THEPAY payment gate implementation manual

Content

Principle of gate workflow 3

Steps for on-line payment 3

Steps for off-line payment 3

Component for implementation 3

PHP API Merchant components 3

Class TpMerchantConfig 3

Class TpMerchantHelper 4

Class TpDivMerchantHelper 4

Class TpRadioMerchantHelper 4

Class TpButtonMerchantHelper 4

Class TpPayment 4

Class TpEetDph 4

Class TpReturnedPayment 4

Component usage 4

Display of payment button 4

Customer's return - accepting the payment 5

Additional payment state validation 5

Usage of component with TpRadioMerchantHelper 6

Rendering radio buttons 6

Appearance adjustments and styling 6

Processing payment 6

Customer return - handling payment result 7

Test payment gate 7

API parameters 7

Payment parameters 7

Signature generating 9

Parameters for rendering payment buttons 9
Integration with EET 10
Return from gate to merchant's website 10
Gate call - creating payment 11
Online payment 11
Offline payment 12
Card operations 12
Charging money 12
Payment cancel 12
Recurring payments 13
Get card details 13
Implementation 13
Method cardDepositPaymentRequest (TpCardHelper.depositPayment) 13
Method cardStornoPaymentRequest (TpCardHelper.stornoPayment) 13
Method cardCreateRecurrentPaymentRequest (TpCardHelper.createNewRecurrentPayment). 14
Method getCardInfoRequest (TpCardHelper. getCardInfo) 14
Recurring payments (bank transfer) 15
How recurring payments works 15
Restrictions 15
Implementation 15
Method createPermanentPaymentRequest (TpPermanentPaymentHelper.createPermamentPayment) 16
Method getPermanentPaymentRequest (TpPermanentPaymentHelper.getPermamentPayment) 16
Object TpPermanentPaymentResponseMethod 16
Payment return request 17
Implementation 17
Method returnPaymentRequest (TpPaymentReturnHelper. returnPayment) 17

Principle of gate workflow

Workflow of the payment gate differs based on chosen payment method. Payment methods can be either completely online (completed instantly – card payment) or offline (customer has to confirm payment e.g. with SMS). Online methods are synchronous – customer makes payment and is returned to your web. Offline methods are asynchronous – customer will receive instructions for payment, make the payment later and your web will by informed by a script.

Steps for on-line payment

1. Customer picks goods, which he wants to pay.
2. ThePay payment component should appear on page with payment instructions.
3. Customer picks payment method and is redirected to payment gate, makes the payment.
4. After payment customer is redirected back to your web and payment data are sent to your web as request parameters.
5. If the payment method requires additional confirmation – such as mPeníze – second payment information about the final state of the payment is sent = if the second payment information didn't come, the payment wasn't confirmed by the system. User returns to Your web with payment status TpReturnedPayment::STATUS_ WAITING and confirmation notifications is send with TpReturnedPayment:: STATUS_OK state.

Steps for off-line payment

1. Customer picks goods, which he wants to pay.
2. ThePay payment component should appear on page with payment instructions.
3. Customer picks payment method and is redirected to payment gate, makes the payment.
4. Customer makes the payment. He can do it instantly, or later.
5. When we receives the payment, the gate sends HTTP message on address of your choice, which contains the payment info.

You can pick payment methods, which will be available to the customers, in admin interface. If you use only online methods, you don't need to implement offline payments processing and vice versa.

You only have to implement 2 things to the application:

- Appearance of the payment button.
- Processing the payment after customer paid.

Both tasks are done by component designed by us. If you don't want or cannot use it (e.g. you don't use PHP), you can implement this functionality on your own.

Component for implementation

PHP API Merchant components

Payment component support all operations and APIs provided by ThePay gate.

Class TpMerchantConfig

Serves for saving configuration of merchant's access to THEPAY system.

Attribute **merchantId** contains numeral ID of merchant in the system.

Attribute **accountId** contains numeral ID of trade inside merchant's account.

Attribute **password** contains password for authentication of merchant during communication with the system. This password should be different than the one used for administration.

All these parameters are automatically assign to You by ThePay. Id's are immutable, password can be changed on request.

