Paynet Direct 1.x

eZ Publish Extension Manual

Corrections and/or suggestions might be sent to `info@ez.no`.

This PDF file is generated automatically from the online documentation available at `http://doc.ez.no`.

This version was generated on September 30, 2014.

Contents

List of Figures

Introduction

This manual covers the ”Paynet Direct” extension and explains how it can be installed, configured and tested. The ”Paynet Direct” extension provides a direct interface to the online credit card payment system of Paynet. Paynet is a payment service provider that operates within the European market. Partnerships are made with the largest card payment companies 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 solutions are system-neutral and do not require any proprietary software on the client or the server side.

The ”Paynet Direct” extension plugs into the built-in e-commerce engine of eZ Publish and thus provides a seemless integration of eZ Publish and Paynet. The extension allows customers to purchase goods that are available in an eZ Publish web solution by making use of the services that Paynet provides. The following list outlines the topics that are covered by this manual.

Please note that the ”Paynet Direct” extension does not redirect the customer to the Paynet site. In other words, the customer will always stay within the same domain/site (the eZ Publish web solution) and thus the payment is carried out in the smoothest possible way. The customer can be redirected to the Paynet site by making use of the ”Paynet Terminal” extension. Both extensions are included in the ”Paynet Gateway” package, which can be purchased from eZ Systems.

Chapter 1

Requirements and preparation

The ”Paynet Direct” extension will only work if the following requirements are met:

- The web server must be configured in a special way.
- The eZ Publish version must be 3.5.0 or higher.
- For production use, a Paynet merchant account is needed.

Web server preparation

The web server needs to be configured in a special way. The following list reveals the most important issues that have to be taken care of.

- The web server must be able to access the Paynet server.
- The web server should use PHP 4.3.2 or higher.
- CURL support must be installed for PHP.
- The web server should be SSL-enabled.

Access to Paynet server

The web server on which eZ Publish is running must be able to access the Paynet server using HTTPS on port 8210. In particular, eZ Publish will attempt to access the following address: https://www.paynet.no:8210/if2. This means that the outer firewall must be opened, the local firewall (which is running on the web server itself, if any) must be configured, and so on.

Testing access

The ”telnet” command line application (available on both Linux/UNIX and Windows systems) can be used to test whether the web server (the machine running the eZ Publish solution) has access to the paynet service or not. Instructions:

1. Bring up a shell / command line interface.
2. Attempt to access the paynet server using port 8210:

```
telnet paynet.no 8210
```

   A connection should be established. On Linux/UNIX systems, this is indicated with a line saying ”connected to paynet.no.”. On Windows, an empty command window will be displayed.

3. Use the established connection to send a bogus GET request to Paynet:

```
GET / HTTP/1.0
```

   There is a space between the first slash and HTTP. Hereafter press enter a couple of times.

4. The Paynet server should reply with something like ”Your browser sent a request that this server could not understand”. You should be able to see the HTTP response.

If everything works as described in the list above, it means that the web server running eZ Publish has proper access to the Paynet server. If something goes wrong or if no response is produced, a firewall somewhere between the web server running eZ Publish and the Paynet server is most likely blocking the traffic. This or these firewalls will need to be reconfigured. If you do not know how to do this or do not have access, you will have to consult either the local system administrator or the internet service provider.

CURL support for PHP

The ”Paynet Direct” extension uses the ”Client URL Library” (CURL) functions, which provide solutions for communication over a range of protocols (in particular HTTPS). This means that the CURL extension for PHP must be installed and enabled. Furthermore, the PHP version should be at least 4.3.2 if not higher. The installation instructions are available within the documentation page of CURL. The following text shows two methods that can be used to check if CURL is installed or not.

Checking CURL support using eZ Publish

1. Log into the administration interface of eZ Publish.
2. Click on the ”Setup” tab.

3. Select ”System information” from the menu on the left
4. Check if ”CURL” is displayed under the ”Extensions” section for PHP.

Checking CURL support using a PHP file

1. Create a PHP file (for example ”test.php”) and make sure it contains the following:

```
<?php phpinfo(); ?>
```

2. Access the script (for example ”http://www.example.com/test.php”) from within a browser.
3. The output should reveal whether CURL is installed or not (just search for ”curl” on the page).

