly or if you first wish to authorise them and then debit the payment at a later point.

Depending on your acquiring contract, a reservation is only guaranteed for a specific period of time. Should you fail to debit the payment within that period, the authorisation may therefore no longer be guaranteed. Further information on this process can be found below.

> ### Different settings between PayUnity and the module
>
> It may be that settings saved in the payment modules overwrite settings saved in PayUnity.

## 5.4 Uncertain Status

You can specifically label orders for which the money is not guaranteed to be received. This allows you to manually control the order before shipment.

### 5.4.1 Setting the order state

For each payment method you may select in which state the order should be set to depending on the booking state. This is the initial state of the order.

## 5.5 Optional: Validation

Note: It can be that this option is not visible in your module. In this case just ignore this section.

With the option 'Validation' you can define the moment when the payment method should be made visible to the customer during the checkout process. This setting is relevant for modules where the usage depends on the customer's compliance with specific preconditions. For example, if a solvency check has to be carried out or if the payment method is only available in certain countries. In order for the credit check or address validation to also work with European characters, the charset of the "Blowfish mode" must be set to "UTF-8" for certain PSP settings.

You have the choice between these options:

- **Validation before the selection of the payment method:** A validation verification is carried out before the customer selects the payment method. If the customer does not fulfill the requirements, the payment method is not displayed
- **Validation after selection of the payment method:** The verification of the compliance occurs after the selection of the payment method and before the confirmation of the order
- **During the authorisation:** The validation verification is carried out by PayUnity during the authorisation process. The payment method is displayed in any case

# 6 Settings / Configuration of Payment Methods

## 6.1 General Information About the Payment Methods

The plugin contains the most common payment methods. In case a desired payment method is not included per default, please contact us directly.

In order to be able to use a payment method, it must be activated in your account with PayUnity as well as in your shop. Information about the configuration of the payment methods can be found further above.

Below you can find important information for specific payment methods that deviate from the standard process.

## 6.2 Information on Payment Status

For each payment method you can define an initial payment status (status for authorized payments etc.). You hereby define the payment status for each state depending on the processing type of the order (captured, authorized, etc.). It's the initial status which the order assumes. Depending on the mutation carried out by you, the status can change.

> Important info regarding Order Status
>
> Never set the status to **Pending PayUnity** or any similar pending status which is implemented by the module.

### 6.2.1 Order status "pending" / imminent payment (or similar)

Orders with the status 'pending PayUnity' are pending orders. Orders are set to that status if a customer is redirected in order to pay but hasn't returned successfully or the feedback hasn't reached your shop yet (Customer closed window on the payment page and didn't complete payment). Depending on the payment method these orders will automatically be transformed into cancelled orders and the inventory will be cleared (so long as the Cronjob is activated). How long this takes depends on the characteristics of the payment method and cannot be configured.

If you have a lot of pending orders it usually means that the notifications from your webserver to PayUnity are being blocked. In this case check the settings of your firewall and ask the Hoster to activate the IPs and User Agents of PayUnity.

### 6.2.2 Order status "cancelled"

Orders with the status "cancelled" have either been set to that status automatically due to a timeout, as described above, or have been cancelled directly by the customer.

## 6.3 Klarna

You can process Klarna directly viaPayUnity.In the following we will expand on the characteristics of the Klarna setup.

### 6.3.1 Supported / Unsupported Functions

The module does not support the following functions:

- You can not create or change any orders in the backend of OpenCart .
- Partial activation and refunds can only be performed through the backend of PayUnity ( https://payunity.com/bip/login) Full activation and full refunds can be performed directly inside OpenCart. For more information visit the Chapter around transaction management.

> **Edit Klarna transaction within OpenCart**
>
> Please make sure that you **never** modify or edit any Klarna transaction in your OpenCart backend.

The remaining functions are supported by the module. In the following you will find a description of the most important functions.

