API Manual

PayByMail

Version 4.55

Author: S.B. Bakker
Version: 4.55
Latest change: July 25th 2018

Contents

1. Introduction

Using the PayByMail API a payment can be created by an external system. The way this API is implemented, there's hardly need to change the supplied values to a PayByMail format because these values can be mapped to internal values within the PayByMail environment by the entity administrator.

1.1 Parameter mapping

As described, the PayByMail system uses an internal mapping system, to map input values to other input values that are mandatory for creating payments. There is no need to comply to the internal names of parameters because names, as well as values for these names, can be mapped as well using mapping rules.

Example; to create a payment, a currency is mandatory. Internally the name for this parameter is named: '*CurrencyCode'. We can choose to set this value in the PayByMail system to a fixed (and allowed) value, like 'EUR' but we can also choose to map a value that is supplied externally to the correct internal value.

In this example, we have input parameters called '**Currency**' and '**LanguageName**' with its' own values that must be mapped to an accepted *CurrencyCode and *LanguageCode value. The following mapping rules can apply:

#	Output name	Input condition	Output value
1	*CurrencyCode	[Currency] = 'Euros'	EUR
2	*CurrencyCode	[Currency] = 'Dollars' or [Currency] = 'Us dollars'	USD
3	*CurrencyCode		AUD
4	*LanguageCode	[LanguageName] = 'Dutch'	nl-NL
5	*LanguageCode		en-US
6	*ExpirationDate		[*AddHours([*DateNow], 4)]
7	UserName		[LoggedOnUser]

The system checks the rules from top to bottom and skips a rule if an output name already has been set before. When rule 1 is checked, the system asks for the input value for 'Currency' and checks this with the value 'Euros'. If these values are the same, the system sets the (internal) value '*CurrencyCode' to 'EUR'. If not, the system checks if 'Currency' contains the value 'Dollars' or 'Us dollars' and, if true, sets the '*CurrencyCode' to 'USD'. If this is also not the case, the '*CurrencyCode' value is set to 'AUD' so *CurrencyCode will always get a value.

Also values can contain functions. In rule 6 we set the ExpirationDate to four hours from now. Finally in rule 7, we just create our own parameter called 'UserName that is set by using the input value for 'LoggedOnUser' that is supplied. If a parameter is used as input in the mapping, and this parameter is not supplied, an error is thrown. In this example 'Currency', 'LanguageName' and 'LoggedOnUser' must be supplied as input parameters.

Only input parameters that were mapped, can be used within the PayByMail system. So in the mapping example '[LanguageName], '[Currency]' and '[LoggedOnUser]' cannot be used, but only the 'Output name' parameters they were mapped to internally. In the system these 'own' parameters can be used by mentioning them like [<name>]. For instance: '[UserName]' from the mapping example.

1.2 Mapping templates

Using this mapping tool, PayByMail can be used by any external system to create a payment hardly without any expensive development needed. When a connection is made to the PayByMail system, payments can be created by 'configuring' instead of 'developing'! This can be done by the entity administrator or by PayByMail employees.

Because multiple mapping templates can be created within the PayByMail system on entity or on a license level, the caller that uses the api has to identify what mapping must used to process the supplied information. This can be done by supplying a parameter with the name **MappingTemplate**. If this parameter is not supplied, or if it has no or an empty value, the default mapping will be used for that entity.

When a new mapping is created, all the mandatory output names will be created automatically. These values must be supplied but some of them can be left empty.

Parameters that can be used to create payments or in the mappings are described in chapter 3.

2. API connection

The connection to be PayByMail api is a secure connection that is protected by IP address an Extended Validation SSL certificate and a username and a password

To create a connection, the following things must be arranged:

Subject	Description
IP Address	The IP address(es) of the server that connects to the PayByMail system must be supplied to the PayByMail or Entity administrator. He can enter these addresses in the system so that connections that are created from other IP addresses will be refused.
Username Password	An API user must be created within the PayByMail system with its' own unique username and corresponding password. These values must be used to create a secure connection.

The url's to connect to the PayByMail system for **payments** are:
Test environment: **https://testpbmapi.icepay.com/payment/<command>**
Production environment: **https://pbmapi.icepay.com/payment/<command>**

The url's to connect to the PayByMail system for **mandates** are:
Test environment: **https://testpbmapi.icepay.com/mandate/<command>**
Production environment: **https://pbmapi.icepay.com/mandate/<command>**

The url's to connect to the PayByMail system for **direct collects** are:
Test environment: **https://testpbmapi.icepay.com/collect/<command>**
Production environment: **https://pbmapi.icepay.com/collect/<command>**

The <command> value can be replaced by any trusted command that are described next.

2.1 Create a payment via the API (http POST or http GET)

Using the '**create**' command a payment can be created for a specific entity using the supplied values.
The <command> value in the url must be replaced by: **create/<entity key>**
A secure connection, as described in the 'API Connection' chapter, must be created.

Url parameter	Description
entity key	Every PayByMail entity within the system has its' own unique key. When connecting to the PayByMail system, this key must be supplied as well so PayByMail knows within which entity the payment must be created.

Example: https://pbmapi.icepay.com/payment/create/e7140a88-cbba-4cc2-be9f-6e90536703ef

2.1.1 Posted values

Parameter name	Description
MappingTemplate	With this parameter you can let the API know what mapping template must be used. If left empty; the default payment mapping for that entity is used
*	You can post any value to the API. Only those values that are used in the mapping template will be used but they're all logged in the system

2.1.2 Return values

When the payment could be created successfully, the Payment Url is returned. This url can also be used by the calling application to send within its' own communication to the customer when 'SendType' = 'NoSend'.

If the payment could not be created successfully, a 'BadRequest' with a describing message is returned. All the calls and parameters to and from the API are logged within the PayByMail system.

2.2 Find the url of a payment by the order reference (http GET)

Using the '**url**' command a payment can be found for a specific order reference.
The <command> value in the url must be replaced by: **url/<entity key>/<order reference>**
A secure connection, as described in the 'API Connection' chapter, must be created.

Url parameter	Description
entity key	Every PayByMail entity within the system has its' own unique key. When connecting to the PayByMail system, this key must be supplied as well so PayByMail knows within which entity the payment must be created.
order reference	The unique reference of the order that must be searched

Example: https://pbmapi.icepay.com/payment/url/e7140a88-cbba-4cc2-be9f-6e90536703ef/2016REF-A

2.2.1 Return values

When the payment could be found successfully, the Payment Url is returned.
If the payment could not be found, a 'BadRequest' with a describing message is returned.

2.3 Change the status of a payment by the order reference (http POST)

Using the '**status**' command the status of a payment with the specified order reference can be changed.
The <command> value in the url must be replaced by: **status/<entity key>/<status>/<order reference>**
A secure connection, as described in the 'API Connection' chapter, must be created.

Url parameter	Description
entity key	Every PayByMail entity within the system has its' own unique key. When connecting to the PayByMail system, this key must be supplied as well so PayByMail knows within which entity the payment must be created.

status	The status that the payment must get. The following values are allowed;
	• LinkSent
	  ○ Only allowed from statuses: Booked, SendError
	• SendError
	  ○ Only allowed from statuses: Booked, LinkSent
	• MailDelivered
	  ○ Only allowed from statuses: Booked, LinkSent
	• MailOpened
	  ○ Only allowed from statuses: Booked, LinkSent, MailDelivered
	• LinkClicked
	  ○ Only allowed from statuses: Booked, LinkSent, MailDelivered, MailOpened
	• PaymentInProgress
	  ○ Allowed from all statuses except: Expired, Aborted, Success, Error
	• PaymentPending
	  ○ Allowed from all statuses except: Expired, Aborted, Success, Error
	• PaymentFailed
	  ○ Allowed from all statuses except: Expired, Aborted, Success, Error
	• PaymentDeclined
	  ○ Allowed from all statuses except: Expired, Aborted, Success, Error
	• Success
	  ○ Allowed from all statuses except: Aborted
	• Aborted
	  ○ Allowed from all statuses except: Expired, Success
	• Expired
	  ○ Allowed from all statuses except: Success, Aborted, Error
	• Error
	  ○ Allowed from all statuses except: Expired, Success, Aborted

order reference	The unique reference of the order that must be searched

Example: https://pbmapi.icepay.com/payment/status/e7140a88-cbba-4cc2-be9f-6e90536703ef/Expired/2016REF-A

2.3.1 Return values

When the status of the payment could be successfully changed, an http-code 200 will be returned.
If the status of the payment could not be changed, a 'BadRequest' with a describing message is returned.