SSL support

The CURL extension for PHP (mentioned above) will only encrypt communication between the web server running eZ Publish and the Paynet server. However, communication between the customers and the web server running eZ Publish should also be encrypted. Otherwise, the solution will be open and thus extremely insecure. Having SSL support on the web server running eZ Publish is highly recommend because it will encrypt the communication between the customers and eZ Publish. This means that it should be possible to access an eZ Publish page using a secure HTTP (HTTPS) connection. Please refer to the documentation of Apache SSL for information about installation, configuration, etc. The following illustration shows an overview of the visitor, the web server running eZ Publish and the Paynet server.

(see figure 1.1)

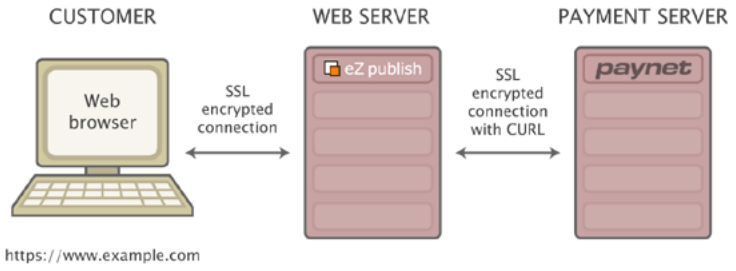


Figure 1.1: *Payment traffic and encryption*

Testing the SSL mechanism

To test whether SSL is working or not, attempt to access an eZ Publish page using the HTTPS protocol. This can be done by instructing a browser to access an URL that is similar to the following:

```
https://www.example.com/content/view/full/2
```

If everything works, the expected eZ Publish generated page should be visible in the browser.

Registering a Paynet merchant account

The Paynet service can only be used by authorized/registered Paynet merchants. In other words, you will have to sign up and become Paynet merchant. More information about becoming a Paynet merchant can be found at the following address: http://www.paynet.no/index.php/paynet/sign_up/merchant_application.html. When setting up the eZ Publish Paynet extension, make sure that you have the following information available:

- Your Paynet merchant number (also known as the agreement number)
- Your Paynet key-certificate

Please note that the Paynet extension can be tested using a special demo key-certificate. In other words, you do not need an actual Paynet account to test the solution. Please refer to the ”Testing” (page 15) section for more information about testing and the demo key-certificate.

Chapter 2

Installation

The requirements for using the ”Paynet Direct” extension must be met. Please refer to the previous section if you are not sure about the requirements. The extension must be extracted and activated. Please refer to the ”Extensions” section of the ”Installation” chapter within the technical manual of eZ Publish for more information about these operations. This section will guide you through the following steps:

1. Initializing the database
2. Adding a workflow and setting up a trigger
3. Configuring the extension

Initializing the database

The ”Paynet direct” extension makes use of two custom database tables:

- ezcreditcard_info
- ezpaynetdirect_transaction

These tables must be added using the SQL scripts that come with the extension; they are located within the ”sql” subdirectory of the extension (for example: ”/path/to/ez_publish/extensions/ezpaynetdirect/sql”). Please refer to the instructions above for creating the tables on a MySQL or a PostgreSQL database.

MySQL database

1. Bring up a system command line / shell.
2. Navigate into the ”sql” directory of the ”Paynet Direct” extension.

3. Run the ”create-tables.sql” script:

```
$ mysql -u <username> -p<password> <database> < create-tables.sql
```

username	The MySQL user (if no user is set up, use ”root”).
password	The password that belongs to the username.
database	The name of the database, for example ”my_database”.

PostgreSQL database

1. Bring up a system command line / shell.
2. Navigate into the ”sql” directory of the ”Paynet Direct” extension.
3. Run the ”create-tables.psql” script:

```
$ psql <database> <username> < create-tables.psql
```

username	The PostgreSQL user.
database	The name of the database, for example ”my_database”.

Adding a workflow and setting up a trigger

The ”Paynet Direct” extension integrates into eZ Publish through a workflow event. This means that a new workflow must be created and assigned to a trigger. The following text explains how this can be done.

