



# **IDEAL SHOP INTEGRATION GUIDE**

**PHP Merchant Plug-in**

**composed by:  
Friesland Bank N.V.**

version 0.6, September 19, 2005

© Copyright 2005, Friesland Bank N.V.



## Contents

---

<b>Contents .....</b>	<b>2</b>
<b>1 Introduction .....</b>	<b>3</b>
<b>2 Installation .....</b>	<b>4</b>
2.1 Prerequisites .....	4
2.2 Global configuration .....	4
2.3 Security configuration .....	5
2.4 Demoshop installation .....	6
<b>3 Transactions .....</b>	<b>7</b>
3.1 Directory request.....	7
3.2 Transaction request.....	8
3.3 Customer Redirect.....	9
3.4 Status request.....	9
3.5 Errors .....	10
<b>APPENDIX A: Datacatalogue .....</b>	<b>11</b>
<b>APPENDIX B: Troubleshooting .....</b>	<b>12</b>



## 1 Introduction

---

This document is intended to help software developers to integrate the iDEAL payment method into their PHP webshop(s). This document describes the iDEAL Merchant Plug-in (the so-called 'thinMPI') and offers installation instructions and some example shop code.

This document builds on the general introductory information. Readers not familiar with the general introductory information are advised first to absorb the information presented in the document 'iDEAL Shop Integration Guide – General Reference', so that detailed information about the iDEAL protocol and the integration process can be placed in the proper context.

## 2 Installation

---

This chapter describes all the steps necessary to install and configure the iDEAL Merchant Plug-in (thinMPI) and the example code.

### 2.1 Prerequisites

- PHP capable webserver like Apache or IIS
- PHP Version 4.3
- OpenSSL-module 0.9.7
- Internet access over port 443 (SSL)
- Public key of acquirer (included in thinMPI)
- MerchantID and SubID
- thinMPI files copied in subdirectory `thinMPI\` at your webserver
- Access permissions (at least reading permissions) for thinMPI files

To install PHP and the OpenSSL-module for PHP see also <http://www.php.net> and <http://www.openssl.org>.

If you compile PHP yourself, the SSL library must be included. Please verify using `phpinfo()`.

### 2.2 Global configuration

The parameters below needs to be set in the configuration file `thinMPI\config.conf` (see APPENDIX A: Datacatalogue for a description of these parameters).

- `merchantID`
- `subID`
- `authenticationType`
- `merchantReturnURL`
- `currency`
- `expirationPeriod`
- `language`
- `description`
- `entranceCode`

Security related entries are explained in 2.3 Security configuration.

## 2.3 Security configuration

In order to create a private/public key pair that can be used by the thinMPI, the following steps have to be completed:

1. Download the openssl library to your computer (<http://www.openssl.org>). There are a number of binaries for different operating systems available.
2. Generate a RSA private key:

```
openssl genrsa -des3 -out priv.pem -passout pass:myPassword 1024
```

3. Create a new Certificate based on this private key:

```
openssl req -x509 -new -key priv.pem -passin pass:myPassword -days 365 -out cert.crt
```

4. Copy the private key and the certificate file into the directory `thinMPI\security\`
5. Edit the following entries in the configuration file `thinMPI\config.conf`, using the data from the steps 2 and 3 above (see also APPENDIX A: Datacatalogue):
  - `privateKey`
  - `privateKeyPass`
  - `cert`

The public certificate entry `certificate0` does not need to be modified. This entry holds the public certificate of the acquirer for message authentication and is included in the thinMPI by default.

If a new public certificate is received from the acquirer, a new entry needs to be added for this (and every next) certificate. To add a new acquirer public certificate, edit the the configuration file `thinMPI\config.conf` and add the certificate as follows:

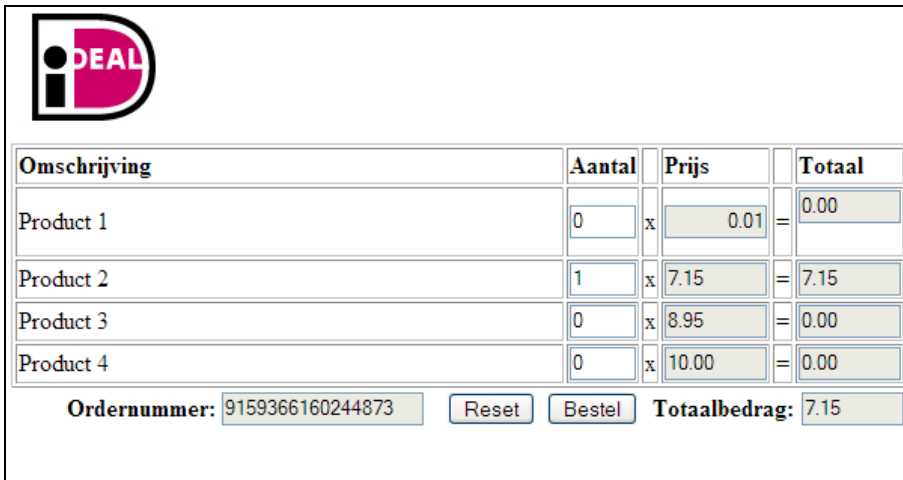
```
certificate0=<filename of default certificate>  
certificate1=<filename of new certificate>  
etc.
```

## 2.4 Demoshop installation

The example shop consists of three files that have to be copied into the directory `thinMPI\`:

- `DirReq.php`
- `TransReq.php`
- `StatReq.php`.

Open the file `DirReq.php` in a browser. You now see a simple shopping system. You can enter quantities and prices for four different products.



The screenshot shows a web interface for an example shop. At the top left is the iDEAL logo. Below it is a table with four columns: 'Omschrijving', 'Aantal', 'Prijs', and 'Totaal'. The table contains four rows for 'Product 1' through 'Product 4'. Each row has input fields for quantity and price, and a calculated total. Below the table, there is an 'Ordernummer' field with the value '9159366160244873', a 'Reset' button, a 'Bestel' button, and a 'Totaalbedrag' field with the value '7.15'.

Omschrijving	Aantal	Prijs	Totaal
Product 1	<input type="text" value="0"/>	<input type="text" value="0.01"/>	<input type="text" value="0.00"/>
Product 2	<input type="text" value="1"/>	<input type="text" value="7.15"/>	<input type="text" value="7.15"/>
Product 3	<input type="text" value="0"/>	<input type="text" value="8.95"/>	<input type="text" value="0.00"/>
Product 4	<input type="text" value="0"/>	<input type="text" value="10.00"/>	<input type="text" value="0.00"/>

Ordernummer:    Totaalbedrag:

Figure 1 Example shop

## 3 Transactions

---

This chapter describes the three iDEAL requests:

- Directory Protocol
- Transaction Protocol
- Status Protocol

This chapter uses the example code to explain these three protocols. Each request is implemented in three separate example files: `DirReq.php`, `TransReq.php` and `StatReq.php` respectively.

### 3.1 Directory request

The file `DirReq.php` shows a (very simple) webshop in which the user can select some products.

By opening the file `DirReq.php` a Directory Request is automatically processed. This means that the most up-to-date list of iDEAL Issuers is requested from the acquirer. The selectbox below the orderform is populated with the results of the Directory Request.

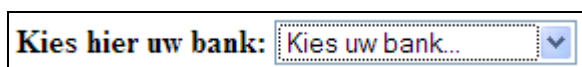


Figure 2 Selectbox populated after Directory Request

The following is done in `DirReq.php` to perform a Directory Request.

1. Create a `DirectoryRequest` object:

```
$data = & new DirectoryRequest();
```

2. Set parameters. Relevant parameters for the Directory Request are `merchantID`, `subID` and `authenticationType` (see APPENDIX A: Datacatalogue). These parameters are already set in the configuration file `thinMPI\config.conf`.

3. Create an instance of `thinMPI`:

```
$rule = new ThinMPI();
```



#### 4. Process the request using the thinMPI:

```
$result = $rule->ProcessRequest($data);
```

Check the file `DirReq.php` to see how the result is handled to populate the selectbox.

Please note that the demoshop performs a Directory Request every time a user accesses `DirReq.php`. This is not mandatory, because the list of iDEAL issuers does not change very often. It is also allowed to perform the directory request periodically and store the list on your system. In 'iDEAL Shop Integration – General Reference' is described what errors may occur when the issuer list is not up-to-date.

After the user clicks the Order-button the form is posted to the file `TransReq.php`.

### 3.2 Transaction request

The file `TransReq.php` performs a Transaction Request based on the values it receives from `DirReq.php`. The process of performing a Transaction Request is very similar to performing a Directory Request:

1. Create Request object (`AcquirerTrxRequest` in this case)
2. Set parameters. Relevant parameters for the Transaction Request are set in the configuration file `thinMPI\config.conf`. The parameters `issuerID`, `purchaseID` and `amount*`, are obtained from the Directory Request.
3. Create an instance of `thinMPI`
4. Process the request using the `thinMPI`
5. Check and handle the result

\* Please note that on the integration environment the parameter `amount` determines the result of the transaction. Use `amount=100` to simulate a successful transaction. See 'iDEAL Shop Integration – General Reference' for a description of the 7 (mandatory) test transactions.