### 6.3.2 Set Up / Configuration of the Payment Method

Activate the payment method as usual. Please note the following regarding the installation:

#### 6.3.2.1 Displayed Name of the Payment Method

You may change the displayed name in the frontend via **title**. Please use the title **Invoice**. Furthermore, you can define whether the Klarna logo is visible to your customers in the frontend via **Display Logo**.

#### 6.3.2.2 Description of the Payment Method

You may change the description of the payment method in the frontend via **description**. Please make sure you display the following description in your frontend: "**Pay within 14 days**".

#### 6.3.2.3 Select Authorization Method

We recommend to set **Server** as authorization method. In this case you will not be redirected to the payment page of PayUnity.

#### 6.3.2.4 Klarna Merchant ID

In the payment method you can also define the **Klarna Merchant ID (EID)**. You will either find this in your Klarna account or ask for it directly at PayUnity.

### 6.3.3 Canceling the Invoice

How to cancel invoices will be explained below in the chapter Activating / Canceling Orders. For further questions please refer to this section.

### 6.3.4 Full Activating Invoices

For more information visit the Chapter around transaction management.

### 6.3.5 Partial Refund and Partial Activation

Partial refunds and partial activations can only be done through the BIP of PayUnity.

### 6.3.6 On Hold Orders (Pending)

Pending orders are not supported by the module. In order you want those to be handled correctly by the system, please leave the field "Status Check interval" blank in your BIP. Please contact PayUnity to make this setting.

### 6.3.7 Testing

In order for you to test the payment method Klarna you will need specific test data. But first make sure the the **operation mode is on test and the test mode is on external**.

You will find the official test data for Klarna directly in the developer portal via this link: https://developers.klarna.com/en/de+php/kpm/test-credentials.

### 6.3.8 Payment fees

If you want to charge payment fees for a specific payment method, feel free to use the compatible payment fees modules. Those fees are directly transmitted to Klarna.

> **Payment Fees**
>
> Please make sure to not charge payment fees within the PayUnity BIP Backend. Please use exclusively the compatible modules as described above.

# 7 The Module in Action

Below you will find an overview of the most important features in the daily usage of the PayUnity module.

## 7.1 Useful Transaction Information on the Order

You can find an overview of the transaction information in within the order detail view. Among others, this information allows for the definite attribution of the orders to their corresponding transaction, seen in the backend of PayUnity.