2.4 Create a payment from a URL for a QR code (http GET)

Using the '**createUrl**' command a payment can be created for a specific entity using the supplied values from the url. When this url is clicked on, the customer gets redirected to the payment directly after it has been created with the supplied values. The <command> value in the API connection url must be replaced by: **createUrl/<entity key>**

Attention; if you want to create a payment in the system, do not use this method but use the 'create' method as described in chapter 2.1. Only use the method as described in this chapter if you want to send out a QR code.

Url parameter	Description
entity key	Every PayByMail entity within the system has its' own unique key. When connecting to the PayByMail system, this key must be supplied as well so PayByMail knows within which entity the payment must be created.

In this way, you can send out a lot of links to customers and a payment is only created in the PayByMail system when the customer clicks on that link. For instance; if you want to send out 200.000 invoices to customers but you're not sure if the person wants to pay the invoice using PayByMail, you can create 200.000 payment url's by yourself and only when the customer clicks on this link, a new payment is created in the PayByMail system and the user is redirected to the payment so he can pay instantly. Big advantage; you get invoiced by PayByMail for the payments that were created successfully. So when only 15.000 customers click on the link, your company will only be invoiced for the 15.000 created payments.

A secure connection, must NOT be used because the payment is created once the link has been clicked on by a customer. The security is implemented using a hash as described in this chapter.

2.4.1 Posted values

Parameter name	Description
MappingTemplate	With this parameter you can let the API know what mapping template must be used. If left empty; the default payment mapping for that entity is used
User	Valid API user name (mandatory)
Exu	The url where the customer is redirected to when the payment is expired. If the payment has not been created yet, it won't be created if the payment is expired.
Eru	The url where the customer is redirected to when an error occurs if he clicks on the link (for instance a technical error).
*	You can post any value. Only those values that are used in the mapping template will be used but they are all part of the HASH calculation.
Hash	A SHA256 hash is calculated with all the parameters that are supplied with the password of the API User at the end. The hash is then added as the last parameter.

The link to create a payment request has always the same layout:

https://testpbmapi.icepay.com/payment/createUrl/<guid of entity>?MappingTemplate=<valid mapping>&user=<valid API user name>&exu=<url to 'payment expired' page>&eru=<url to 'error' page>&…=…&…=…&hash=<valid hash>

An example of a valid create-payment-link with url-encoded parameters could be:
https://pbmapi.icepay.com/payment/createUrl/2b2b9bff-0c96-4f33-a20d-c831ac87e571?MappingTemplate=CreatePaymentLinkMapping&user=secretapiuser&exu=https%3a%2f%2fwww.icepay.com&eru=https%3A%2F%2Fwww.icepay.com%2Fen%2Fabout-us%2Fcontact%2F&l=nl-NL&c=EUR&a=28%2c50&ex=2016-02-29&or=201601171959&pn=Belastingaanslag&g=M&n=Jan+Bakker&m=jan%40PayByMail.com&p=06-12345678&hash=7395D3C1EFCBE26AF6EC8B895D5A8B62D5A3AB465A25D7B5B48008427A68C0D7

For this example entity an api user exists with the name '**Secretapiuser**' and the password '**T3st2016!**'.
In this example url, the following parameters are used;

Madatory parameters:

Parameter	**Value**
Entity key (part of the url)	2b2b9bff-0c96-4f33-a20d-c831ac87e571
MappingTemplate	CreatePaymentLinkMapping
User	Secretapiuser
Exu	https://www.icepay.com
Eru	https://www.icepay.com/en/about-us/contact/

Custom parameters:

Parameter	**Value**
L	nl-NL
C	EUR
A	28,50
Ex	2016-02-29
Or	201601171959
Pn	Belastingaanslag
G	M
N	Jan Bakker
M	jan@PayByMail.com
P	06-12345678

The hash will then be calculated using a SHA256 calculation over all these fields without the url-encoding, in the order that they are supplied and with the api password '**T3st2016!**' at the end. This password is NOT supplied via the url but only known and used on both sides to calculate the SHA256 hash using the supplied values:

```
2b2b9bff-0c96-4f33-a20d-c831ac87e571CreatePaymentLinkMappingSecretapiuserhttps://www.PayByMail.comhttps://www.icepay.com/en/about-us/contact/nl-NLEUR28,502016-02-29201601171959Belast ingaanslagMJan Bakkerjan@PayByMail.com06-12345678T3st2016!
```

This results in the hash value:
7395D3C1EFCBE26AF6EC8B895D5A8B62D5A3AB465A25D7B5B48008427A68C0D7

2.4.2 Return values

If the payment could be created successfully, or if a payment already exists, the customer is directly redirected to the payment page unless the payment is expired. Then the customer is redirected to the 'exu' url. If the payment could not be created, the customer is redirected to the 'eru' link. All the calls and parameters to and from the API are logged within the PayByMail system.

2.5 Create a mandate request via the API (http POST or http GET)

Using the '**create**' command a mandate request can be created for a specific entity using the supplied values.
The <command> value in the url must be replaced by: **create/<entity key>**
A secure connection, as described in the 'API Connection' chapter, must be created.

Url parameter	Description
entity key	Every PayByMail entity within the system has its' own unique key. When connecting to the PayByMail system, this key must be supplied as well so PayByMail knows within which entity the mandate request must be created.

Example: https://pbmapi.icepay.com/mandate/create/e7140a88-cbba-4cc2-be9f-6e90536703ef

2.5.1 Posted values

Parameter name	Description
MappingTemplate	With this parameter you can let the API know what mapping template must be used. If left empty; the default mandate-mapping for that entity is used
*	You can post any value to the API. Only those values that are used in the mapping template will be used but they're all logged in the system

2.5.2 Return values

When the mandate request could be created successfully, the Mandate Url is returned. This url can also be used by the calling application to send within its' own communication to the customer when 'SendType' = 'NoSend'.

If the mandate request could not be created successfully, a 'BadRequest' with a describing message is returned.
All the calls and parameters to and from the API are logged within the PayByMail system.

2.6 Find the url of a mandate request by the mandate number (http GET)

Using the '**url**' command a mandate request can be found for a specific mandate number.
The <command> value in the url must be replaced by: **url/<entity key>/<mandate number>**
A secure connection, as described in the 'API Connection' chapter, must be created.

Url parameter	Description
entity key	Every PayByMail entity within the system has its' own unique key. When connecting to the PayByMail system, this key must be supplied as well so PayByMail knows within which entity the mandate request must be created.
mandate number	The unique mandate number of the mandate request that must be searched

Example: https://pbmapi.icepay.com/mandate/url/e7140a88-cbba-4cc2-be9f-6e90536703ef/PAYBY1764

2.6.1 Return values

When the mandate request could be found successfully, the Mandate Url is returned.
If the request could not be found, a 'BadRequest' with a describing message is returned.

2.7 Change the status of a mandate by the mandate number (http POST)

Using the '**status**' command the status of a mandate request with the specified mandate number can be changed. The <command> value in the url must be replaced by: **status/<entity key>/<status>/<mandate number>**
A secure connection, as described in the 'API Connection' chapter, must be created.

