eZ

Paynet Terminal 1.x

eZ Publish Extension Manual

Contents

List of Figures

Introduction

The Paynet Terminal Extension provides a solution to connect an eZ Publish shop to the Paynet's card payment system. The extension redirects customers from the eZ Publish site to the Paynet site for their payment. After the payment, the customer is redirected back to the eZ Publish System, where the shopping traject continues.

This document explains how you can

- set up the extension,

- configure and test it.

The terminal gateway interface described here does not allow the seamless integration of the Paynet service into eZ Publish - this can be achieved with the also available Paynet Direct extension.

Chapter 1

Concepts and basics

Paynet is a payment service provider which operates on the European Market. Partnerships are made with the largest card payment acquirers in Norway as well as in other Nordic countries. Paynet supports all currencies for Visa and Mastercard. The default languages are Norwegian and English. Other languages are available upon request.

The Paynet redirect solution (coupled to eZ Publish) redirects customers to the Paynet Terminal for their payment. The following diagram illustrates the interaction of the involved parties:

(see figure 1.1)

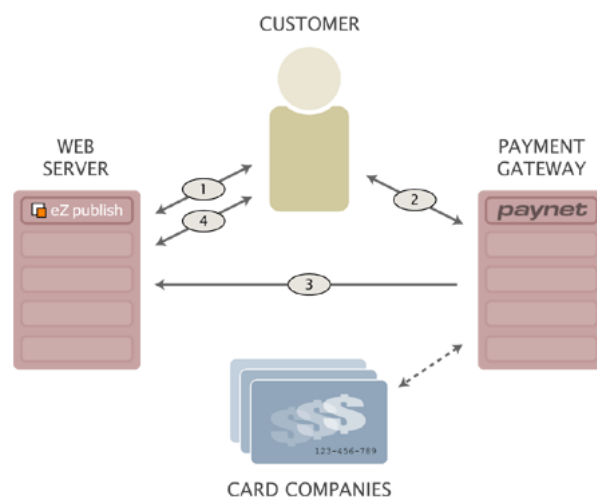Figure 1.1: *Diagram shows the cooperation among the web-server, customer, payment gateway, and card companies for a payment transaction.*

The parties shown in the diagram are:

Party	Description
Customer	Uses the webbrowser to do the checkout in the eZ Publish shop.
eZ Publish	Provides the shop which is stored on a web-server.
Paynet	Provides a payment gateway, which will

	handle our payments and acquires the money from the euroConnex or Teller.
Card acquirers	Have a relationship with Paynet to enable them to accept card payment. Currently, the acquireres EuroConnex and Teller can be selected to handle the payment for Paynet.

The arrows in the illustrate the interactions between the different parties. The numbers at the arrows define the sequence of the interactions:

1. The customer does a checkout from the products he or she wants to order. After the customer has filled in its address information, the customer is redirected to the Paynet Terminal.

2. The customer has to fill in his credit card information on the Paynet Terminal site. If the credit card seems to be valid, the customer is redirected back to the eZ Publish webshop.

3. The customer gets a summary of the order when the transaction was succesfull. Otherwise the customer is offered to checkout again.

4. Paynet will send a notification about the, successful or failed, payment to eZ Publish. This notification is also known as a callback. eZ Publish relies on the callback to approve or decline orders.

The redirects from eZ Publish to Paynet are configured at the eZ Publish site. Redirects from Paynet to eZ Publish are configured at the Paynet interface. The notification is configured at both sides. More information about the configuration is described in the Configure section.

Chapter 2

Requirements

Before you can start to install the eZ Publish Paynet Terminal extension, the following requirements have to be met:

- The extension needs a specific server environment configuration to work properly.

- A Paynet merchant account must be created for production use.

These points will be explained in detail below.

Preparing the Web server

Make sure you have installed PHP 4.3.4 or higher on the eZ Publish host server and that it runs eZ Publish version 3.6 or higher.

Furthermore, Paynet must be able to reach a certain URL within the eZ Publish installation when redirecting from the Paynet terminal to the eZ Publish shop. Due to the fact that your shop is connected to the Internet, meeting this requirement is not a problem.

Registering a Paynet merchant account

You must be a registered Paynet merchant if you want to make use of the Paynet service in a production environment. This means you need to have an agreement with Paynet and with a credit card company: euroConnex, Teller, or both. Paynet provides you a merchant number and access to their Terminal administration interface, which is later used to configure the Paynet Terminal extension.

More information about becoming a Paynet merchant can be found at http://www.paynet.no/apply.

For development or testing purposes, bogus transactions can be done with the paynet-demo-key certificate as described in the Testing section (page 12).