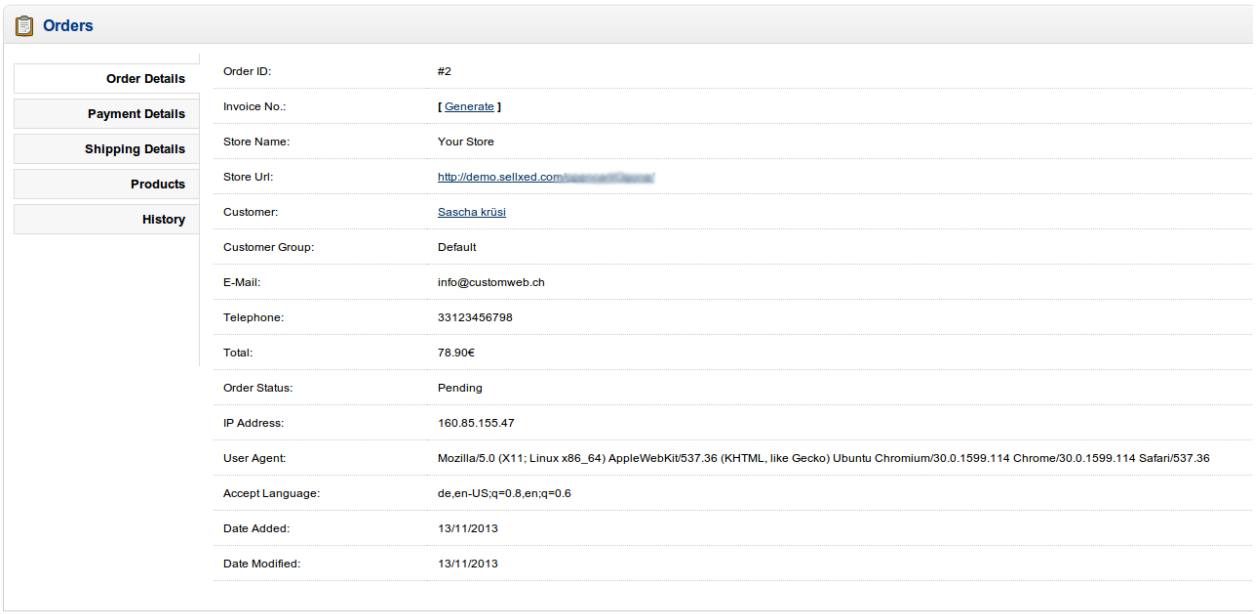


**Figure 7.1:** Transaction Information in OpenCart.

## 7.2 Usage of the Alias Managers / Token Solution

With the Alias Manager, your customers can securely save their credit cards with PayUnity for later purchases. You can enable this by activating the option "Alias Manager" in the configuration of the Payment Method. The customer can then choose from his or her saved credit cards without having to re-enter all the details.

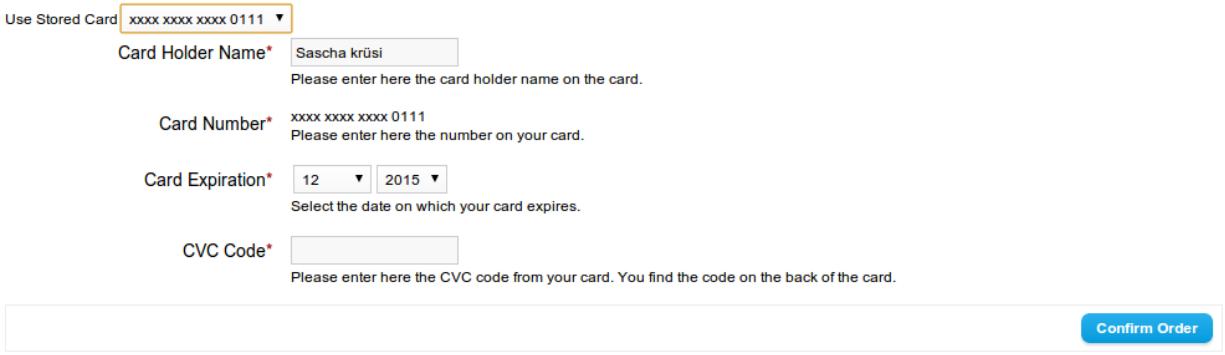


**Figure 7.1:** Alias Manager

### Alias Manager Options

The usage of the Alias Managers requires the activation of the correct option with PayUnity. To do so, please contact the support directly.

## 7.3 Capturing / Cancelling of Orders

### 7.3.1 Capturing Orders

In order to capture orders, open the transaction manager under Sales > PayUnity Transactions. Here you will find an overview of all transactions. Search for the order in the field with the order numbers. By clicking "view" you will open the transaction of the order.

#### 7.3.1.1 Capturing Complete Orders or Partial Capturing

By clicking the button "Capture Transaction" a new window opens up. You can now capture every item individually in case you do not wish to send all items at once. The amount of possible partial captures depends on your contract. Please contact PayUnity directly in order to clarify questions.

### Capturing of Orders in the backend of PayUnity

The transaction management between your shop and PayUnity is not synchronised. If you capture payments with PayUnity, the status in the shop will not be updated and a second capturing in the shop is not possible.

In case you do not wish to capture all items of an order, you can close the transaction by clicking the box.