Adding the workflow

1. Log into the administration interface.
2. Click on the ”Setup” tab.
3. Click on ”Workflows” in the menu on the left.
4. Navigate into one of the existing workflow groups or create a new group.
5. Create a new workflow.
6. Enter a suitable name, for example ”Paynet Direct Workflow”.
7. Select the ”Paynet / Direct Payment” event using the dropdown menu.
8. Click the ”Add event” button.
9. Click ”OK”.

Adding the trigger

1. Log into the administration interface.
2. Click on the ”Setup” tab.
3. Click on ”Triggers” in the menu on the left.
4. Select the previously created workflow (”Paynet Direct Workflow”) for the ”shop/checkout/before” trigger.
5. Click ”Apply changes”.

Modifying the certificate

From paynet you got one zip file with several certificates. Only the file that ends with: ”keycert.pem” should be copied in the settings directory of the PaynetDirect extension. Hereafter rename this file so that it ends with php (For example: MySite-keycert.pem.php).

The next step is to add the php-opening, php-closing, and comment tags to this certificate file. Open the certificate with your favorite editor and
add the following two lines to the beginning of the file:

```
<?php
/*
```

At the end of the file add the following lines:

```
*/
?>
```

So the complete certificate should look like:

```
<?php
/*
-----BEGIN RSA PRIVATE KEY-----
MIICXQIBAAKBgQCkkaPKumTOGkYDYd1YnJKmsEINbkSfr/aSqBm4U0mS3W8DDtHn
...
< snip >
...
dICl058wh1492fFR36aW9f2Zx4lYD8y+V05RIUQcRMzb
-----END RSA PRIVATE KEY-----
-----BEGIN CERTIFICATE-----
MIICVjCCAb8CARUwDQYJKoZIhvcNAQEEBQAwdTELMAkGA1UEBhMCTk8xDTALBgNV
...
< snip >
...
```

```
lHoA0QkxCwUJ0ApaH6k5sP7OSF5XPw2qNCQ=
-----END CERTIFICATE-----
*/
?>
```

Chapter 3

Configuration

The ”Paynet Direct” solution can be configured using the ”paynetdirect.ini” configuration file which is located in the ”settings” subdirectory of the extension (for example: ”/path/to/ez_publish/extension/ezpaynetdirect/settings/paynetdirect.ini”). The following text reveals the different configuration settings and options.

Configuration settings

Setting	Description
MerchantID= *number*	The ”MerchantID” must be a valid Paynet merchant ID number. For more information about becoming a Paynet merchant, please refer to the following address: http://www.paynet.no/index.php/paynet/sign_up/merchant_application.html. For testing with the test certificate any number can be used as MerchantID.
SSL= *enabled*|*disabled*	This setting controls whether the extension should redirect the user to an SSL secured version of the page that is used for entering the credit card number, etc. This setting should always be enabled because of security issues. However, it may be disabled for testing purposes. When enabled, the server specified by the ”SecureHost” setting will be used.
SecureHost= *string*	This setting is only available in version 1.0. Use this setting when the ”SSL” directive is set to ”enabled”. The extension will use the specified host when redirecting to an SSL secured page. For example, if eZ publish is running on a webserver called ”www.example.com”, then the ”SecureHost” should be set to ”https://

	www.example.com". Redirecting to another hostname will result in a session loss. Therefore this setting is quite useless, and is removed from version 1.1 and up.
SSLCertFile= *string*	The value of this setting must be a path to the SSL certificate file that should be used during encrypted connections between eZ publish and the Paynet server. For testing with fake transactions you can use the test certificate. To use the test certificate, set this value to: extension/ezpaynetdirect/settings/paynet-demo-keycert.pem.php
StoreCreditCardInfo= *enabled*|*disabled*	This setting controls whether eZ publish should let the customers store their creditcard information for later use. When enabled, the system will provide a smoother experience for frequent shoppers. If users can shop anonymously then set this feature to disabled! Otherwise people can see the creditcard information from a previous anonymous user.
FakeError= *enabled*|*disabled*	This option can be used to generate fake error messages. When set to "enabled", the Paynet server will return a random error whenever the test credit card is accepted.

Please note that the solution should always be re-tested whenever the configuration is changed. Please refer to the "Testing" (page 15) section for more information about testing.