### 3.3 Customer Redirect

After the Transaction Request the customer is redirected to the issuer. The URL of the issuer is provided in the response of the Transaction Request. It can be retrieved from the result of

`ProcessRequest()` with the function `getIssuerAuthenticationURL()`:

```
//Get IssuerURL en decode it

$ISSURL = $result->getIssuerAuthenticationURL();

$ISSURL = html_entity_decode($ISSURL);

//Redirect the browser to the issuer URL

header("Location: $ISSURL");

exit();
```

### 3.4 Status request

After the transaction the customer is redirected back to the `merchantReturnURL`. To perform the mandatory (!) Status Request the steps below have to be completed. Read everything about the so-called 'collection duty' of the status of every iDEAL transaction in 'iDEAL Shop Integration – General Reference'.

1. Create Request object (`AcquirerStatusRequest` in this case)
2. Set parameters. Relevant parameters for the Status Request are set in the configuration file `thinMPI\config.conf`. The parameter `transactionIDssuerID` is obtained from the Transaction Request.
3. Create `thinMPI` instance
4. Process the request using the `thinMPI`
5. Check and handle the result

### 3.5 Errors

When an Error Response is returned instead of a normal response this is handled by the thinMPI. By calling the function `isOK()` on the result of `ProcessRequest()` you can check if an `ErrorRes` is received. If so, the function `getErrorMessage()` returns a description of the error. For example (from `DirReq.php`):

```
if(!$result->isOK())
{
    print("Er is op dit moment geen betaling met iDEAL mogelijk.<br>");
    print("Foutmelding van iDEAL: ");
    $Msg = $result->getErrorMessage();
    print("$Msg<br>");
}
```

See 'iDEAL Shop Integration – General Reference' for an overview of all possible errors.

## APPENDIX A: Datacatalogue

Parameter	Description
IssuerID	ID of the issuer that the user has selected
merchantID	ID that identifies the merchant (you)
subID	Set to "0" unless you have specific reasons not to
authenticationType	Set to SHA1_RSA
merchantReturnURL	URL on the merchants system that the customer is redirected to after the payment. This page should perform the status request
purchaseID	Ordernumber from the merchant's system (your system)
amount *	Total amount for the order  <b>N.B.</b> Please note that on the test environment the parameter amount determines the result of the transaction. See 'iDEAL Shop Integration – General Reference' for an overview of the 7 mandatory test transactions.
currency	Set to EUR
expirationPeriod	Timeframe during which the transaction is allowed to take place. Notation PnYnMnDTnHnMnS, where every n indicates the number of years, months, days, hours, minutes and seconds respectively. E.g. PT1H indicates an expiration period of 1 hour. PT3M30S indicates a period of 3 and a half minutes. Maximum allowed is PT1H; minimum allowed is PT1M.
language	nl for dutch, en for english (used for showing errormessages is the preferred language)
description	Description of the order
entranceCode	Mandatory code to identify the customer when he/she is redirected back to the merchantReturnURL
acquirerURL	The URL where the Directory, Transaction and Status Requests have to be send to.  Integration environment: <a href="https://testidealkassa.frieslandbank.nl/ideal/iDeal">https://testidealkassa.frieslandbank.nl/ideal/iDeal</a>  Production environment: <a href="https://idealkassa.frieslandbank.nl/ideal/iDeal">https://idealkassa.frieslandbank.nl/ideal/iDeal</a>
privateKey	RSA private key generated by merchant (you)
privateKeyPass	Password used to generate Certificate based on privateKey
cert	Certificate generated with privateKeyPass and privateKey
certificate0	Default public certificate from acquirer (included with thinMPI)



## APPENDIX B: Troubleshooting

---

### Invalid signature or Authentication error

Make sure you uploaded the correct certificate via the iDEAL Dashboard and that the correct security parameters are defined in the configuration file `thinMPI\config.conf`.

### MerchantID unknown

Make sure you use either the demoshop `merchantID` (which was originally in the code) or your own `merchantID`. If the error occurs when using your own `merchantID`, your application is most probably not verified by the iDEAL Servicedesk.