Class TpMerchantHelper

Abstract class encapsulating most methods necessary to create new payment request. From this class are other specific classes generating specific ways to present payments to the customer.

Class TpDivMerchantHelper

Default method for rendering payment button. Downloads payment button code from ThePay gate by AJAX request and insert it into page in DIV element. Optionally default CSS styles can be switch off.

Class TpRadioMerchantHelper

Alternative method which renders payment button not as set of buttons, but as several radio button choices. Supports integration with another payment methods used in application.

Class TpButtonMerchantHelper

Used to show simple „donate" button, clicking which will redirect you to THEPAY server, where you can choose payment method. This class can be used either for rendering graphical button, or only for creating URL address which you can use as you need.

Class TpPayment

Contains information about the payment. Through method „getSignature()" it is possible to generate checksum for this payment (parameter „signature" when calling THEPAY gate).

Class TpEetDph

It represents the distribution of the total amount of the payment to the partial sums of VAT rates for the EET system. Attributes naming corresponds to official naming used by EET. An instance of this class is passed to the TpPayment instance by the setEetDph () method.

Class TpReturnedPayment

This class is used to verification and processing returning request from THEPAY system. Class automatically checks monitoring count of parameters and in case of false monitoring count initiates the exception TpInvalidSignatureException.

Component usage

Display of payment button

```
$p = new TpPayment(new TpMerchantConfig());
$p->setValue(499); // goods price
$p->setDescription("Description of purchase.");// payment description - optional
```

```
$p->setMerchantData("1662");   // merchant's data, such as ID of the order - optional
$p->setReturnUrl("http://www.adresa-eshopu.cz/return-url/");// URL on which will the customer be returned after completing the payment, or where payment notification will be send
// other optional settings, see class TpPayment
$hlp = new TpDivMerchantHelper($p);
echo $hlp->render();// display of component with payment button
```

Component is rendered into page as div HTML element generated by JavaScript. It is possible to style it by CSS as you need. Default CSS can be completely turn off if you need it - see methods of class TpDivMerchantHelper.

Customer's return - accepting the payment

```
$p = new TpReturnedPayment(new TpMerchantConfig());
if($p->verifySignature()){
	// payment is valid – contains all necessary data and signature is valid
	if($p->getStatus() == TpReturnedPayment:: STATUS_OK){
		// payment is paid  and you can process it
	} else if($p->getStatus() == TpReturnedPayment::STATUS_UNDERPAID){
		// payment is paid but customer sent lower amount than payment's amount
	} else if($p->getStatus() == TpReturnedPayment::STATUS_CARD_DEPOSIT){
		// amount was successfully blocked on customer's account - only for card transactions
	} else if($p->getStatus() == TpReturnedPayment::STATUS_NOT_PAID){
		// customer didn't pay
	} else if($p->getStatus() == TpReturnedPayment::STATUS_CANCELED){
		// customer canceled the payment
	} else if($p->getStatus() == TpReturnedPayment::STATUS_ERROR){
		// some error during payment
	} else if($p->getStatus() == TpReturnedPayment::STATUS_WAITING){
		// payment was successful, but waiting for additional confirmation from payment system. Notification will be sent do you after confirmation arrives.
	}
} else {
	// invalid request – error occurred, possibly a fraud etc.
}
```

Same code can be used both for customer return after online payment and for handling notifications from offline payment methods.
For offline method notifications page must return HTTP code 200 or 302, if another code is returned, notifications is considered as not delivered and will be send again after some time.
Notifications are sent by HTTP method HEAD, redirect for online methods is standard GET request.

Additional payment state validation

You can validate state of payment in ThePay systems using API method **getPaymentState** by payment's ID (return parameter *paymentId*).

By this check you will improve security of whole process.

If you want to be even more secure, use API method **getPayment**, which will return all available information about payment, and check if value *merchantData* matches value expected by your system.

You can fined more information about subject in API documentation in **Data Api v1.3.doc**

Usage of component with TpRadioMerchantHelper

Usage of radio button component is divided into two separate steps - rendering buttons and processing payment (redirect to ThePay gate). So you must implement it on two places. For rendering radio buttons you don't need to know payment information (value, merchant data atp.)

This component's variant also allows you to combine ThePay with your own payment methods.