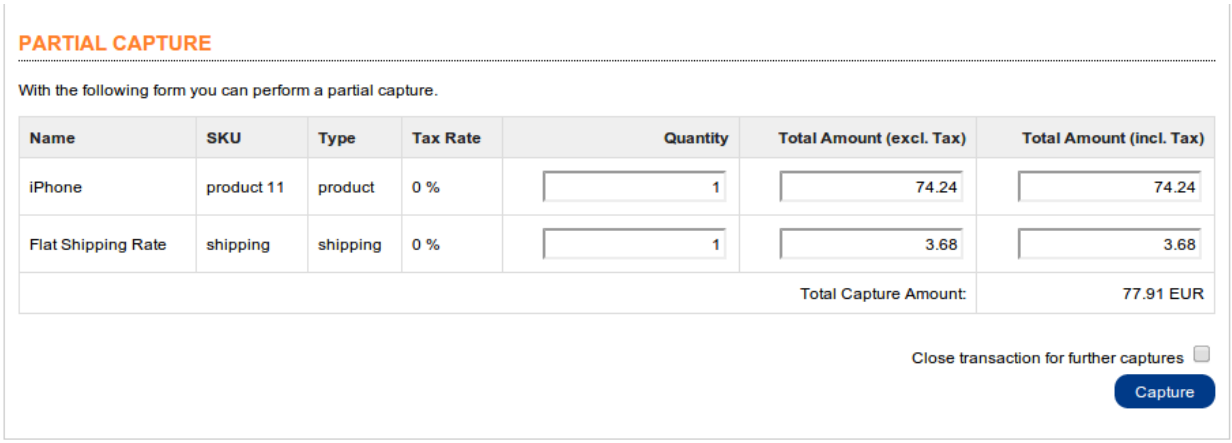


**Figure 7.1:** Capturing of Orders

> ### Partial Capturing
>
> Please find out if the capturing of partial amounts is supported by your PayUnity contract. If this is not the case, it might happen that the transaction is closed for further transactions after a partial capture.

### 7.3.2 Cancel Orders

By clicking "Cancel Transaction", the transaction is **cancelled** and the reserved amount on your customer's card is released automatically.


**Figure 7.1:** Capture or Cancel in OpenCart.

## 7.4 Refunding Orders

In order to refund orders, open the transaction information (cf. above).

You can refund individual items or any amount of your choice by modifying the total amount or the item quantity.

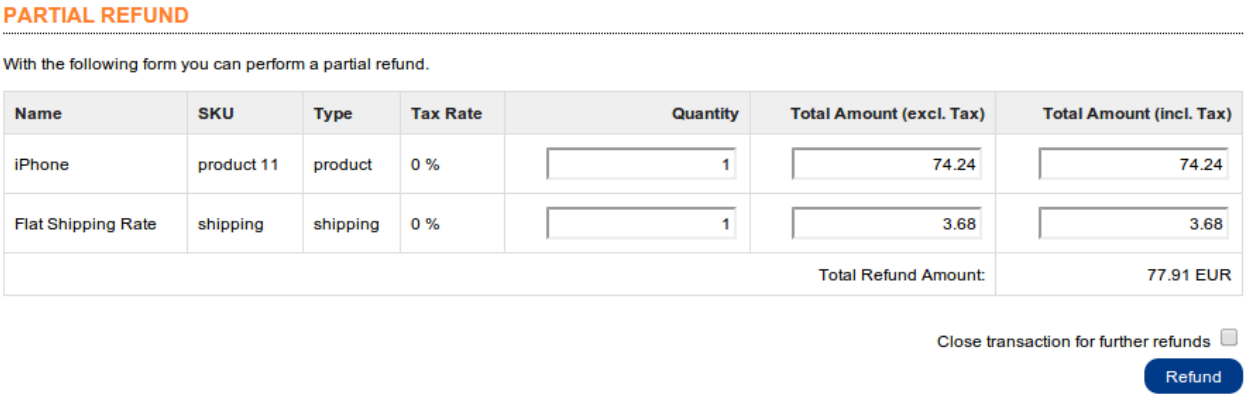


**Figure 7.1:** Refunds in OpenCart for PayUnity.

> ### Maximal Refund
>
> With our module it is not possible to refund more than 100% of the originally authorised amount.