Url parameter	Description
entity key	Every PayByMail entity within the system has its' own unique key. When connecting to the PayByMail system, this key must be supplied as well so PayByMail knows within which entity the mandate must be created.

status	The status that the mandate request must get. The following values are allowed;
	• LinkSent
	  ○ Only allowed from statuses: Booked, SendError
	• SendError
	  ○ Only allowed from statuses: Booked, LinkSent
	• MailDelivered
	  ○ Only allowed from statuses: Booked, LinkSent
	• MailOpened
	  ○ Only allowed from statuses: Booked, LinkSent, MailDelivered
	• LinkClicked
	  ○ Only allowed from statuses: Booked, LinkSent, MailDelivered, MailOpened
	• Success
	  ○ Allowed from all statuses except: Aborted
	• Aborted
	  ○ Allowed from all statuses except: Expired, Success
	• Expired
	  ○ Allowed from all statuses except: Success, Aborted, Error
	• Error
	  ○ Allowed from all statuses except: Expired, Success, Aborted

mandate number	The unique mandate number of the mandate request that must be searched

Example: https://pbmapi.icepay.com/mandate/status/e7140a88-cbba-4cc2-be9f-6e90536703ef/Expired/PAYBY95

2.7.1 Return values

When the status of the mandate request could be successfully changed, an http-code 200 will be returned.
If the status of the request could not be changed, a 'BadRequest' with a describing message is returned.

2.8 Create a mandate request from a URL for a QR code (http GET)

Using the '**createUrl**' command a mandate request can be created for a specific entity using the supplied values from the url. When this url is clicked on, the customer gets redirected to the mandate request directly after it has been created with the supplied values. The <command> value in the API connection url must be replaced by: **createUrl/<entity key>**

Attention; if you want to create a mandate request in the system, do not use this method but use the 'create' method as described in chapter 2.1. Only use the method as described in this chapter if you want to send out a QR code to customers and/or if you're not sure that the mandate request will be clicked on by the customer.

Url parameter	Description
entity key	Every PayByMail entity within the system has its' own unique key. When connecting to the PayByMail system, this key must be supplied as well so PayByMail knows within which entity the mandate request must be created.

In this way, you can send out a lot of links to customers and a mandate request is only created in the PayByMail system when the customer clicks on that link. For instance; if you want to send out 200.000 invoices to customers but you're not sure if the person wants to pay the invoice using a mandate, you can create 200.000 mandate

request url's by yourself and only when the customer clicks on this link, a new mandate request is created in the PayByMail system and the user is redirected to the request so he can start the e-mandate. Big advantage; you get invoiced by PayByMail for the mandate requests that were created successfully. So when only 15.000 customers click on the link, your company will only be invoiced for the 15.000 created mandate requests.

A secure connection, must NOT be used because the mandate request is created once the link has been clicked on by a customer. The security is implemented using a hash as described in this chapter.

2.8.1 Posted values

Parameter name	Description
MappingTemplate	With this parameter you can let the API know what mapping template must be used. If left empty; the default mandate mapping for that entity is used
user	Valid API user name (mandatory)
exu	The url where the customer is redirected to when the mandate request is expired. If the request has not been created yet, it won't be created if the request is expired.
eru	The url where the customer is redirected to when an error occurs if he clicks on the link (for instance a technical error).
*	You can post any value. Only those values that are used in the mapping template will be used but they are all part of the HASH calculation.
hash	A SHA256 hash is calculated with all the parameters that are supplied with the password of the API User at the end. The hash is then added as the last parameter.

The link to create a mandate request has always the same layout:

https://testpbmapi.icepay.com/mandate/createUrl/<guid of entity>?MappingTemplate=<valid mapping>&user=<valid API user name>&exu=<url to 'mandate request expired' page>&eru=<url to 'error' page>&…=…&…=…&hash=<valid hash>

An example of a valid create-mandate-request-link with url-encoded parameters could be:

https://testpbmapi.icepay.com/mandate/createUrl/2b2b9bff-0c96-4f33-a20d-c831ac87e571?MappingTemplate=CreateMandateLinkMapping&user=secretapiuser&exu=https%3a%2f%2fwww.icepay.com&eru=https%3A%2F%2Fwww.icepay.com%2Fen%2Fabout-us%2Fcontact%2F&l=nl-NL&c=EUR&a=28%2c50&ex=2016-02-29&mnr=201601171959&pn=Belastingaanslag&g=M&fn=Jan&ln=Bakker&m=jan%40PayByMail.com&p=06-12345678&s=Test%20street&nr=28&z=1234AB&cit=Nijmegen&cou=The%20Netherlands&hash=9DC09A79C9FB6D605D05416684CF43DBD772EA0131FF4783BE9D4622C1AD3E2D

For this example entity an api user exists with the name '**Secretapiuser**' and the password '**T3st2016!**'.
In this example url, the following parameters are used;

Madatory parameters:

Parameter	Value
Entity key (part of the url)	2b2b9bff-0c96-4f33-a20d-c831ac87e571
MappingTemplate	CreateMandateLinkMapping
user	Secretapiuser
exu	https://www.icepay.com
eru	https://www.icepay.com/en/about-us/contact/

Custom parameters:

Parameter	Value
l	nl-NL
c	EUR
a	28,50
ex	2016-02-29
mnr	201601171959
pn	Belastingaanslag
g	M

n	Jan Bakker
m	jan@PayByMail.com
p	06-12345678
s	Test street
nr	28
z	1234AB
cit	Nijmegen
cou	The Netherlands

The hash will then be calculated using a SHA256 calculation over all these fields without the url-encoding, in the order that they are supplied and with the api password '**T3st2016!**' at the end. This password is NOT supplied via the url but only known and used on both sides to calculate the SHA256 hash using the supplied values:

```
2b2b9bff-0c96-4f33-a20d-c831ac87e571CreateMandateLinkMappingSecretapiuserhttps://www.PayByMail.comhttps://www.icepay.com/en/about-us/contact/nl-NLEUR28,502016-02-29201601171959Belast ingaanslagMJan Bakkerjan@PayByMail.com06-12345678Test street281234ABNijmegenThe NetherlandsT3st2016!
```

This results in the hash value:
9DC09A79C9FB6D605D05416684CF43DBD772EA0131FF4783BE9D4622C1AD3E2D

2.8.2 Return values

If the mandate request could be created successfully, or if a request with this mandate number already exists, the customer is directly redirected to the mandate request page unless it is expired. If the request is expired, the customer is redirected to the 'exu' url. If the mandate request could not be created, the customer is redirected to the 'eru' link. All the calls and parameters to and from the API are logged within the PayByMail system.

2.9 Create a direct collect request via the API (http POST or http GET)

Using the '**create**' command a direct collect request can be created for a specific entity using the supplied values.
The <command> value in the url must be replaced by: **create/<entity key>**
A secure connection, as described in the 'API Connection' chapter, must be created.

Url parameter	Description
entity key	Every PayByMail entity within the system has its' own unique key. When connecting to the PayByMail system, this key must be supplied as well so PayByMail knows within which entity the direct collect request must be created.

Example: https://pbmapi.icepay.com/collect/create/e7140a88-cbba-4cc2-be9f-6e90536703ef

2.9.1 Posted values

Parameter name	Description
MappingTemplate	With this parameter you can let the API know what mapping template must be used. If left empty; the default direct-collect-mapping for that entity is used
*	You can post any value to the API. Only those values that are used in the mapping template will be used but they're all logged in the system

2.9.2 Return values

When the direct collect request could be created successfully, the unique key of the created direct collect request is returned.

If the direct collect request could not be created successfully, a 'BadRequest' with a describing message is returned. All the calls and parameters to and from the API are logged within the PayByMail system.

3. Callback(s) to external system(s)

Callbacks can be used to inform external users or systems whether a payment, mandate or direct collect changed status. Multiple callbacks can be issued to inform multiple systems of a certain action within the system.