Rendering radio buttons

```
$hlp = new TpRadioMerchantHelper($config, 'method_select', '3');
echo $hlp->renderRadio();
```

Atribut **name** contains original name of radio button with payment method choice.
Atribut **value** contains value of original radio button with ThePay payment method.
If you have in your system e.g. these payment methods: payment on delivery with id 1, cash payment with id 2 and ThePay with id 3 and radio button with payment method selection has name=" method_select", then you will use example code above.
If you use only ThePay and don't have your own payment methods, leave attributes empty.

No information about payment are needed in this phase.

Appearance adjustments and styling

Attribute **showIcon** (boolean) shows or hide payment method logos (default on).
Attribute **appendCode** (string) contains HTML code which will be appended at the end of div with each payment method.
If you define JavaScript function **tp_after_load_callback** it will be called after rendering payment component (you can modify component's code).

Component does not have any default styling (it is only HTML), all styling is in your hands.

Processing payment

After processing of order in your system (e.g. saving e-shop order) you must call helper methods which will handle payment (redirect user to payment gate).

```
if($hlp->isTpMethodChosen()){
        $p = new TpPayment($config);
        $hlp->redirectOfflinePayment($p);
```

```
        echo $hlp->showPaymentInstructions($p);
}
```

Method **isTpMethodChosen** returns true it ThePay payment method was chosen in previous step. Method **redirectOfflinePayment** and **showPaymentInstructions** redirect user to payment gate or show him payment instructions.

Both methods do their job only if user select corresponding payment method. Condition if($hlp->isTpMethodChosen()) is not necessary.

Customer return - handling payment result

This part is the same as for normal payment component with buttons.

Test payment gate

For testing of functionality of your implementation, you can use our testing payment gate. Test gate works the same way as the live gate, only instead of redirecting to the payment system displays parameters of the payment and button to simulate payment.

There is special form available for simulating notifications send after offline payment is paid.

Gate URL:	https://www.thepay.cz/demo-gate/
Merchant ID:	1
Account ID:	3
Password:	my$up3rsecr3tp4$$word
Data API password:	my$up3rsecr3tp4$$word
Notifications test form	https://www.thepay.cz/demo-gate/testNotif.php

API parameters

Payment parameters

Parameters used by all payment button types and direct gate calls.

Parameter	Data type	Default value	Description
merchantId	int	-	Merchant ID
accountId	int	-	Account ID
value	decimal	-	Amount paid
currency	string	CZK	Currency
description	string	""	Description specified when submitting the payment.
merchantData	string	""	Optional data specified during sending

