

Hyves Afrekenen Implementation Manual

Version October 05, 2010 (haim_20101005.pdf)

Contents

Version management	3
Introduction	4
What is MiniTix and what is <i>Hyves Afrekenen</i> ?	4
Actors in the <i>Hyves Afrekenen</i> environment.....	4
Customer	4
Merchant	4
Hyves.....	4
MiniTix	4
MiniTix payment flow	5
Security	6
Cryptography	6
Keys 6	
OpenSSL	6
Integration of <i>Hyves Afrekenen</i> / MiniTix in the shop.....	7
MiniTix payment URL.....	7
<i>Hyves Afrekenen</i> payments buttons.....	7
Dialog 7	
Example	7
Message specification.....	8
MPS_PAYREQ.....	8
MPS_TICKET	9
MPS_ERROR.....	9
Syntax error [010]	10
Integrity error [020]	10
Merchant unknown [030]	10
Merchant disabled [031]	10
Payment cancelled [040]	10
Request outside time window [080].....	10
System error [090]	10
Unauthorized user [100].....	11
Payment confirmation started [110]	11
Insufficient balance [120]	11

Data types	11
String	11
Number	11
Timestamp	11
Signing messages	12
Examples	12

Version management

Version	Date	Description	Author
20100402	04-02-2010	<i>Hyves Afrekenen</i> impl. Man.	Rabobank / Hyves
20100504	05-04-2010	Various small changes	Rabobank / Hyves
20101005	10-05-2010	Added forbidden characters, updated links	Hyves

Introduction

This document is intended for developers who integrate *Hyves Afrekenen* with MiniTix in a website or application.

This document is best used together with the implementation examples.

What is MiniTix and what is *Hyves Afrekenen*?

MiniTix is an advanced payment transaction system. It can be used by customers of Dutch banks.

Hyves Afrekenen is an implementation of the MiniTix system for the Hyves community. Another example of a community implementation of MiniTix is MyOrder. Registered members of these communities can initiate payment transactions through MiniTix. With *Hyves Afrekenen*, customers are using their mobile number and community ID for creating and using their MiniTix account. *Hyves Afrekenen* uses SMS text messaging to confirm payments.

Actors in the *Hyves Afrekenen* environment

Customer

The customer is the person browsing a Web Shop, or any another channel, who has access to a *Hyves Afrekenen* with MiniTix wallet; i.e. he/she registered a wallet as a member of the Hyves community.

Merchant

The merchant is the party that has a merchant agreement with MiniTix and offers *Hyves Afrekenen* payments as an option to customers.

The responsibility of the merchant is to supply the correct payment information in a call to *Hyves Afrekenen*, and to accept and process transaction response tickets from *Hyves Afrekenen*.