3.1 Additional callback parameters

The following additional parameters can be used when a callback trigger is used;

Triggers	Description
*CallbackType	The technical status message (for instance: Aborted, Error, Success, etc.)
*CallbackMessage	The readable status message of the item (ie: 'The payment is aborted' or 'The mandate is successfully signed'
*CallbackMessage(<language code>)	The readable status message of the item in the supplied language (ie: 'The payment is aborted' or 'The mandate is successfully signed')
*CallbackDetailedMessage	A detailed message if available (ie: 'Insufficient funds (AM04)')
*PspOrderReference	The order id that was given to this payment by the entity when starting the payment
*Username	The full name of the user that created the payment (if created manually)
*PspTransactionId	The (technical) transaction id that was used by the psp to follow the payment
*PspOrderId	The order id that was given to this payment by the psp
*PspName	The name of the psp that was used as set by the entity (ie: Main Molly Account)
*PspType	The psp type that was used (ie: Ingenico, Mollie, Buckaroo, Adyen, etc.)
*PaymentMethod	The payment method that was used on a successful payment (ie: iDEAL, Master card, etc.)
*PaymentMethodCode	The code of the payment method that was used for a successful payment (ie: IDEAL, MASTERCARD, etc.). See glossart A for a list of codes

3.2 Callback triggers

The following triggers can be used to issue a callback;

Trigger	Message	Description	Payments	Mandates	Direct Col.
on link not delivered	LinkNotDelivered	Issued when a link could not be delivered by mail	X	X	-
on send error	SendError	Issued when there was an issue when sending the link by mail or sms	X	X	-
on reminder sent	Reminder	Issued when a reminder is sent to the customer	X	X	-
on payment declined/refused	Declined	Issued when a payment is declined by the PSP	X	-	-
on success	Success	Issued when a payment was successfully paid or when a mandate was successfully signed	X	X	X
on expired	Expired	Issued when a link is expired	X	X	-
on aborted	Aborted	Issued when a link is aborted (by the user/api)	X	X	X
on error	Error	Issued when a (system) error occurs	X	X	X
on reversed payment	Reversed	Issued when a charge back occurs	-	-	X
on failed	Failed	Issued when the direct collect failed	-	-	X
on direct collect sent	DirectCollectSent	Issued when the direct collect is sent to the financial institute	-	-	X

The value for ‘Message’ is returned by the system parameter [*CallbackType]. The following paragraphs describe the possible actions that can be performed. In all these callbacks, parameters within the system can be used within the message. Custom created parameters like (for instance) [FlightNumber] or [RoomType] can be used as well.

3.3 Callback by mail

When a callback by mail is issued, a mail is sent to one or more mail addresses as set in the parameters. If multiple mail addresses must receive an e-mail, all the addresses can be put in one field, separated by a semi-colon.

Parameter	Description	Example
Address(es)	The mail address(es) that must receive an e-mail	john@doe.nl; jane@doe.nl

The contents of the mail that is sent with this callback, can be changed with the ScreenText called 'Callback Result Mail'.

3.4 Callback by url

To inform an external system in case of a specific trigger, a custom connection is a possibility but this costs time and money. Also when the external system is changed, the connections to it must be altered as well.
That is why we have chosen to create a very generic callback-by-url method that can be used to inform systems in a generic http way using http-get, http-post and http-put. If the external system doesn't support this callback, a custom connection can be created as well as a separate project.

Parameter	Description	Example
Url	The Url (optionally with parameters) that must be issued for this trigger	https://trans.system.nl/Verify.svc/SetToPaid?ResID=[*OrderId]&Amount=[*AmountNumber]
Username	The username in case of a secure connection	Theusername
Password	The password in case of a secure connection	S3cr3t

When a secure connection is used with a username and a password, it is strongly advised to use HTTPS.

3.4.1 Using http GET

With http GET, only a url and (optionally) a username and password can be entered because that is all that will be send to the external system.

3.4.2 Using http POST or http PUT

With http POST or http PUT, besides a url and (optionally) a username and password, a body can be send as well. This body is made up out of three different parts;

Parameter	Description	Example
Header	This is the first part of the body and can be set up to identify starting elements in case of (for instance) an XML message	<?xml version="1.0" encoding="UTF-8" ?> <result flightno="[FlightNumber]">
Body	The second part can contain the important elements that describe the contents of a message. For instance what the result is of a payment, the amount, etc.	<payment paid="true"> <orderid>[*OrderID]</orderid> <amount>[*AmountNumber]</amount> </payment>
Footer	This is the last part of the body and can be used to identify any closing elements	</result>

If the examples above would be used, the following message would be send to the system as body of the http message:

```
<?xml version="1.0" encoding="UTF-8" ?>
<result flightno="[FlightNumber]">
  <payment paid="true">
    <orderid>[*OrderID]</orderid>
    <amount>[*AmountNumber]</amount>
```

</payment>
</result>

Of course other (non-xml) information can be used in these fields as well.

3.5 Callback to an FTP server

In some cases, a system requires a file to be placed on a specific location using an FTP server. This can performed by this callback method.

Parameter	Description	Example
FTP Server	The host or IP address of the ftp server	10.0.0.5
Port number	The number of the port that must be used for this connection	21
Username	The name of the FTP user for this connection	Theftpuser
Password	The password of the FTP user for this connection	S3cr3tFTP
Folder	The name of the folder on the FTP server where a file must be created	PBL\PaymentCallback
Filename	The name of the file that will be created. This can be a 'fixed' file name but also parameters or functions can be used like [*UniqueId]	[*UniqueId].csv
Header	The first part of the body in the file	ref;amount;date;paymethod;status
Body	The second part of the body of the file	[*OrderId];[*Amount];[*DateNow]; [*PaymentMethod];[*CallbackType]
Footer	The third and last part of the body of the file	

Combined, the example values will return in a file with a unique number that will be placed in the folder PBL\PaymentCallBack and with the contents:

ref;amount;date;paymethod;status
[*OrderId];[*Amount];[*DateNow]; [*PaymentMethod];[*CallbackType]

3.6 Callback to RezExchange

RezExchange is a system that is used by hotels to connect to their reservation system. This callback is used by those hotels to inform their backend reservation system of a payment that is made by a guest. This will result in a permanent booking and a successful reservation.

Because a generic callback to RezExchange wasn't possible a custom connection was created.
For the callback, RezExchange needs secure connection details, a reservation number and a code for the payment method that was used for this payment. These details are supplied by the company that offers RezExchange to the hotel.

Connecting to the external system. If a hotel chain has multiple entities, the connection can be set up in the license entity and is then used by all the entities within this license. You must set up the connection one time and then, I the other entity settings, use the check box '**Use parent connection settings**' to indicate that the license settings must be used to connect to RezExchange.

Parameter	Description	Example
Url	The url to the RezExchange service	https://azrxcwi.itsrezexchange.net/RX10/PBL/Service.asmx
Username	The name of the user	PayByMail
Password	The password of the user	S3cr3t
Property code	Every hotel has an own (unique) code that is used to identify it at RezExchange.	HOTELBOTEL
Payment method	Every payment mehod that can be used to pay for a reservation, must be mapped to an	JCB = 30 Maestro = 31

	internal (hotel-specific) payment method code that is known by RexExchange.	iDEAL = 32 Mastercard = 30 VISA = 33

4. Payment, mandate and direct collect glossary

This chapter describes what parameters are needed or can be used to create fixed, variable or partial payments or to create a mandate.

- = not allowed (must be left empty)
X = mandatory
O = mandatory with rules

4.1 Payment parameters

The system uses a range of parameters internally to create a fixed payment (only one amount), a variable payment (the user can choose out of multiple amounts or can enter his own amount within a specified range, for instance for donations), a partial payment setup or a 'direct collect' payment. With a partial payment the total amount is spread out over two or more payments within a specific time. A direct collect can only be initiated if a mandate contract is available.

In these payment types the system parameters can be used and also system functions can be used as described in the first chapter.
Own parameters can be added too if needed in logging or communication to the customer. You can always create your own internal parameters and set them to fixed values or to values that were supplied to the system within an external api parameter. The names of the parameters can be chosen freely, as long as they are mapped within the PayPerLink system to the mandatory system parameters.