<table>
<tr><td></td><td></td><td></td><td>of request</td></tr>
<tr><td>customerEmail</td><td>string</td><td>null</td><td>Customer's e-mail address. If set we will send to it info about payment creation and confirmation after payment it paid.</td></tr>
<tr><td rowspan="2">customerData</td><td>string</td><td>null</td><td>Customer’s billing address as JSON encoded object. Used for 3D Secure 2.0 card payment authorization. Data should match with the cardholder. It is optional, but missing data can increase number of declined card payments.</td></tr>
<tr><td colspan="3">Object parameters (all optional)
<table><tr><th>Parameter</th><th>Description</th></tr><tr><td>full_name</td><td>Name and surname of customer</td></tr><tr><td>country</td><td>Country in ISO 3166-1 Alpha-2 format</td></tr><tr><td>city</td><td>City</td></tr><tr><td>postcode</td><td>Postal code</td></tr><tr><td>street</td><td>Street and street number</td></tr><tr><td>email</td><td>E-mail address. If omitted, customerEmail will be used instead</td></tr></table></td></tr>
<tr><td>returnUrl</td><td>string</td><td>-</td><td>URL address for redirecting user after payment and sending notifications</td></tr>
<tr><td>backToEshopUrl</td><td>string</td><td>URL of merchant's account</td><td>URL for link "Zpět do eshopu" on page with payment instructions for offline payment methods</td></tr>
<tr><td>merchantSpecificSymbol</td><td>string</td><td>-</td><td>Specific symbol provided by merchant (don't mistake for specificSymbol)</td></tr>
<tr><td>eetZaklNepodlDph</td><td>decimal</td><td>-</td><td>EET - Total amount of exempt from VAT, other payments in CZK</td></tr>
<tr><td>eetZaklDan1</td><td>decimal</td><td>-</td><td>EET - Total tax base with basic VAT rate in CZK</td></tr>
<tr><td>eetDan1</td><td>decimal</td><td>-</td><td>EET - Total VAT with basic rate in CZK</td></tr>
<tr><td>eetZaklDan2</td><td>decimal
decimal</td><td>-</td><td>EET - Total tax base with the first reduced VAT rate in CZK</td></tr>
<tr><td>eetDan2</td><td>decimal</td><td>-</td><td>EET - Total VAT with the first reduced rate in CZK</td></tr>
<tr><td>eetZaklDan3</td><td>decimal</td><td>-</td><td>EET - Total tax base with the second reduced VAT rate in CZK</td></tr>
<tr><td>eetDan3</td><td>decimal</td><td>-</td><td>EET - Total VAT with the second reduced rate in CZK</td></tr>
<tr><td>eetCestSluz</td><td>decimal</td><td>-</td><td>EET - Total VAT amount for travel service in CZK</td></tr>
<tr><td>eetPouzitZboz1</td><td>decimal</td><td>-</td><td>EET - The total VAT amount for the sale of second-hand goods at the basic rate in CZK</td></tr>
<tr><td>eetPouzitZboz2</td><td>decimal</td><td>-</td><td>EET - The total VAT amount for the sale of second-hand goods with the first reduced rate in CZK</td></tr>
<tr><td>eetPouzitZboz3</td><td>decimal</td><td>-</td><td>EET - The total VAT amount for the sale of second-hand goods with the second reduced rate in CZK</td></tr>
</table>

eetUrcenoCerpZuct	decimal	-	EET - Total amount of payments for subsequent drawdown or settlement in CZK
eetCerpZuct	decimal	-	EET - The total amount of payments that are subsequently drawn or settled in CZK
signature	string	-	MD5 hash of all sent parameters + merchant's password

In case of direct call, parameters are part of request (both GET and POST woks).

In case of PHP component usage, parameters are taken from TpMerchantConfig class (merchantId, accountId and password) and TpPayment instance (value, currency, description, merchantData, returnUrl). Signature is calculated by component.

Signature generating

Signature is MD5 hash of all not null (empty parameters are not included) payment parameters, except of customerData (this parameter is not included in the signature), and account password in format for GET request. Parameters are NOT urlencoded.

Order of parameters must correspond to order in which they are listed in this manual.

All amounts (decimal numbers) are in format with two decimal points and dot as decimal separator (\d+.\d\d). Boolean parameters are represented as 0 or 1.

Example:

If you have this values: merchantId = 1, accountId = 1, value = 999, description = desc, returnUrl = http://www.nejaky-obchod.cz/return/, password = 12345

then signature is generated this way: signature = md5("merchantId=1&accountId=1&value=999&description= desc &returnUrl=http://www.nejaky-obchod.cz/return/&password=12345")

Parameters for rendering payment buttons

PHP component generates URL of JavaScript (see TpDivMerchantHelper) which downloads payment component code via AJAX and include it into page.

Special parameters:

Parameter	Data type	Default value	Description
disableButtonCss	boolean	false	Switch off CSS styling of payment component. You can use your own style for it.
disablePopupCss	boolean	false	Switch off CSS for popup with instructions for offline payment method.

Paraeters for selecting payment method on ThePay gate

If you want to have payment method selection on ThePay gate, not in your applications, you have to send parameter **methodSelectionAllowed** in request. Value of parameter is not important (it can be empty), but it has to be present in request. TpButtonMerchantHelper class use this parameter automatically.

This parameter **is not part of the signature**.

Integration with EET

ThePay payment gate allows you to automatically connect to EET. For this functionality to work, merchant have to upload a communication certificate in his administration, set up his account and activate the EET communication.

If merchant sells goods with same VAT rate in one e-shop (one accountId), he is using VAT calculation from top (calculates the value and base of VAT from final amount) and does not round total amount, it is not necessary to change implementation and send data for EET. The payment gateway calculates everything itself using VAT rate set in account setting by merchant.