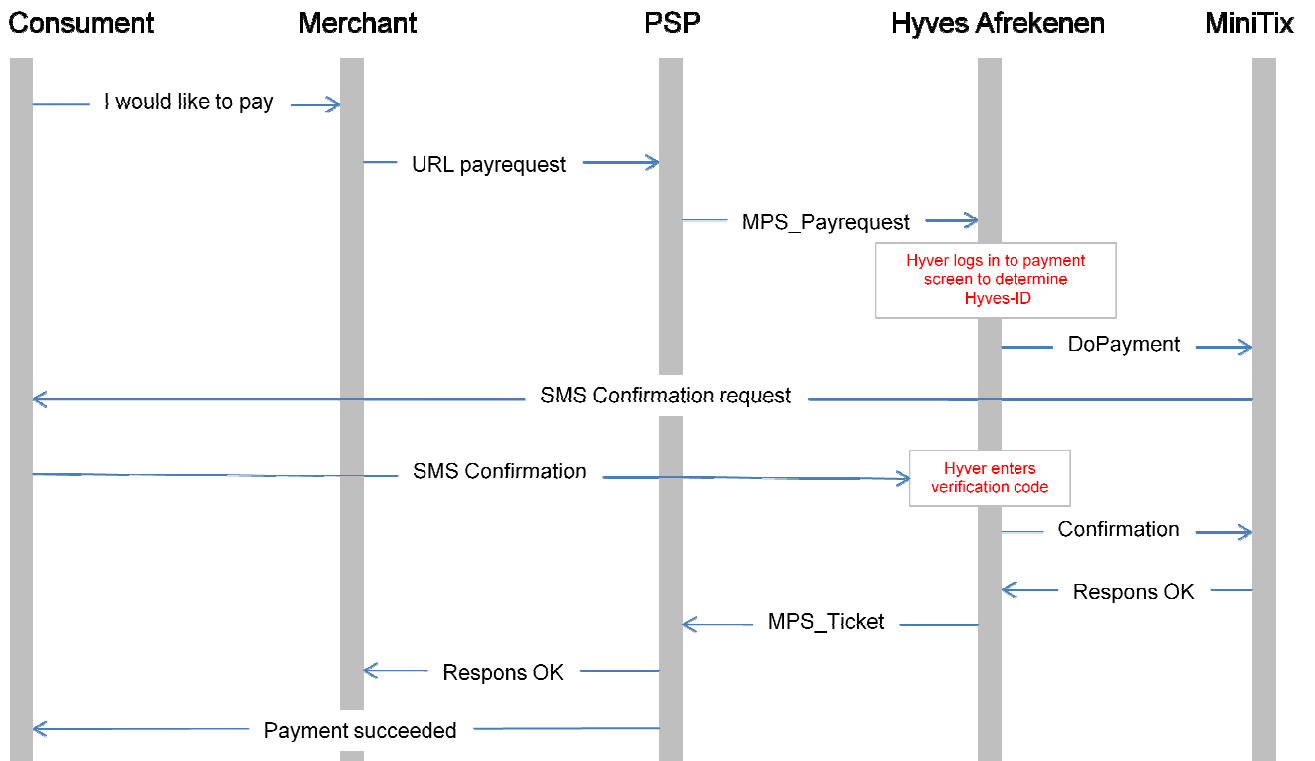
Hyves

The community that delivers the user interface to the merchant's customer and handles all *Hyves Afrekenen* MiniTix requests and responses. Hyves accepts payment requests by merchants, has them processed by MiniTix and communicates the results.

MiniTix

The MiniTix system is responsible for the transactions that start when *Hyves Afrekenen* sends payment requests to the MiniTix system. This responsibility includes recording transactions, passing confirmation tickets to Hyves and taking care of the actual financial transactions.

MiniTix payment flow



Security

Hyves Afrekenen security is provided by MiniTix. MiniTix uses certificates and cryptography to make sure the different actors receive authentic messages from each other.

Cryptography

MiniTix uses Public-key cryptography on payment requests and responses.

The cryptography is not used for encrypting or decrypting the messages. It is used for creating and confirming digital signatures on the messages.

Check out http://en.wikipedia.org/wiki/Public-key_cryptography for more information concerning public key cryptography.

Keys

Test certificates and a test key store will be given will be provided by email after you registered at MiniTix. This can be used so the merchant can test the MiniTix payment flow in their shop.

The merchant will be given a set of public key certificates by MiniTix for production use. These certificates can be used to verify incoming responses.

The merchant will also have to create an own key store for production use. With the keystore they need to create a production use public key certificate. They will send a Public Key certificate to MiniTix. The next paragraph will explain how this key can be created.

OpenSSL

OpenSSL (<http://www.openssl.org>) can be used to create and maintain MiniTix certificates.

To create the public/private certificates and to create a key store, which can be used in for example Java and Microsoft.NET, you need to take the following steps (please make sure you use your own passphrase instead of *secret*):

```
opensslgenrsa -des3 -outform PEM -out priv_key.pem -passout pass:secret 1024
```

This command creates a unique RSA private key.

```
openssl req -x509 -new -key priv_key.pem -passin pass:secret -days 365 -out certificate.pem
```

This command creates a public X509-certificate using the generated key.

```
openssl pkcs12 -export -in certificate.pem -inkey priv_key.pem -out keystore.p12 -passin pass:secret -name "keystore"
```

This command imports the private key certificate in to a key store that can be used in the web shop.

Integration of *Hyves Afrekenen* / MiniTix in the shop

After registration at MiniTix you will receive an email with instructions concerning your certificates.

There will follow some examples of how *Hyves Afrekenen* can be implemented in a web shop soon.