The following system parameters can be used in the mapping:

			Mandatory per 'payment type'			
Output name	**Description**	**Format**	**Fixed**	**Variable**	**Partial**	**Comments**
*LanguageCode	The payment language	xx-XX (ie: en-US)				If empty: entity language is used
*CurrencyCode	The currency of the amount	XXX (ie: EUR, USD)				If empty: entity currency is used
*Amount	The total amount to pay	Decimal (xx,xx or xx.xx)	X	-	X	
*VariableAmounts	2 or more amounts where the customer can choose from or enter an own amount within a range	<v1>#<v2>#<v3>… Ie: 5#7,50#10#12,50#15 <v1>#<v2>#range:<v3>:<v4> Ie: 5#50#range:10:45	-	X	-	
*Description	A brief order description	String				If empty: no company name
*OrderID	A unique order ID	String	X	X	X	Must be unique per entity
*Gender	Gender of the client	Char (M, F or U)				If empty: U (unknown) is used
*ClientName	Name of the client	String	X	X	X	
*CompanyName	Company name	String				
*MailAddress	Mail address	String (xx@xx.xx)	O	O	O	Mandatory when SendType is mail
*PhoneNumber	(Mobile) phone number	String	O	O	O	Mandatory when SendType is sms
*SendType	Indicates via what medium a payment link must be send	mail, sms or nosend				If empty: nosend

*SendDateTime	The date when a payment must be send to the customer	Date				If empty: current date and time (send immediately)
*ExpirationDateTime	The date that this payment will expire	Date				If empty: default expiration time after send is used
*MailTemplate	The name of the mail template that must be used	String (name of the mail template)				If empty and *SendType is mail: default mail template
*AttachmentUrl	The url to a pdf attachment that can be downloaded from the Internet or from an ftp/sftp server. The username and password can be added before the url.	String (see comments below for more information)				If supplied, only a pdf file with a maximum of 2mb is allowed. Url must start with: http:// or https:// or ftp:// or sftp://
	A valid value must be supplied in the form: <user>#<pwd>#<url> or: <url> (if no user/pwd is used) The <url> can contain the port number to connect to as well: For instance: ftp://spbl.eu:21/doc.pdf In case of an FTP server, extra parameters 'ssl' / 'nossl' and/or 'active' / 'passive' can be send too, to indicate if an SSL or NO SSL (default) connection and/or ACTIVE or PASSIVE (default) ftp connection must be used. For instance: adm#pwd1#active#ssl#ftp://pb.eu/a/x.pdf or adm#pwd1#nossl#ftp://pb.eu/a/x.pdf					
*PaymentMethodCodes	A list of allowed payment method codes for this payment, seperated with #. See further chapter for a list of allowed codes.	<CODE1>#<CODE2>#... Ie:IDEAL#VISA#MASTERCARD#BANCONTACT				If empty, the default enabled payment methods on entity-level are used. Only payment methods that are enabled on entity level can be supplied
*PartialTemplate	If 'partial payments' must be created, this parameter holds the name of the template	String (name of the template)	-	-	O	Only one of these parameters can have a value: [*PartialTemplate] [*PartialPercentages] [*PartialAmounts] [*PartialPercentageAmounts]
*PartialPercentages	If 'partial payments' must be created, this parameter holds the percentages per partial	<perc1>#<perc2>#<perc3>… Ie: 20#30#40#10 means 4 partial payments	-	-	O	
*PartialAmounts	If you want to supply your own amounts for each partial, you can use this parameter. Otherwise the percentages will be used to calculate the amounts for each partial payment using the *Amount value	<am1>#<am2>#<am3>… The sum of all these amounts MUST be equal to the *Amount value	-	-	O	
*PartialPercentageAmounts	If a the payment must be paid in several parts, the percentages and/or amounts for each payment can be set here, seperated with # and must sum up to the amount entered at *Amount	<amnt/perc1>#<amnt/perc2> 150#50%#? indicates three partial payments for an amount of 150, 50% and the remaining amount by indicating ?	-	-	O	
*PartialExpirationDates	If you want to supply your own expiration date per partial, you can use this parameter.	<date1>#<date2>#<date3>… The last date MUST be equal to the *ExpirationDateTime value	-	-		If empty, the percentages/amounts will be used to calculate the time between each partial payment using the time between the '*SendDateTime' value and '*ExpirationDateTime' value

*PartialFreeAmountsAllowed	Used to indicate if the user is allowed to enter a free amount between the current partial payment amount and the total outstanding amount	true/false	-	-		If this value is empty and PartialPercentages, PartialAmounts or PartialExpirationDates are filled, 'false' is used
*PartialWholeAmountAllowed	Used to indicate if the user is allowed to pay the total outstanding amount at once	true/false	-	-		If this value is empty and PartialPercentages, PartialAmounts or PartialExpirationDates are filled, 'false' is used

4.2 Mandate parameters

The following system parameters can (or must) be used in the mapping to create a valid mandate request:

Output name	Description	Format	Mandatory	Comments
*MandateContract	The name of the contract that must be used for this mandate	String	X	
*LanguageCode	The mandate request language	xx-XX (ie: en-US)		If empty: entity language is used
*MandateNumber	The unique number for this mandate request	String	?	If empty: the mandate provider will create one
*MandateProviderLinkUrl	An optional link to the mandate if this is known already	String	?	If empty: the mandate provider will create one. If supplied, [*MandateNumber] is mandatory too
*Description	A brief description for this mandate	String	X	
*CurrencyCode	The currency of the amount	XXX (ie: EUR, USD)		If empty: entity currency is used
*Amount	The total amount to pay	Decimal (xx,xx or xx.xx)		
*Gender	Gender of the client	Char (M, F or U)		If empty: U (unknown) is used
*FirstName	The first name of the client	String	?	Not mandatory for Twikey
*LastName	The last name of the client	String	?	Not mandatory for Twikey
*MailAddress	Mail address	String (xx@xx.xx)	X	Always mandatory
*PhoneNumber	(Mobile) phone number	String		Only mandatory when SendType is sms
*Street	The street of the customer or company	String	X	
*Number	The number for the street	String	X	
*ZipCode	The zip code of the address	String	X	
*City	The city name of the address	String	X	
*Country	The country of the address	String	X	
*CompanyName	Company name	String	?	Mandatory if company number is supplied
*CompanyNumber	Company number (IE: VAT number)	String	?	Mandatory if company name is supplied
*SendType	Indicates via what medium a mandate request link must be send	mail, sms or nosend		If empty: nosend
*SendDateTime	The date when a mandate request must be send to the customer	Date		If empty: current date and time (send immediately)
*ExpirationDateTime	The date that this request will expire	Date		If empty: default expiration time after send is used

Output name	Description	Format	Mandatory	Comments
*MailTemplate	The name of the mail template that must be used	String (name of the mail template)		If empty and *SendType is mail: default mail template

4.3 Direct collect parameters

The following system parameters can (or must) be used in the mapping to create a valid direct collect request:

Output name	Description	Format	Mandatory	Comments
*DirectCollectContract	The name of the contract that must be used for this request	String (name of the contract)	X	
*ContractNumber	The number of the direct collect contract under which it is known at the direct collector	String	X	This is mostly the reference under which the mandate is created. Ie the mandate number
*OrderId	The unique number for this direct collect for this customer	String	X	
*Description	A description (the reason) of this direct collect	String	X	For instance; “Monthly rent for September”
*CurrencyCode	The currency of the amount	XXX (ie: EUR, USD)		If empty: entity currency is used
*Amount	The total amount to pay	Decimal (xx,xx or xx.xx)	X	
*CollectDate	The date when the amount must be collected	Date		If empty: the amount is collected immediately (within 24 hours)
The next items are only necessary if you want the system to create a payment link automatically when the direct collect fails because of an error or when the customer started a reverse charge to return the collected amount back				
*CreatePaymentOnFailure	Used to indicate if a payment link must be created after a reversed charge or an error	True, False		Default: False
*PaymentMethodCodes	A list of allowed payment method codes for this payment, seperated with #. See further chapter for a list of allowed codes.	<CODE1>#<CODE2>#... Ie:IDEAL#VISA#MASTERCARD#BANCONTACT		If empty, the default enabled payment methods on entity-level are used. Only payment methods that are enabled on entity level can be supplied
*ExpirationDateTime	The moment when a payment expires. If left empty, the default expiration is calculated using the info on the 'Payment info' tab of the entity settings.	Date		Only one of these parameters can be entered. If both left empty, the default expiration date is calculated based on the date that the payment is created after the reverse charge occurred
*RelativeExpirationHours	The amount of hours AFTER the payment has been created that indicates when a payment expires. If left empty, the default expiration is calculated using the info on the 'Payment info' tab of the entity settings.	Number		
*LanguageCode	The mandate request language	xx-XX (ie: en-US)		If empty: entity language is used

Output name	Description	Format	Mandatory	Comments
*Gender	Gender of the client	Char (M, F or U)		If empty: U (unknown) is used
*FirstName	The first name of the client	String	X	
*LastName	The last name of the client	String	X	
*CompanyName	Company name	String		
*MailAddress	Mail address	String (xx@xx.xx)	O	Only mandatory when SendType is mail
*PhoneNumber	(Mobile) phone number	String	O	Only mandatory when SendType is sms
*SendType	Indicates via what medium a mandate request link must be send	mail, sms or nosend	O	Only mandatory when CreatePaymentOnFailure = True
*MailTemplate	The name of the mail template that must be used	String (name of the mail template)		If empty and *SendType is mail: default mail template will be used

4.4 Usage of dates

When dates and times are used in the mapping, these are always interpreted according to the language settings of the entity.