In other cases, it is necessary to pass the breakdown of the total amount to partial sums for VAT rates. The amounts are passed as payment parameters, see Pament parameters. If you are using an implementation component, the TpEetDph class is there for this purpose. The partial amounts passed by this class will be sent to the EET system unchanged. The payment gateway only verifies the basic consistence of the data. The merchant is responsible for its correctness.

Return from gate to merchant's website

After successful payment is customer redirected back at the merchant's website specified in returnUrl parameter of gate call.

Parameters submitted via GET method will be shown in following table

Parameter	Data type	Description
value	decimal	Amount paid
currency	string	Currency
methodId	int	ID of payment method used.
description	string	Description specified when submitting the payment.
merchantData	string	Optional data specified during sending of request
status	enum	Status of the payment, possible values (constants in TpReturnedPayment class): STATUS_OK (2) – successfully paid STATUS_CANCELED (3) – payment cancelled by customer STATUS_ERROR (4) – error during payment. In this case please contact our tech support. STATUS_UNDERPAID (6) - customer paid, but he send less money than "value" STATUS_WAITING (7) - payment was successful, but waiting for additional confirmation from payment system. STATUS_CARD_DEPOSIT (9) - amount was successfully blocked on customer's account - only for card transactions
paymentId	int	ID number of payment generated by system THEPAY. This ID is

		always unique in the whole system
ipRating	int	**DEPRECATED**
isOffline	0|1	If it's offline payment method(isOffline=1) or online payment method (isOffline=0).
needConfirm	0|1	**DEPRECATED**
isConfirm	1	**DEPRECATED**
customerAccountNumber	string	Customer's account number. Only for payment methods which use account number and provide it by their API. This feature is available only on request. Please contact us if you need it.
customerAccountName	string	Customer's account name (usually name and surname). Only for payment methods which use account name and provide it by their API. This feature is available only on request. Please contact us if you need it.
signature	string	MD5 hash of all sent parameters + merchant's password

Parameters *needConfirm* and *isConfirm* are deprecated and were replaced by status STATUS_WAITING (7).

In case of recurrent payment notifications are added these parameters:

Parameter	Data type	Description
specificSymbol	string	Specific symbol entered by customer into bank transfer

Parameter is part of signature only if payment is recurrent and specific symbol was set by customer.

In case of card payment are added these parameters:

Parameter	Data type	Default value	Description
deposit	0|1	1	Indicate if amount should be charged from customer's account immediately (1) or only blocked and charged later by calling API (0).
isRecurring	0|1	0	Indicate if payment is recurring (another payments can be charged from customers account without his participation by API calls if payment is recurrent).

Parameters are part of signature only if there are not empty.

Gate call - creating payment

Payment processing is done after GET/POST request on https://www.thepay.cz/gate/ with previously described parameters. All available component variants end up with call to this URL, which will create payment and handle payment by selected payment method.

Online payment

After payment through online payment method customer is redirected back to merchant's site on URL given in returnUrl parameter.

Gate take these parameters sent via GET or POST

Parameter	Data type	Default value	Description
merchantId	int	-	Merchant ID
accountId	int	-	Account ID
value	decimal	-	Amount paid
currency	string	CZK	Currency
description	string	""	Description specified when submitting the payment.
merchantData	string	""	Optional data specified during sending of request
returnUrl	string	-	URL address for redirecting user after payment and sending notifications
backToEshopUrl	string	URL of merchant's account	URL for link "Zpět do eshopu" on page with payment instructions for offline payment methods
methodId	int	-	ID of chosen payment method.
signature	string	-	MD5 hash of all sent parameters + merchant's password

Offline payment

In case of offline payment, instructions for payment are shown, along with button „Continue“, which leads back to the merchant's website.

Offline payment parameters are the same as with online payment.

If payment is paid, notification is send to URL specified in returnUrl, so merchant's system can process the payment. Notification contains same parameters as user redirect request in case of online payment.

Card operations

Card payment support several additional operations depending on "deposit" and "isReccuring" parameters.

For all these operations payments are identified by value of "merchantData", so if you want to use them, you must use unique values of "merchantData" for each payment.