MiniTix payment URL

Payment URL's are provided by the community. For *Hyves Afrekenen* the payment URL is:

<https://secure.hyves.org/miniTix/ticket>

Hyves Afrekenen payments buttons

Payment buttons for *Hyves Afrekenen* are available on: <http://hyv.es/dgPk7q> . We strongly advise you to use the static url's which are provided.

Dialog

When the user presses the button the page should open a new (pop-up) browser screen. The browser screen should be opened with location bar and status bar opened. This is because the URL and SSL key needs to be shown to the user.

Example

Hyves Afrekenen

```
<a
onclick="javascript:window.open('https://secure.hyves.org/miniTix/ticket?[Message]','miniTix',
'width=850, height=600, left=0, top=0, status=yes, location=yes, resizable=yes'); return false;"
href="">

</a>
```

Message specification

The text format of the content of the message should be ISO 8859-1.

MPS_PAYREQ

#	Logical Name	Req.	HTTP parameter	Format	Description
1	messageType	X	mtype	String	Value is MPS_PAYREQ
2	messageVersion	X	mver	Number	Value is 0011
3	messageTimestamp	X	mtime	Timestamp	
4	merchantId	X	vid	Number	This is the “Aansluitnummer” that the merchant gets when they subscribe for MiniTix.
5	merchantName	X	vname	String	This is the same name the merchant used during the MiniTix subscriptions.
6	merchantSite		vsite	String	This can be used for merchants who use different websites to differentiate.
7	orderId	X	ordid	Number	
8	amount	X	amount	Number	Number of cents
9	currencyCode	X	currency	String	Value is a 3 letter ISO 4217 code like EUR
10	orderDescription1		desc1	String	Maximum length is 20
11	orderDescription2		desc2	String	
12	orderDescription3		desc3	String	
13	validUntil	X	until	Timestamp	
14	customerIdType		cidtype	String	CID_MOBILENR
15	customerIdValue		cidvalue	String	
	mobileNrSuggestion			String	
16	returnURL	X	retURL	String	
17	errorURL		errURL	String	
	dataBlock		data	String	
18	certificateId	X	certid	String	
	signature	X		String	Base 64

Please note: “<” and “>” are forbidden characters in the orderDescription[1-3] fields.

The dataBlock field can be used to include merchant information. Like the number of products ordered or some description or id concerning the order. It will be passed back with the response messages.

MPS_TICKET

The MPS_Ticket is used to acknowledge to the merchant that the customer completed a certain payment. This ticket is added in the HTTP Header as well as the HTTP body to the returnUrl, so the customer browser sends the message to the web site. The digital signature in the ticket should be verified by the merchant.

#	Logical Name	Req.	HTTP parameter	Format	Description
1	messageType	X	mtype	String	Value is MPS_TICKET
2	messageVersion	X	mver	Number	Value is 0010
3	messageTimestamp	X	mtime	Date	
4	merchantId		vid	Number	
5	orderId		ordid	String	
6	amount		amount	Number	Number of cents
7	currencyCode		currency	String	Value is a 3 letter ISO 4217 code like EUR
8	stackId		sid	Number	
9	ticketId		tid	Number	
10	certificateId		certid	String	
11	dataBlock		data	String	Returns the data that the merchant added in the request
12	signature	X	sig	String	

MPS_ERROR

This message will be returned when a payment request goes wrong. Like the MPS_TICKET message the message will be returned in the HTTP Request driven by the customer's browser. It will be send to the returnUrl.

The merchant should make sure they handle these messages appropriately.

Please note that **no** digital signing will be added to error messages.

#	Logical Name	HTTP parameter	Format	Description
1	messageType	mtype	String	Value is MPS_ERROR
2	messageVersion	mver	Number	Value is 0010
3	messageTimestamp	mtime	Date	
4	merchantId	vid	Number	
5	orderId	ordid	Number	
6	errorCode	errcode	String	See list below
7	errorDescription	errdesc	String	Gives a human readable description of the error
	dataBlock	data	String	Returns the data that the merchant added in the request

Error codes:

Code	Description
010	Syntax error
020	Integrity error
030	Merchant not known
031	Merchant disabled
040	Payment cancelled
080	Request outside time window
090	System error
100	Unauthorized customer
110	Payment confirmation started
120	Insufficient balance

Syntax error [010]

One or more fields of the payment request message are not following the specification

The error description field will state which field(s) are not correct.