For instance; if the entity language is set to 'Dutch' the value '12-11-2016, 07:00' would be interpreted as 7 am on November 12th 2016.
When the language was set to 'English', the same value would have been interpreted as 7 am on December 11th 2016. So it is very important to check if the date values that are supplied to the mapping ar2e in the same format as the language of the entity.

A. Payment method codes

When payment methods codes are supplied when a payment is created, the following codes are available:

Name	Code
7-Eleven	7ELEVEN
AMEX	AMEX
Baloto	BALOTO
Bancontact	BANCONTACT
Bankoverschrijving	BANKOVERSCHRIJVING
Belfius Direct Net	BELFIUS
Bitcoin	BITCOIN
Boleto Bancario	BOLETO
CBC Online	CBC
Diners	DINERS
ELV	ELV
Fashioncheque	FASHIONCHEQUE
giropay	GIROPAY
iDEAL	IDEAL
ING HomePay	INGHOMEPAY
JCB	JCB
KBC Online	KBC
Maestro	MAESTRO
MasterCard	MASTERCARD
MasterPass	MASTERPASS
Nationale EntertainmentCard	ENTERTAINMENTCARD
Oxxo	OXXO
Pago Efectivo	PAGOEFECTIVO
Pago Facil	PAGOFACIL
PayPal	PAYPAL
paysafecard	PAYSAFECARD

PODIUM CK	PODIUMCK
PSE	PSE
Safetypay	SAFETYPAY
SEPA Direct Debit	SEPADIRECTDEBIT
Sofort (AT)	SOFORTAT
Sofort (BE)	SOFORTBE
Sofort (DE)	SOFORTDE
Sofort (IT)	SOFORTIT
VISA	VISA
Vpay	VPAY
VVV Cadeaukaart	VVV
Webshop Giftcard	WSGIFTCARD
Western Union	WESTERNUNION
Wire Transfer	WIRETRANSFER
YourGift Card	YOURGIFTCARD

B. Variables and functions

The following variables and functions are available for mails, screen texts, callbacks and/or mappings:

Parameter	Description
[*AddDays(<date>, <amount>)]	Get the supplied date value and add some days (ie: [*AddDays(12-09-1975, 5)] returns: 17-09-1975)
[*AddHours(<date>, <amount>)]	Get the supplied date value and add some hours (ie: [*AddHours(12-09-1975 16:20, 2)] returns: 12-09-2015 18:20)
[*AddMinutes(<date>, <amount>)]	Get the supplied date value and add some minutes (ie: [*AddMinutes(12-09-1975 16:20, 50)] returns: 12-09-2015 17:10)
[*AddMonths(<date>, <amount>)]	Get the supplied date value and add some months (ie: [*AddMonths(12-09-1975, -2)] returns: 12-07-1975)
[*AddSeconds(<date>, <amount>)]	Get the supplied date value and add some seconds (ie: [*AddSeconds(12-09-1975 16:20, 120)] returns: 12-09-2015 16:22)
[*AddYears(<date>, <amount>)]	Get the supplied date value and add some years (ie: [*AddYears(12-09-1975, 40)] returns: 12-09-2015)
[*Amount]	The amount that must be paid
[*AmountDue]	The amount due (remaining amount of the total payment)
[*AmountDueNumber]	The amount due as number using the entity language code (ie: 73.25)
[*AmountDueText]	The amount due as text including the currency symbol using the entity language code (ie: € 73,25)
[*AmountNumber]	The amount as number using the entity language code (ie: 128.50)
[*AmountText]	The amount as text including the currency symbol using the entity language code (ie: € 128,50)
[*AttachmentUrl]	A url that points to an file that will be attached to a payment mail when it is sent.
[*Bc]	Get a bracket close value:]
[*Bo]	Get a bracket open value: [
[*Calculate(<formula>)]	Perform a calculation and return the result (ie: [*Calculate(2 * 4 + 5)] returns: 13)
[*CalculateHash(<SHA1, SHA256 or SHA512>, <password>, <v1>, <v2>)]	Calculates a hash value using the supplied password and values concatenated (ie: [*CalculateHash(SHA512, P@ssw0rd, myFirstValue, otherValue)])
[*CallbackDetailedMessage]	A detailed message if available (ie: 'Insufficient funds (AM04)')
[*CallbackMessage(<language code>)]	The readable status message of the item in the supplied language (ie: 'The payment is aborted' or 'The mandate is successfully signed'
[*CallbackMessage]	The readable status message of the item (ie: 'The payment is aborted' or 'The payment is successfully paid'
[*CallbackType]	The technical status message of the payment (possible values are: MailNotDelivered, Expired, Aborted, Reminder, Declined, Success, PaymentError or SendError)
[*City]	The city of the address
[*ClientGender]	The gender of the client. Accepted values are: M (for male), F (for female) or U (for unknown, this is the default).
[*ClientGenderText]	The gender of the client translated to the right salutation in the specified language (ie: 'mr.', 'mrs.', 'mr./mrs.')
[*ClientMailAddress]	The mail address of the client (mandatory if a mail must be send)
[*ClientName]	The full name of the client
[*ClientPhoneNumber]	The phone number of the cient (mobile number is mandatory if an sms must be send)
[*Comma]	Get a comma value

Parameter	Description
[*CompanyName]	The company name of the client
[*CompanyNumber]	The company number of the client
[*Country]	The country name of the address
[*CreatePaymentOnFailure]	Set this value to 'true' to create a payment link when the direct collect fails (default is 'false')
[*CurrencyCode]	The ISO currency code for the amount of this payment (ie: EUR, USD, etc.)
[*CurrencySymbol]	The currency symbol of this payment (ie: €, $, etc.)
[*CurrentDate]	Get the current date, using the language code of the payment (ie: '18-08-2016' for a 'Dutch' language)
[*CurrentDateTime]	Get the current date and time, using the language code of the payment (ie: '18-08-2016, 13:35' for a 'Dutch' language
[*CurrentPartialPaymentNumber]	In case of a partial payment; the number of the current partial payment
[*CurrentTime]	Get the current time, using the language code of the payment (ie: '1:35 PM' for an 'English' language)
[*DateNow]	Get the date of this exact moment
[*DateOnly(<value>)]	Get the date only of the supplied date using the entity language code (ie: [*DateOnly(1975-09-12 16:20)] returns 12-09-1975 when the language code is Dutch)
[*DateOnly(<value>, <language code>)]	Get the date only of the supplied date using the supplied language code (ie: [*DateOnly(12-09-1975 16:20, en-US)] returns 12-09-1975 because the language code is English)
[*DateTime(<value>)]	Convert the given <value> to a valid date time value using the entity language code (ie: [*DateTime(1975-09-12 16:20)] returns a date for the 12th of september 1975 at 4:20pm when the language is set to Dutch)
[*DateTime(<value>, <language code>)]	Convert the given <value> to a valid date time value using the entity language code (ie: [*DateTime(09-12-1975 6:20 pm, en-US)] returns a date for the 12th of september 1975 at 4:20pm because the language is set to English)
[*Day(<date>)]	Get the day of the supplied value (ie: [*Day(12-09-1975)] returns: 12
[*Decimal(<value>)]	Convert a supplied value to a decimal number using the language code of the entity (ie: [*Decimal(1000.50)] returns a decimal 1000.50 when the language is English)
[*Decimal(<value>, <language code>)]	Convert a supplied value to a decimal number using the supplied language code (ie: [*Decimal(1000.50, en-US)] returns a decimal 1000.50 because the language is set to English)
[*Description]	A description of the payment (ie: hotel reservation on saturday)
[*EntityAdditionalImage]	The url to the additional image of the entity
[*EntityBackgroundColor]	The background color for this entity in #0099FF form
[*EntityBodyColor]	The background color of the body for this entity in #0099FF form
[*EntityBodyTextColor]	The body text color for this entity in #0099FF form
[*EntityButtonBorderColor]	The button border color for this entity in #0099FF form
[*EntityButtonColor]	The button color for this entity in #0099FF form
[*EntityButtonTextColor]	The button text color for this entity in #0099FF form
[*EntityDepartment]	The name of the department of the entity
[*EntityDisplayName]	The display name of the entity
[*EntityExpiredBackgroundColor]	The expired box background color for this entity in #0099FF form
[*EntityExpiredBorderColor]	The expired box border color for this entity in #0099FF form