All operations could be performed by TpCardHelper class.

Charging money

If you create payment with deposit=0, amount was not charged, but only blocked on customers account. Charging is made by calling this operation. If charge is not called, money will be automatically unblocked by bank after some time.

Payment cancel

If you create payment with deposit=0, you can cancel payment and release money blocked on customer's account.

Recurring payments

If you create payment with isRecurring=1, it is possible to create new payment without participation of customer anytime later. Copy of original payment with new amount and merchantData is created (attention to the "deposit", which is also preserved from original payment).

Get card details

You can get additional information about customer's payment card – masked card number, card brand, issuing bank, country and card type. You can for example display this info to customer if you are using recurrent payments, so he will know which card is used. Or you can use it for fraud detection.

Implementation

These operations are implemented as web service (SOAP).
Production WSDL is here https://www.thepay.cz/gate/api/gate-api.wsdl, testing WSDL is here https://www.thepay.cz/demo-gate/api/gate-api-demo.wsdl
TpCardHelper class should be used for implementation.

Method cardDepositPaymentRequest (TpCardHelper.depositPayment)

Charge money blocked on customer's account by payment with parameter deposit=0. Just one payment with provided merchantData, deposit=0 and in STATE_CARD_DEPOSIT state must exist on merchant's account. Blocked money is released by bank after some time, so it will not work for old payments.

Request

Parameter	Data type	Description
merchantId	int	Merchant ID
accountId	int	Account ID
merchantData	string	Merchant data of payment
signature	string	MD5 hash of all sent parameters + merchant's password

Response

Parameter	Data type	Description
status	boolean	Result of the operation. TRUE = payment was successfully charger, FALSE = there was an error
errorDescription	string	Description of error if status = FALSE

Method cardStornoPaymentRequest (TpCardHelper.stornoPayment)

Cancel payment - unblock money previously blocked on customer's account by payment with parameter deposit=0. Just one payment with provided merchantData, deposit=0 and in STATE_CARD_DEPOSIT state must exists on merchant's account. Blocked money are released by bank after some time, so it will not work for old payments.

Request

Parameter	Data type	Description
merchantId	int	Merchant ID
accountId	int	Account ID
merchantData	string	Merchant data of payment
signature	string	MD5 hash of all sent parameters + merchant's password

Response

Parameter	Data type	Description
status	boolean	Result of the operation. TRUE = payment was successfully cancelled, FALSE = there was an error
errorDescription	string	Description of error if status = FALSE

Method cardCreateRecurrentPaymentRequest (TpCardHelper.createNewRecurrentPayment)

Create new payment based on existing paid payment with parameter isRecurring=1. Just one payment with provided merchantData, isRecurring=1 and in STATE_PAID state must exist on merchant's account.

Request

Parameter	Data type	Description
merchantId	int	Merchant ID
accountId	int	Account ID
merchantData	string	Merchant data of first payment authorized by customer
newMerchantData	string	Merchant data of newly created payment
value	decimal	Amount of ne payment
signature	string	MD5 hash of all sent parameters + merchant's password

Response

Parameter	Data type	Description
status	boolean	Result of the operation. TRUE = payment was successfully created and charged, FALSE = there was an error
errorDescription	string	Description of error if status = FALSE

Method getCardInfoRequest (TpCardHelper. getCardInfo)

Get additional information about customer's payment card. Merchant must have this feature enabled and the payment must be paid by the card.

Request

Parameter	Data	Description

	type	
merchantId	int	Merchant ID
accountId	int	Account ID
paymentId	string	Payment ID
signature	string	MD5 hash of all sent parameters + merchant's password

Response

Parameter	Data type	Description
status	boolean	Result of the operation. TRUE = everything if OK, FALSE = there was an error
errorDescription	string	Description of error if status = FALSE
cardNumberMasked	string	Masked card number in format 123456******1234
cardBrand	string	Name of the card association, e.g. MC, VISA
countryCode	string	Name of issuing bank
bankName	string	Country code ISO 3166-1 alpha-2
cardType	string	Card type, e.g. DEBIT, CREDIT
cardLevel	string	Card level, e.g. BUSINESS, CLASSIC, PREPAID

Recurring payments (bank transfer)

How recurring payments works

Normal payment is only one-off, and its variable symbol can't be used again. It is necessary to show payment component to customer again for another payment (and customer must select payment method and new payment is created).