Executing a refund will not change the status of the order.

## 7.5 Setup a Cron Job to Activate the Timed Operations

To activate the timed operations of the plugin (e.g. update service, deleting pending orders, etc.) make sure that you set up the OpenCart Cron engine. Especially the update function allows you to automatically retrive additional information or changes of your order directly via the API of PayUnity. Please note it could be that in order to use the update feature it may be necessary that PayUnity activates additional options in your account.

In order to use the timed operations, please schedule a cron job in your server to the following controller:

https://www.your-shop.com/index.php?route=payunitycw/cron/cron

Here we suggest you use a Cron Engine like for example EasyCron. That way you can This allows you to open the file ( URL ) with an external service.

# 8 Testing

Before switching from test to live mode it is important that you test the module extensively.

> **Testing**
>
> Do not forget to switch the operating mode from test to live after having successfully tested the module.

## 8.1 Test Data

In the following section you can find the test data for the various payment methods:

### American Express

| | | |
|---|---|---|
| Card number | 377777777777770 | No 3D Secure |
| Expiry Date | 12/2020 | |
| CVC | 123 | |
| Card number | 375987000000005 | 3D Secure |
| Expiry Date | 12/2020 | |
| CVC | 123 | |

### Carte Bleue

| | |
|---|---|
| Card number | 5555555555554444 |
| Expiry Date | 12/2020 |
| CVC | 123 |

### Dankort

| | |
|---|---|
| Card number | 5019717010103742 |
| Expiry Date | 12/2020 |
| CVC | 123 |

### Diners Club

| | |
|---|---|
| Card number | 36961903000009 |
| Expiry Date | 12/2020 |
| CVC | 123 |

### Sepa Direct Debits

| | | |
|---|---|---|
| IBAN | AT152011128161647502 | Austria (AT) |
| BIC | GIBAATWWXXX | |
| IBAN | DE23100000001234567890 | Germany (DE) |
| BIC | MARKDEF1100 | |
| IBAN | ES9121000418450200051332 | Spain (ES) |
| BIC | CAIXESBBXXX | |

### Discover Card

| | | |
|---|---|---|
| Card number | 6011587918359498 | |
| Expiry Date | 12/2020 | |
| CVC | 123 | |

### giropay

| | | |
|---|---|---|
| IBAN | AT152011128161647502 | Austria (AT) |
| BIC | GIBAATWWXXX | |
| IBAN | DE23100000001234567890 | Germany (DE) |
| BIC | MARKDEF1100 | |
| IBAN | ES9121000418450200051332 | Spain (ES) |
| BIC | CAIXESBBXXX | |

### JCB

| | | |
|---|---|---|
| Card number | 3541599999092431 | |
| Expiry Date | 12/2020 | |
| CVC | 123 | |

### Klarna Invoice

| | | |
|---|---|---|
| | | Klarna (Approved) |
| | | Klarna (Denied) |
| | | Klarna (Pending -> Approved) |
| | | Klarna (Pending -> Denied) |

### Maestro

| | | |
|---|---|---|
| Card number | 6799851000000032 | |
| Expiry Date | 12/2021 | |
| CVC | 123 | |

### MasterCard

| | | |
|---|---|---|
| Card number | 5454545454545454 | No 3D Secure |
| Expiry Date | 12/2021 | |
| CVC | 123 | |
| Card number | 5212345678901234 | 3D Secure |
| Expiry Date | 12/2021 | |
| CVC | 123 | |

### Visa

| | | |
|---|---|---|
| Card number | 4200000000000000 | No 3D Secure |
| Expiry Date | 12/2020 | |
| CVC | 123 | |
| Card number | 4012001037461114 | 3D Secure |
| Expiry Date | 12/2020 | |
| CVC | 123 | |
| Card number | 4000000000000010 | 3D Secure 2.0 |
| Expiry Date | 12/2021 | |

| | |
|---|---|
| CVC | 123 |

**V PAY**

| | |
|---|---|
| Card number | 4822000000000000003 |
| Expiry Date | 12/2020 |
| CVC | 123 |