Chapter 3

Installation

To install the Paynet Terminal extension, the following steps need to be performed in the given order:

1. exctract the compressed archive
2. activate the extension
3. add a workflow and trigger
4. configure the extension

The first two steps are explained in the extension installation chapter, the rest will be explained here.

Add a workflow and trigger

The Paynet Terminal extension is hooked into the eZ Publish system as a workflow event. This means that once a shopping checkout is executed, the Paynet Terminal routines will be triggered as part of a eZ Publish workflow. To achieve this functionality, a new workflow as well as a trigger need to be defined. This can be done conveniently in the eZ Publish administration interface:

Add a workflow

Navigate to "Setup > Workflows" and add a new workflow to the CMS. Choose a name like "Paynet Terminal Workflow" and select the "Paynet / Payment Gateway" event from the list of events. Hereafter click on the "Add event" button. If you like you can optionally add a description, then press the "OK" button.

Add a trigger

Go to the list of triggers located at "Setup > Triggers". Here, select the previously created workflow "Paynet Terminal Workflow" from the "shop checkout before" drop-down list, and press "Apply changes".

Configure the extension

After a fresh install of the extension, its configuration must be updated in order to make it work properly. The available configuration options (page) are explained in the next section.

3

Chapter 4

Configuration

Due to the fact that the Paynet Terminal extension is related to a third-party service outside of the eZ Publish installation, configuration has to be done at two places:

- Configuring the eZ Publish Paynet Terminal extension
- Configuring the Paynet Terminal settings on the Paynet Web site

Configuring the extension

The settings for the ezpaynet extension are stored in the file "paynetterminal.ini" which is located in the "settings/" directory of the extension, for example.

```
/ezpublish/extension/ezpaynetterminal/settings/paynetterminal.ini
```

Now comes a list and explanations of available configuration options.

setting	description
ServerIP[]= *IP*	The *IP* of the server which is allowed to send us the receipt. This is most frequenty the IP number of the paynet server. If the ServerIP variable is not present in the settings, the IP check will be skipped.
ShopID= *number*	The Shop ID you will get from Paynet. For testing you can use the ShopID: 00000001. If you get an error that the Merchant ID does not exist, make sure that you use the correct currency.
ShopSubID= *number*	You should use the ShopSubID for testing only. This ID will identify your shop in their test system. Comment the ShopSubID if you are not using the eZPaynet testsystem.
Language= *language code*	The two character language code indicates the language that the Paynet redirect page will use.
SecretWord= *string*	This secret word is a string which is only

	known by Paynet and you (the merchant). This word ensures that the callback from Paynet to eZ Publish is coming from a trusted source (Paynet). This SecretWord can be generated on the Paynet site and should be regenerated when you suspect that your key is compromised.

Configuring Paynet

The settings of the Paynet Terminal system can be changed as described here. Login to this system at http://www.paynet.no and make sure that the following settings correspond with your eZ Publish configuration:

- Shop id

- Department id (If 0000, comment the ShopSubID in your ezpaynet settings.)

- Currency (The currency must correspond with your locale currency.)

It is important that you use the following settings:

Setting	Description
Site URL	The URL of your site. This URL will be shown in the payment interface. Make sure that the URL starts with http!
Complete URL	The URL to which customers are redirected after the payment. Set this URL to:http:// *[www.yoursite.com]*/paynetterminal/ redirect/complete. Versions prior to PaynetTerminal 1.1 should redirect to: http:// *[www.yoursite.com]*/ shop/checkout/
Abort URL	The URL called when the user clicks on the abort button during the payment. Set this URL to:http:// *[www.yoursite.com]*/ paynetterminal/redirect/abort Versions prior to PaynetTerminal 1.1 should redirect to: http:// *[www.yoursite.com]*/ shop/basket/
Receipt URL	The callback URL that notifies eZ Publish of suc- cesfull or failed payments. Set this URL to *[www.yoursite.com]*/paynetterminal/receipt

After a configuration update, you should do some test payment-transactions (page 12).

Chapter 5

Testing

This section describes simple testing procedures which allow you to verify the proper functioning of the installed extension.

Test merchant

First, you could do some test transactions with the Paynet demo merchant. Set in "paynet-terminal.ini" your "ShopID" to "00000001". Go to the Paynet test server and choose a "Shop-SubID" that is not in use. (At the time of writing the URL to the paynet testserver is: "https://www.paynet.no/term193/".)

On the Paynet Web site, configure the URLs for receipt, abort and completion.

Creditcard validation

Once everything is configured, you can do some test transactions by mimiking a customer in the eZ Publish online shop. When it comes to the checkout, you can provide

- a random number, which will fail.