Recurrent payment allows to have one unique variable symbol under which customer can send money several times. And all received payments will be transferred to merchant.

Recurrent payments are useful only for some specific cases - for example for paying subscriptions, regular fees, repayment of loans etc.

Restrictions

- recurring payments are available only for bank transfer, not for other payment methods
- only one payment per day and variable symbol is allowed (more payments with same variable symbol in one day will not work)

Implementation

Functionality is provided as web services which allows you to create permanent payments and get payment instructions for existing payment (which you give to customer).
Production WSDL is here https://www.thepay.cz/gate/api/gate-api.wsdl, testing WSDL is here https://www.thepay.cz/demo-gate/api/gate-api-demo.wsdl
Use TpPermanentPaymentHelper for implementation.

Method createPermanentPaymentRequest (TpPermanentPaymentHelper.createPermamentPayment)

Create new recurrent payment and return information for payment (variable symbol and account numbers). If recurrent payment with same value of merchantData already exists for account, new payment is not created, but data for existing payments are returned.

Request (Object TpPermanentPayment)

Parameter	Data type	Description
config	TpMerchantConfig	Merchant's configuration
merchantData	string	Unique identifier of transaction. Is send to merchant as value of merchantData in notification
description	string	Optional description for merchant. Not displayed to customer
returnUrl	string	URL address for sending notifications about payments
signature	string	MD5 hash of all sent parameters + merchant's password

Response

Same as for getPermanentPaymentRequest

Method getPermanentPaymentRequest (TpPermanentPaymentHelper.getPermamentPayment)

Return payment information for existing recurrent payment. New payment is not created, empty result is returned if recurrent payment with given merchantData not exists.

Request

Parameter	Data type	Description
merchantId	int	Merchant ID
accountId	int	Account ID
merchantData	string	Merchant data of permanent payment
signature	string	MD5 hash of all sent parameters + merchant's password

Response (class TpPermanentPaymentResponse)

Parameter	Data type	Description
status	boolean	State of request. True if given parameters was correct and payment was created or found. False if some error occurred
errorDescription	string	Error description. Empty if status=true
paymentMethods	Array of TpPermanentPaymentResponseMethod	Array of payment instructions for available payment methods. Show these information to customer

Object TpPermanentPaymentResponseMethod

Represents payment information for one payment method.

Parameter	Data type	Description
methodId	integer	Unique payment method identifier
methodName	string	Payment method name
url	string	URL of this bank's internet banking
accountNumber	string	Account number (complete with bank number and prefix)
vs	integer	Variable symbol

Payment return request

This feature is designed to return a paid payment back to the customer if it is not possible to fulfill the order due to the problem on merchant's side (inaccessible goods, etc.) or the customer wants to cancel the payment. Only full amount of payment can be returned.

There are two ways to request a payment refund. The first option is to manually enter a request in the payment details in the client section. The second option is to create it by API request.

The request may be rejected if the merchant does not have a sufficient balance in the account.

Merchant is informed about the result of processing the request by e-mail.

Implementation

Functionality is provided as web service.
Production WSDL is here https://www.thepay.cz/gate/api/gate-api.wsdl, testing WSDL here https://www.thepay.cz/demo-gate/api/gate-api-demo.wsdl
Use class TpPaymentReturnHelper for implementation.

Method returnPaymentRequest (TpPaymentReturnHelper. returnPayment)

Creates a refund request for payment identified by paymentId (payment ID in the payment gateway), optionally giving the reason for a refund. Payment can be refunded only if its status allows it (paid or partially paid payment) and the refund request for this payment has not yet been created.

Request

Parameter	Data type	Description
merchantId	int	Merchant ID
accountId	int	Account ID
paymentId	Int	Pyment ID
reason	string	Optional reason of storno
signature	string	MD5 hash of all sent parameters + merchant's password

Response

Parameter	Data type	Description
status	boolean	Result of the operation. TRUE = payment was successfully created and charged, FALSE = there was an error
errorDescription	string	Description of error if status = FALSE