Example configuration file

The following example shows an example configuration for the "Paynet Direct" extension.

```
[PaynetDirectSettings]
MerchantID=12345
SSL=enabled
SecureHost=https://www.example.com
SSLCertFile=extension/ezpaynetdirect/settings/paynet-demo-keycert.pem.php
StoreCreditCardInfo=enabled
FakeError=disabled
```

Chapter 4

Testing

This section explains how the ”Paynet Direct” extension can be tested in order to verify that it works properly. Tests can be carried out using the paynet demo key certificate. It is located in the ”settings” directory of the extension (it is ”called paynet-demo-keycert.pem.php”). The default configuration makes use of this file. The MerchantID can be set to any number.

Credit card testing

1. Visit the eZ Publish site like a regular customer would do.
2. Put some items in the basket and execute the checkout process.
3. When promted to enter credit card information, use the numbers from the table below.

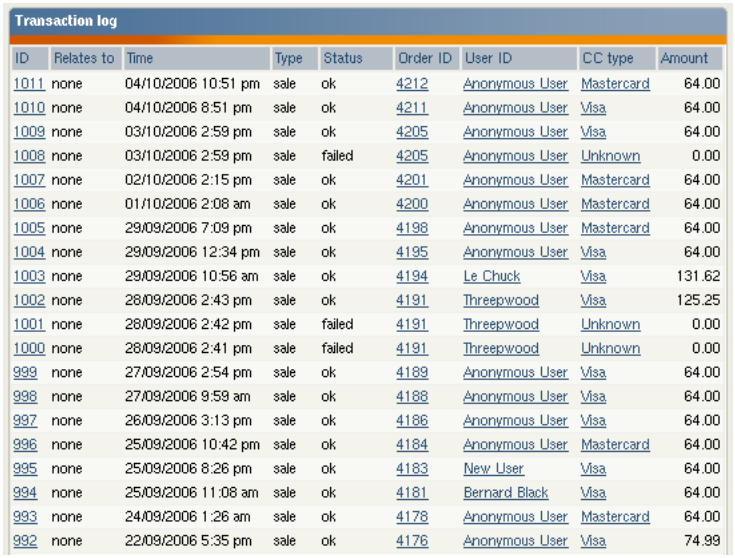
Credit card number	Description	Expiry date
Random number.	Invalid card number.	Any.
4444333322221111	Valid VISA test.	Any.
5555555555554444	Valid Mastercard test.	Any.

The VISA and Mastercard numbers should validate and thus allow the contents of the basket to be purchase. Using a random number should not process and will produce an error message. It is a good idea to test the system with several users and multiple scenarios; more testing usually equals a better / safer environment.

The transaction log

The ”Paynet Direct” extension provides a view that can be used to generate an overview of all transactions. This view can only be accessed by users with sufficient permissions (for example the ”Administrator User”). The following screenshot shows an example of the output that will be generated when the transaction log is requested.

(see figure 4.1)



Transaction log

ID	Relates to	Time	Type	Status	Order ID	User ID	CC type	Amount
1011	none	04/10/2006 10:51 pm	sale	ok	4212	Anonymous User	Mastercard	64.00
1010	none	04/10/2006 8:51 pm	sale	ok	4211	Anonymous User	Visa	64.00
1009	none	03/10/2006 2:59 pm	sale	ok	4205	Anonymous User	Visa	64.00
1008	none	03/10/2006 2:59 pm	sale	failed	4205	Anonymous User	Unknown	0.00
1007	none	02/10/2006 2:15 pm	sale	ok	4201	Anonymous User	Mastercard	64.00
1006	none	01/10/2006 2:08 am	sale	ok	4200	Anonymous User	Mastercard	64.00
1005	none	29/09/2006 7:09 pm	sale	ok	4198	Anonymous User	Mastercard	64.00
1004	none	29/09/2006 12:34 pm	sale	ok	4195	Anonymous User	Visa	64.00
1003	none	29/09/2006 10:56 am	sale	ok	4194	Le Chuck	Visa	131.62
1002	none	28/09/2006 2:43 pm	sale	ok	4191	Threepwood	Visa	125.25
1001	none	28/09/2006 2:42 pm	sale	failed	4191	Threepwood	Unknown	0.00
1000	none	28/09/2006 2:41 pm	sale	failed	4191	Threepwood	Unknown	0.00
999	none	27/09/2006 2:54 pm	sale	ok	4189	Anonymous User	Visa	64.00
998	none	27/09/2006 9:59 am	sale	ok	4188	Anonymous User	Visa	64.00
997	none	26/09/2006 3:13 pm	sale	ok	4186	Anonymous User	Visa	64.00
996	none	25/09/2006 10:42 pm	sale	ok	4184	Anonymous User	Mastercard	64.00
995	none	25/09/2006 8:26 pm	sale	ok	4183	New User	Visa	64.00
994	none	25/09/2006 11:08 am	sale	ok	4181	Bernard Black	Visa	64.00
993	none	24/09/2006 1:26 am	sale	ok	4178	Anonymous User	Mastercard	64.00
992	none	22/09/2006 5:35 pm	sale	ok	4176	Anonymous User	Visa	74.99