- choose a test creditcard number: 5555555555554444 or 4444333322221111 with any expiry date, which should succeed.

Make sure that you will be directed to the correct URL before and after the checkout, especially when creditcard validation failed or continued successfully.

Failed transactions

Failed transactions can be faked by playing with the redirected Paynet URL. If you change the values "id", "subid", "currency", "amount", or "utref", the transaction will fail. This is visible in the "eZPaynetChecker.log" file.

If you change the "Receipt URL" on the Paynet server to an invalid URL, the customer will get a time-out after the payment. This time-out page should contain information how the customer can contact you.

Chapter 6

Trouble shooting

eZ Paynet Payment Gateway is not activated

The eZ Paynet Payment Gateway is not activated in eZ Publish.

You will need to check if the extension is properly activated in site.ini. Edit the
"site.ini.append.php" configuration file located in the "settings/override/" directory (or "settings/siteaccess/example/") and make sure you have have the following entry under the "[ExtensionSettings]" group:

```
ActiveExtensions[]=ezpaynet
```

Make sure there are no spaces before or after the name.

Consider also checking the debug output from eZ Publish to find out what is wrong.

**Browser hangs on the link: https://secure.edb.com/d3SecureAuth/
bankid?javaversion=1.5.0_04&os=WIN**

*The browser hangs on the following link: https://secure.edb.com/d3SecureAuth/
bankid?javaversion=1.5.0_04&os=WIN with the status set to "done".*

This problem may occur when the customer has a Verified by Visa enabled credit card. The problem is not
in eZ Publish nor in PaynetTerminal, but in the authentication site. (Paynet redirects the browser to the
authentication site and further redirection stops.)

To prevent this the cardholder has to lower the "privacy level" in Internet Explorer. The authentication site redirects from mpi1.3dsecure.no to secure.edb.com. In that redirect they use cookies. IE denies to use that cookie and the authentication page won't load properly.

Two solution (and they both depends on the cardholder):

1. lower "privacy level" in ie browser or use another browser could also help
2. manually add that you accept cookies from 3dsecure.no and edb.com

Empty orders appear in the administration interface

Very occasionally an empty order appear in the administration interface

This problem happens in PaynetTerminal prior to version 1.2. Occasionally the Paynet server does not get a reply from the PaynetTerminal extension. Processing the request might be to slow,
a problem on the network, or something in Paynet goes wrong. Anyhow, the Paynet server will send a new order confirmation, which caused the PaynetTerminal extension to behave in a way
it shouldn't.

To solve the problem, do either the first or the second solution:

- Get version 1.2 or higher (recommended)

- Add a check for duplicate receipts in the PaynetTerminal extension.

This check is in PaynetTerminal version 1.2 as follows:

```
Index: modules/paynetterminal/receipt.php
===================================================================
--- modules/paynetterminal/receipt.php  (revision 9686)
+++ modules/paynetterminal/receipt.php  (working copy)
@@ -59,7 +59,7 @@

     $orderID    = $checker->getFieldValue( 'utref' );

-    if( $checker->requestValidation() )
+    if( $checker->requestValidation() && !$checker->isAlreadyApproved(
$orderID ) )
     {
         $currency   = $checker->getFieldValue( 'currency' );
         $amount     = $checker->getFieldValue( 'amount' );
Index: classes/ezpaynetchecker.php
===================================================================
--- classes/ezpaynetchecker.php (revision 9686)
+++ classes/ezpaynetchecker.php (working copy)
@@ -87,6 +87,35 @@
         return false;
     }


+    /*!
+       Sometimes Paynet doesn't a response of their receipt callback in
time. Therefore the
+       callback is resend and caused the eZ Publish shop to show
NULL-Orders in the administration
+       interface.
+       This method makes sure that the order is not approved already.
+    */
+    function isAlreadyApproved( $orderID )
```

```
+    {
+        $paymentObject = eZPaymentObject::fetchByOrderID( $orderID );
+
+        if ( $paymentObject === null )
+        {
+            $order = eZOrder::fetch( $orderID );
+
+            if( $order->attribute("order_nr") > 0 )
+            {
+                $this->logger->writeTimedString( "The order is already
processed. This is probably caused by a duplicated receipt callback from the
Paynet server." );

+
+                return true;
+            }
+            else
+            {
+                $this->logger->writeTimedString( "The paymentObject IS set,
but the order number is NOT set for eZOrder::OrderID: $orderID." );

+            }
+        }
+
+        return false;
+    }
+
    // overrides
    /*!
        We use a hash to validate the callback. So there is no need to
build a
```