Parameter	Description
[*EntityExpiredFontColor]	The expired box font color for this entity in #0099FF form
[*EntityFontFamily]	The font family for this entity
[*EntityFontSize]	The font size for this entity
[*EntityFooterBorderColor]	The selected footer border color for this entity in #0099FF form
[*EntityFooterColor]	The footer background color for this entity in #0099FF form
[*EntityFooterTextColor]	The footer text color for this entity in #0099FF form
[*EntityHeaderBorderColor]	The header border color for this entity in #0099FF form
[*EntityHeaderColor]	The header background color for this entity in #0099FF form
[*EntityLinkColor]	The link color for this entity in #0099FF form
[*EntityMailAddress]	The main mail address of the entity
[*EntityMailHeaderImage]	The url to the mail header image of the entity
[*EntityMailTemplateFile]	The url to the mail template file of the entity
[*EntityMainAddressCity]	The city of the main address of the entity
[*EntityMainAddressCountry]	The country of the main address of the entity
[*EntityMainAddressExtraAddressLine]	The extra address line of the main address of the entity
[*EntityMainAddressNumber]	The street number of the main address of the entity
[*EntityMainAddressState]	The city of the main address of the entity
[*EntityMainAddressStreet]	The street of the main address of the entity
[*EntityMainAddressZipCode]	The zip code of the main address of the entity
[*EntityName]	The contract name of the entity
[*EntityPhoneNumber]	The main phone number of the entity
[*EntityPostalAddressCity]	The city of the postal address of the entity
[*EntityPostalAddressCountry]	The country of the postal address of the entity
[*EntityPostalAddressExtraAddressLine]	The extra address line of the postal address of the entity
[*EntityPostalAddressNumber]	The street number of the postal address of the entity
[*EntityPostalAddressState]	The city of the postal address of the entity
[*EntityPostalAddressStreet]	The street of the postal address of the entity
[*EntityPostalAddressZipCode]	The zip code of the postal address of the entity
[*EntityReminderBackgroundColor]	The reminder box background color for this entity in #0099FF form
[*EntityReminderBorderColor]	The reminder box border color for this entity in #0099FF form
[*EntityReminderFontColor]	The reminder box font color for this entity in #0099FF form
[*EntityScreenHeaderImage]	The url to the screen header image of the entity
[*EntitySuccessBackgroundColor]	The success box background color for this entity in #0099FF form
[*EntitySuccessBorderColor]	The success box border color for this entity in #0099FF form

Parameter	Description
[*EntitySuccessFontColor]	The success box font color for this entity in #0099FF form
[*EntityWebsite]	The url to the website of the entity
[*ExpirationDate]	The expiration date of the payment
[*ExpirationDateTime]	The moment when a payment expires. If left empty, the default expiration is calculated using the info on the 'Payment info' tab of the entity settings.
[*ExpirationTime]	The expiration time of the payment
[*FirstName]	The first name of the client
[*FormatDateTime(<value>, <format>)]	Formats the supplied date using the specified format and the entity language code (ie: [*FormatDateTime(1975-09-12 16:20, MMMM)] returns: September when the language code is English)
[*FormatDateTime(<value>, <format>, <language code>)]	Formats the supplied date using the specified format and the supplied language code (ie: [*FormatDateTime(1975-09-12 16:20, dddd dd MMMM, nl-NL)] returns vrijdag 12 september)
[*FormatDecimal(<value>, <language code>)]	Format a supplied decimal value to a text in the supplied language code (ie: [*FormatDecimal(1000.50, nl-NL)] returns: 1000,50 because the language is set to Dutch)
[*FormatDecimalText(<value>, <language code>)]	Format a supplied decimal value to a text with thousands separator and decimal seperator in the supplied language code (ie: [*FormatDecimalText(1005, nl-NL)] returns: 1.005,00 because the language is set to Dutch)
[*FreeAmountsAllowed]	In case of a partial payment; 'true' if a free amount can be entered by the customer
[*Gender]	The gender of the client. Accepted values are: M (for male), F (for female) or U (for unknown, this is the default).
[*GenderText]	The gender of the client translated to the right salutation in the specified language (ie: 'mr.', 'mrs.', 'mr./mrs.')
[*GetParameterValue(<value>)]	Get a value of a parameter, only if it exists! This prevents parsing errors from the system if a value is used that doesn't exist.
[*Hour(<date>)]	Get the hours of the supplied value (ie: [*Hour(12-09-1975 16:20)] returns: 16)
[*HtmlEncode(<value>)]	Encode the value so that it can be used in html (ie: [*HtmlEncode(<this & that>)] returns: <this & that>
[*If(<condition>, <value if true>, <value if false>)]	Check a condition a return the value that applies to the outcome to the condition (ie: [*If(4>2,this is true,this is not true)] returns: this is true)
[*LanguageCode]	The ISO language code for this payment (ie: nl-NL, en-US, etc.)
[*LastExpirationDate]	In case of a partial payment; the final expiration date
[*LastExpirationDateTime]	In case of a partial payment; the final expiration date and time
[*LastExpirationTime]	In case of a partial payment; the time of the final expiration date
[*LastName]	The last name of the client
[*LongDateOnly(<value>)]	Get a long date including the day and month names for the supplied date using the entity language code (ie: [*LongDateOnly(12-09-1975)] returns Vrijdag 12 september 1975 when the language code is Dutch)
[*LongDateOnly(<value>, <language code>)]	Get a long date including the day and month names for the supplied date using the supplied language code (ie: [*LongDateOnly(12-09-1975, nl-NL)] returns Vrijdag 12 september 1975 because the language code is Dutch)
[*LongTimeOnly(<value>)]	Get a long time including the seconds and milliseconds for the supplied date using the entity language code (ie: [*LongDateOnly(12-09-1975 16:20)] returns 16:20:00.000 when the language code is Dutch)
[*LongTimeOnly(<value>, <language code>)]	Get a long time including the seconds and milliseconds for the supplied date using the supplied language code (ie: [*LongDateOnly(12-09-1975 16:20, en-US)] returns 16:20:00.000 because the language code is English)