Figure 4.1: *Transaction log*

The transaction log is probably the best view to test or debug the paynet orders. The latest order approvals from the Paynet server appear on the top of the list.

The list has the following columns:

Name	**Description**
ID	The ID of the transaction. Click on the link to see the transaction details. Most of the information is already available in the transaction log, but it also includes the raw data sent from the paynet server.
Relates to	The relation to another entry in the log. For example a refund from a previous order (the customer paid too much). The refund will relate to the first transaction that bought the product.
Time	The time the order was processed.
Type	The type of the transaction. This will be either ”sale” or ”refund”. More information can be revealed when this field is combined with the ”status” field (see below).
Status	Possible values are ”ok”, ”failed” and ”rolled back”. When combined with the ”type” field, the following information is revealed: • ”sale”, ”ok”: The money was transferred from the customer to the retailer. • ”sale”, ”failed”: Something went wrong

	with the transaction. Click on the ID link to bring up more information. • ”refund”, ”ok”: The money was successfully refunded (from retailer to customer). • ”refund”, ”failed”: Something went wrong with the refund. Click on the ID link to bring up more information. • ”sale”, ”rolled back”: A transaction that first was ”sale” ”ok” but later was refunded will have it’s type and status set to ”sale” ”rolled back”.
Order ID	The ID number of the order (as a link). The link brings up a view that shows the products/ services that the customer bought.
User ID	The ID number of the user that represents the customer (as a link). The link brings up a view that shows a complete overview of all orders that the user has placed. Please note that this list might be very long for the Anonymous user.
CC Type	The type of the credit card that was used (as a link). The link brings up the transaction view which can also be accessed from the Paynet oderview (both views will be discussed later).
Amount	The amount of transferred money.

The transaction log shows a detailed log of every transaction that has been made. It is useful when something goes wrong and needs to be checked. When everything works, it is probably easier to use the Paynet direct orders.

The order view

This view shows a list of payed orders. New orders will appear on the top the of the list. Note that only the orders that have been payed end up in this list. The following screenshot shows an example of the output that will be generated when this view is requested.

(see figure 4.2)

The columns of this list are:

Name	**Description**
ID	The ID number of the order (as a link). The link brings up more information about the order itself (works in the same way as the ID link