Integrity error [020]

The signature of the send message cannot be confirmed. The integrity of the message is breached.

This error can be caused by any of the following:

1. Signature is set different data then the current payment request contains;
2. The data in the request has been changed after the message got signed;
3. The wrong private key is used for signing the message, or there is a different public key on the Minitix system.

Merchant unknown [030]

The merchant specified in the payment request is unknown in the MiniTix system.

Merchant disabled [031]

MiniTix has blocked the merchant from the system, so there can be no payment request at the current time.

Payment cancelled [040]

The customer has cancelled the payment. This happened when the customer presses the cancel button in the authorization screen or when the customer is blocked from payments in MiniTix.

Privacy wise there will be no difference in cancellations and blocked users.

Request outside time window [080]

The payment request was delivered before the specified message timestamp. Or the payment request was delivered after the validUntil timestamp.

System error [090]

An error in the MiniTix system has occurred so the payment request could not be handled at the current time.

Unauthorized user [100]

The MiniTix customer id is not valid in the current supplier domain.

This can happen when a test customer is used for regular payments or if a regular customer wants to make a test payment.

Payment confirmation started [110]

The payment cannot be confirmed within the same HTTP request. The user has been given a payment confirmation screen to confirm the current payment. This will be u

Insufficient balance [120]

Balance of the customer is not sufficient.

Data types

String

Strings are free text in the ISO-8859-1 format.

This text will be encoded using HTTP encoding. The text cannot contain spaces so they will be replaced with a + sign. This encoding replaces other unsafe characters with "%" followed by two hexadecimal digits corresponding to the character values in the ISO-8859-1 character-set.

Number

Number only consist of numeric characters no others.

Timestamp

The timestamp is a GMT time following the ISO-8601 specification. It is limited to the following format:

YYYY-MM-DDThh:mm:ssZ

YYYY is the year, MM is the month, DD the day, hh the hour, mm the minutes and ss the seconds. The T means to delimit date and time from each other. The Z is post fixed to say that the time is Zulu/GMT time.

Signing messages

Create a message hash string:

1. Take an empty string X.
2. Step through the numbered fields specified in the message specification;
3. For every field take the length of the content of that field. If there is no value the length is 0;
4. If the number of characters of the length is lower than 4, prefix it with zero's till the number of character is 4. So 1 will be 0001 and 312 will be 0312.
5. Paste the length and the content itself to the string X. So for example field "MPS_PAYREQ" will be added as 0010MPS_PAYREQ to the string X.
6. Go to 2, while there are more numbered field.
7. Do an openssl_sign of string X:
"openssl_sign(X, signature, privateKey, OPENSSL_ALGO_SHA1)"
8. Encode the signature
"signature = base64_encode(signature)"

Examples

Payment request:

```
mtype=MPS_PAYREQ&
mver=0010&
mtime=2010-02-03T12%3A28%3A50Z&
vid=999&
vname=Test+Shop&
vsite=www.eentestshopurl.nl&
ordid=1010101010&
amount=1995&
currency=EUR&
desc1=een+test+aankoop&
desc2=&
desc3=&
until=2010-02-03T13%3A28%3A50Z&
returl=https%3A%2F%www.eenshopurl.nl%2Fpaymentready.jsp%3Fpayid%3testid&
errurl=https%3A%2F%www.eenshopurl.nl%2Fpaymenterror.jsp%3Fpayid%3testid&
certid=A1389EF7CE276B5182D1115D9F1E15747076F9B5&
data=&
sig=base64signature%3D
```

Payment ticket return:

```
mtype=MPS_TICKET&
mver=0010&
mtime=2010-02-03T12%3A28%3A50Z&
vid=999&
ordid=10101010&
amount=1995&
currency=EUR&
sid=KASSAROL0001&
tid=20&
certid=837D1D3069F5D8E29279AE0CFCBE20787B8F46B1&
data&
sig=base64signature&
pageid=54RRG3GNKA00WOSK8&
cmobsug
```

Error Return

In case of an error the return ticket will look like:

```
mtype=MPS_ERROR&
mver=0011&
ordid=10101010&
vid=999&
mtime=2010-02-03T12%3A28%3A50Z&
data&
errcode=090&
errdesc=Er+is+een+technisch+probleem+opgetreden&
cmobsug
```