Parameter	Description
[*MailAddress]	The mail address of the client (mandatory if a mail must be send)
[*MailTemplate]	If a mail must be sent, then the name of the template can be supplied. If left empty, the default mail template for the supplied language code is used
[*MandateButton(<button text>)]	The mandate link wrapped inside a button for use in HTML with a text (ie: [*MandateButton(Click me for authorization)]
[*MandateButton]	The mandate button for use in HTML with a default text. Use the [*MandateButton(..)] function to set your own text, ie: [*MandateButton(Subscribe now)]
[*MandateContract]	The name of the contract that must be used for this mandate
[*MandateLink]	The mandate link for use in HTML (ie: http://link)
[*MandateLinkUrl]	The url of the mandate link without HTML tags (ie: http://link)
[*MandateNumber]	Optional, a unique order id for this mandate, mostly a number (ie: 19750912)
[*MandateProviderContractName]	The name of contract of the mandate provider that was used for this mandate (ie: Consumer Mandate Account)
[*MandateProviderLink]	The mandate provider link for use in HTML (ie: http://link)
[*MandateProviderLinkUrl]	Optional, the url of the mandate provider link (ie: http://link)
[*MandateProviderName]	The name of the mandate provider that was used as set by the entity (ie: Main Twikey Account)
[*MandateProviderType]	The mandate type that was used (ie: Twikey)
[*Minute(<date>)]	Get the minute of the supplied value (ie: [*Minute(12-09-1975 16:20)] returns: 20)
[*Month(<date>)]	Get the month of the supplied value (ie: [*Month(12-09-1975)] returns: 9
[*NoMandateLink]	Used to indicate that no mandate link is used
[*NoPayment]	Used to indicate that no payment link is used
[*NoPaymentLink]	Used to indicate that no payment link is used
[*Number]	The address number
[*OrderId]	A unique order id for this payment, mostly a number (ie: 19750912)
[*Origin]	The way this payment was created (possible values are: Manually, Batch, Api, QR or Ftp)
[*ParameterExists(<value>)]	Check if a parameter exists. This can be handy when you only want to use a value when it exists, depending on the mapping. This can be used with an [*If] condition.
[*PartialAmounts]	If a the payment must be paid in several parts, you can use the percentages to let the system calculate the partial amounts of each partial payment or enter them here, seperated with # and must sum up to the amount entered at *Amount (ie: when *Amount is 120,00 you can set this value to: 30#40,50#49,50)
[*PartialExpirationDates]	If a the payment must be paid in several parts, you can use the percentages to let the system calculate the expiration date of each partial or enter them here, seperated with # and the last date must be equal to the *ExpirationDateTime (ie: when *ExpirationDateTime is 1-3-2017 13:00 you can set this value to: 31-12-2016 13:00#31-1-2017 13:00#1-3-2017 13:00)
[*PartialFreeAmountsAllowed]	When partial payments are set and this value is set to True, the customer can add a free amount between the curren partial amount and the total outstanding amount. If left empty this value is set to False.
[*PartialPaymentSchema]	In case of a partial payment; a list of all the planned payments

Parameter	Description
[*PartialPercentageAmounts]	If a the payment must be paid in several parts, the percentages and/or amounts for each payment can be set here, seperated with # and must sum up to the amount entered at *Amount (ie: 150#50%#? indicates three partial payments for an amount of 150, 50% and the remaining amount by indicating ?)
[*PartialPercentages]	If a the payment must be paid in several parts, the percentages for each payment can be set here, seperated with # and must sum up to 100% (ie: 20#30#50 indicates three partial payments for 20%, 30% and 50%)
[*PartialTemplateName]	If a the payment must be paid in several parts, a template can be used with the specified name
[*PartialWholeAmountAllowed]	When partial payments are set and this value is set to True, the customer is also offered a choice to pay the total outstanding amount at once to skip all the ther partial payment. If left empty this value is set to False.
[*PartnerHeaderImage]	The url to the header image of the partner, mostly a logo
[*PartnerName]	The name of the selected parter
[*PartnerPortalUrl]	The url to the portal of this partner (ie: http://secureportal.partner.com
[*Payment]	The payment link for use in HTML (ie: http://link)
[*PaymentButton(<button text>)]	The payment link wrapped inside a button for use in HTML with a text (ie: [*PaymentButton(Click me to pay)]
[*PaymentButton]	The payment button for use in HTML with a default text. Use the [*PaymentButton(..)] function to set your own text, ie: [*PaymentButton(Pay me now!)]
[*PaymentLink]	The payment link for use in HTML (ie: http://link)
[*PaymentLinkUrl]	The url of the payment link without HTML tags (ie: http://link)
[*PaymentMethod]	The payment method that was used on a successful payment (ie: iDEAL, VISA, etc.)
[*PaymentMethodCode]	The code of the payment method that was used for a successful payment (ie: IDEAL, VISA, etc.)
[*PaymentMethodCodes]	A list of allowed payment method codes for this payment, seperated with # (ie: IDEAL#VISA#MASTERCARD#BANCONTACT)
[*PhoneNumber]	The phone number of the cient (mobile number is mandatory if an sms must be send)
[*Pos(<value>, <start>)]	Get the text from a specific value, starting at position <start>, starting from 0 (ie: [*Pos(testing, 3)] returns: ting)
[*Pos(<value>, <start>, <length>)]	Get the text from a specific value, starting at position <start>, starting from 0, with a lengt of <lengt> (ie: [*Pos(testing, 3, 2)] returns: ti)
[*PspName]	The name of the psp that was used as set by the entity (ie: Main Molly Account)
[*PspOrderId]	The order id that was given to this payment by the psp
[*PspOrderReference]	The order id that was given to this payment by the entity when starting the payment
[*PspTransactionId]	The (technical) transaction id that was used by the psp to follow the payment
[*PspType]	The psp type that was used (ie: Ingenico, Mollie, Buckaroo, Adyen, etc.)
[*Quote]	Get a quote value
[*RelativeExpirationHours]	The amount of hours AFTER the payment has been sent that indicates when a payment expires. If left empty, the default expiration is calculated using the info on the 'Payment info' tab of the entity settings.
[*RemainingPartialPaymentSchema]	In case of a partial payment; a list of all the remaining planned payments
[*Replace(<value>, <search string>, <replace string>)]	Replace the value <search string> inside a string <value> with a new value <replace string> (ie: [*Replace(testing, st, mpt)] returns: tempting)

Parameter	Description
[*Round(<value>, <decimals>)]	Round a decimal value properly to the supplied decimals (ie: [*Round(784.2587, 2)] returns: 784.26)
[*Second(<date>)]	Get the second of the supplied value (ie: [*Second(12-09-1975 16:20)] returns: 0)
[*SendDateTime]	The date and time of the moment when the payment must be sent out. If left empty, the payment is sent directly
[*SendType]	Indicates how the client will receive a request to pay. Accepted values are: mail (if a mail must be sent), sms (if an sms must be send) or nosend (if nothing must be sent)
[*StopProcessing]	Stops the processing of values, only allowed in entity mappings as output value
[*Street]	The street name
[*TimeOnly(<value>)]	Get the time only of the supplied date using the entity language code (ie: [*DateOnly(1975-09-12 16:20)] returns 16:20 when the language code is Dutch)
[*TimeOnly(<value>, <language code>)]	Get the date only of the supplied date using the supplied language code (ie: [*DateOnly(12-09-1975 16:20, en-US)] returns 4:20 PM because the language code is English)
[*TotalAmount]	In case of a partial payment; the total amount that must be paid
[*TotalAmountNumber]	In case of a partial payment; the total amount as number using the entity language code (ie: 128.50)
[*TotalAmountText]	In case of a partial payment; the total amount as text including the currency symbol using the entity language code (ie: € 128,50)
[*TotalPartialPayments]	In case of a partial payment; the total number of partial payments
[*Trim(<value>)]	Trim the leading and trailing spaces of a string
[*UniqueId]	Get a unique value (ie: 85dd4403)
[*UserFirstName]	The first name of the user that created the item (if created manually)
[*UserLastName]	The last name of the user that created the item (if created manually)
[*Username]	The full name of the user that created the item (if created manually)
[*VariableAmounts]	Variable amounts that can be chosen by the user (ie: if the user can choose between 5 or 10 or a range between 5 and 15 then the value would be: 5#10#range:5:15
[*WholeAmountAllowed]	In case of a partial payment; 'true' if a the (remaining) amount can be paid in whole by the customer
[*Year(<date>)]	Get the year of the supplied value (ie: [*Year(12-09-1975)] returns: 1975
[*ZipCode]	The zip code of the address