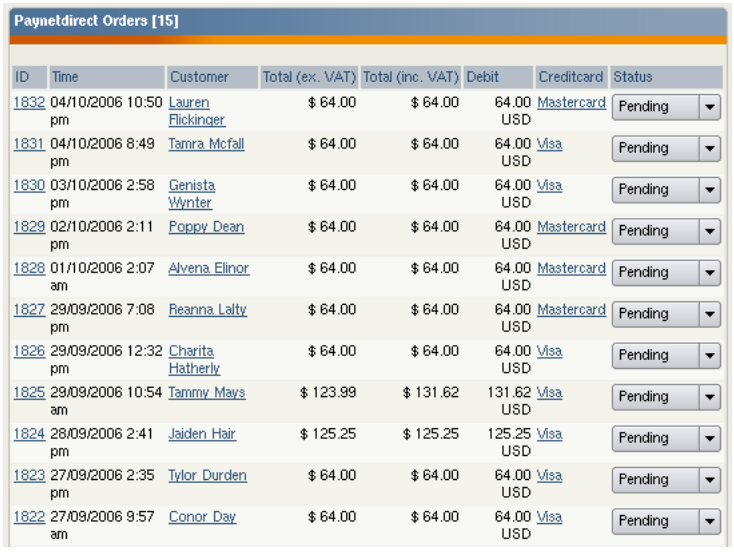


Figure 4.2: *Paynetdirect orders*

	in the transaction log).
Time	The time when the order was placed.
Customer	The name of the customer who placed the order. Note that this is not the user name.
Total (ex VAT)	The total price not including VAT.
Total (inc VAT)	The total price including the VAT.
Debit	The amount of money that was transferred from the customer to the retailer - both should be ”happy” when the debit equals the Total (inc VAT) value.
Creditcard	The type of creditcard that was used (as a link). The link brings up the transaction view.
Status	The status of the order. The shop administrators can (should) change these to keep track of the orders.

To get a more detailed view on the money transfers click on the creditcard link.

The transaction view

The transaction view shows the transactions done for a specific order. The following screenshot shows an example of the output that will be generated when the transaction view is requested.

(see figure 4.3)

Usually one transaction belongs to an order. The transaction view shows in that case only one ”sale” action. Multiple transactions belonging to one order is also possible. The retailer can transfer money back and forth to and from the customer by clicking, respectively, the ”Refund

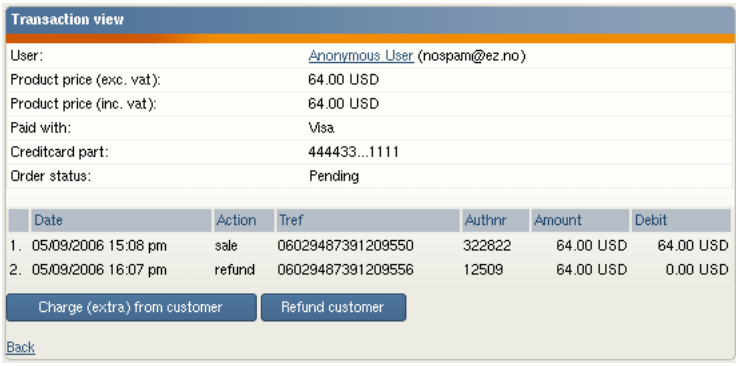


Figure 4.3: *The transaction view*

customer" and "Charge (extra) from customer" buttons. The retailer chooses the amount to refund or charge and presses "OK". A new transaction appears in the transaction view. Notice the change in the debit amount.

Refunding is useful when a customer paid too much. Some examples are: the customer gets a discount or the customer chose the wrong country and pays therefore too much VAT.

Charging extra can be useful when the retailer refunded too much or extra costs can only be determined later (e.g. shipping costs).

Most of the fields are obvious except for the Tref and Authnr fields. These values come from Paynet and may be useful when the expertise from Paynet is required.

Permissions

Different customers should not be able to see eachother's orders. A customer should only have access to the "buy" function of the "shop" module. If it is possible to tamper with the URL to see other people's orders, then something is wrong. It is highly recommended to make sure that this is not the case; the following procedure can be used.

1. Log in as user "A" and put some items in the shopping basket.
2. Do a complete checkout and pay using one of the test credit card numbers.
3. When the order is shown, copy the URL from the address bar of the browser.
4. Log out user "A".
5. Log in as user "B".
6. Attempt to access the URL that was copied in step 3.

The system should respond with an "Access denied" message. However, if user "B" is allowed to see the order of user "A", then the permission settings are wrong.

Testing and monitoring failed transactions

The ”FakeError” setting in the ”paynetdirect.ini” configuration file makes it possible to generate fake error messages. When set to ”enabled”, the Paynet server will return a random error whenever one of the test credit cards is accepted. In other words, this makes it possible to simulate a scenario where the actual transaction fails. Failed transactions should be clearly visible in the transaction log (see above). It is recommended to verify that failed transactions are properly logged before going public with the site. The ”FakeError” setting should only be used for testing purposes and must be set to ”disabled” when the site is released.