# 9 Errors and their Solutions

You can find detailed information under http://www.sellxed.com/en/faq. Should you not be able to solve your problem with the provided information, please contact us directly under: http://www.sellxed.com/en/support

## 9.1 The Referrer URL appears in my Analytics Tool

When a customer and the notification are redirected via Header Redirection, the PayUnity Referrer URL might appear in your Analytics Tool thus hiding the original traffic source. However, most Analytic Tools are able to minimize this problem.

In case you are using Google Analytics as reporting tool, this step by step guide may help you to exclude the URLs: under bullet point 4.

# 10 Compatibility with Third-Party Plugins

The plugins listed below are compatible with our payment modules and allow you to handle certain tasks in an easier way.

## 10.1 Birthday and gender in OpenCart

For certain payment service providers it is necessary to check the birthday and the gender of a customer. OpenCart does not check this by default.

### How to enable gender and birthday checks in your shops checkout

1. Add two new custom fields to your checkout via your shops backend under "Customers > Custom Fields"
2. Modify the order context getters to return the value of your custom checkout field from the order / session (or wherever the previous step saves the data).

   Order Context Getters

   - AbstractOrderContext
   - getBillingDateOfBirth()
   - getBillingGender()

   These functions can be found in "system/library/cw/PayUnity/AbstractOrderContext.php".

# 11 Error Logging

The module will log different unexpected errors or information depending on the configured level. If there is any issue with the module, this log can help identify the cause.

## 11.1 Log Levels

You can configure the log level in the PayUnity settings.

- Error: Logs unexpected errors only. (Default)
- Info: Logs extended information.
- Debug: Logs information helpful for debugging.

## 11.2 Log Location

The log file is stored in the default log folder of OpenCart. The path is configured in the config.php of your shop system. (Default Path: {shopRootDirectory}/system/logs or {shopRootDirectory} /system/storage/logs)

# 12 Advanced Information

This section of the manual is for advanced usage of the module. The content is for advanced users with special requirements. Everything in this section is optional and not required for the daily usage of the module.

## 12.1 Transaction Object

This section describes how to extract information from a transaction, if you need it for further processing. E.g. you require more information of the transaction for further processing an order in your ERP system.

The code snippets in this section assume your script resides in the root folder of the shop with the default shop folder structure.

In your script initialize the base of OpenCart.
**Opencart 3.x**

```
require_once('config.php');
require_once(DIR_SYSTEM . 'startup.php');
// Registry
$registry = new Registry();

// Config
$config = new Config();
$config->load('default');
$config->load('catalog');
$registry->set('config', $config);
$loader = new Loader($registry);
$registry->set('load', $loader);
$registry->set('db', new DB($config->get('db_type'), $config->get
('db_hostname'), $config->get('db_username'), $config->get
('db_password'), $config->get('db_database'), $config->get('db_port')));
```

Include the module specific files and set registry.

```
require_once DIR_SYSTEM.'library/cw/init.php';
require_once DIR_SYSTEM.'library/cw/PayUnityCw/Util.php';
require_once DIR_SYSTEM.'library/cw/PayUnityCw/Entity/Transaction.php';
PayUnityCw_Util::setRegistry($registry);
```

Now you can load the transaction and then extract the transactionObject.
Load the transaction by Id:

```
$transactionById = PayUnityCw_Entity_Transaction::loadById
($transactionId);
$transactionObject = $transactionById->getTransactionObject();
```

Load transactions by Order ID:

```
$transactionsByOrderId = PayUnityCw_Entity_Transaction::
getTransactionsByOrderId($orderId);
foreach($transactionsByOrderId as $transaction){
    $transactionObject = $transaction->getTransactionObject();
    //Do something with each object
}
```