Skrill PHP Library Guide

For credit/debit card and alternative payments

How to integrate to the Skrill Payment Platform using the Skrill PHP library

www.skrill.com

Version 1.6

Skrill Limited, 25 Canada Square, Canary Wharf, London, E14 5LQ, UK

Copyright

Skrill Ltd.

Registered office: Skrill Limited, 25 Canada Square, Canary Wharf, London, E14 5LQ, UK.

Version Control Table

Date	Version	Description
November 2013	1.0	Guide Created
January 2014	1.1	Examples updated, alternative payments and WPF included.
March 2014	1.2	Inclusion of Sandbox URLs.
April 2014	1.3	Updates to requirements.
June 2014	1.4	All API requests and callbacks must use the HTTPS standard. Details added about handling the Skrill response sent to your *responseurl* page.
August 2014	1.5	Removal of reversal request. New examples of payment scenarios.
September 2014	1.6	Removal of payment methods currently not available: Boleto, WebMoney, Neosurf, Lobanet and Multibanco

Publication number: *PSP-PHP-REL-9/9/14*

Contents

1. ABOUT THIS GUIDE

1.1. Objectives and target audience

This guide describes how integrate to the Skrill Payment Platform using the Skrill PHP library. It is intended for merchants who want to integrate with Skrill using PHP. If you are using this method, developer-level knowledge of object-oriented programming using PHP is essential.

1.2. Related documentation

You should use this guide together with the additional Skrill documents described below.

Table 1-1: Other Guides

Guide	Description
Getting Started Guide	Guide setting up your Skrill account, documentation required and an overview of the different product options and services for taking payments through Skrill.
Skrill Integration Guide - card payments	Describes how to connect to the Skrill Payment Platform using JSON.
Merchant Portal Guide	Describes how to manage your account using the Merchant Portal.
Alternative Payment Methods Guide	Guide to processing alternative payment methods with Skrill.
Skrill Web Payment Frontend Guide	Guide to using the Skrill hosted payment page.

1.3. Conventions used in this guide

The table below lists some of the conventions used in this guide.

Table 1-2: List of conventions

Convention	Description
[Directory]	Indicates a generic directory (e.g., C drive), which may be set up according to the client's requirements.
Reference	Indicates a reference to another section in this guide. For example, refer to User Administration on page 34.
`Code example`	Used to illustrate example code, functions and commands.
File path	Used to indicate a file path or folder structure.
Glossary	Glossary term
Menu1 > Menu option2 >	Indicates a menu path.

2. INTRODUCTION

The Skrill PHP library provides a method for connecting to the Skrill Payment Platform for payment processing. The PHP library is a client library providing PHP classes that enable you to send and receive JSON-RPC requests.

2.1. Payment processing flows

2.1.1. Using the PHP library with another integration method

The examples below show some additional card payment flows, which are relevant if you are either not PCI compliant or are looking for a quick and simple way to collect customer card details. In this case, you can use Skrill.js or the Web Payment Frontend (WPF) for the initial register request and then use the PHP library for other payment requests, such as capture, refund, cancel and credit payments.

2.1.1.1. PHP library plus Skrill.js

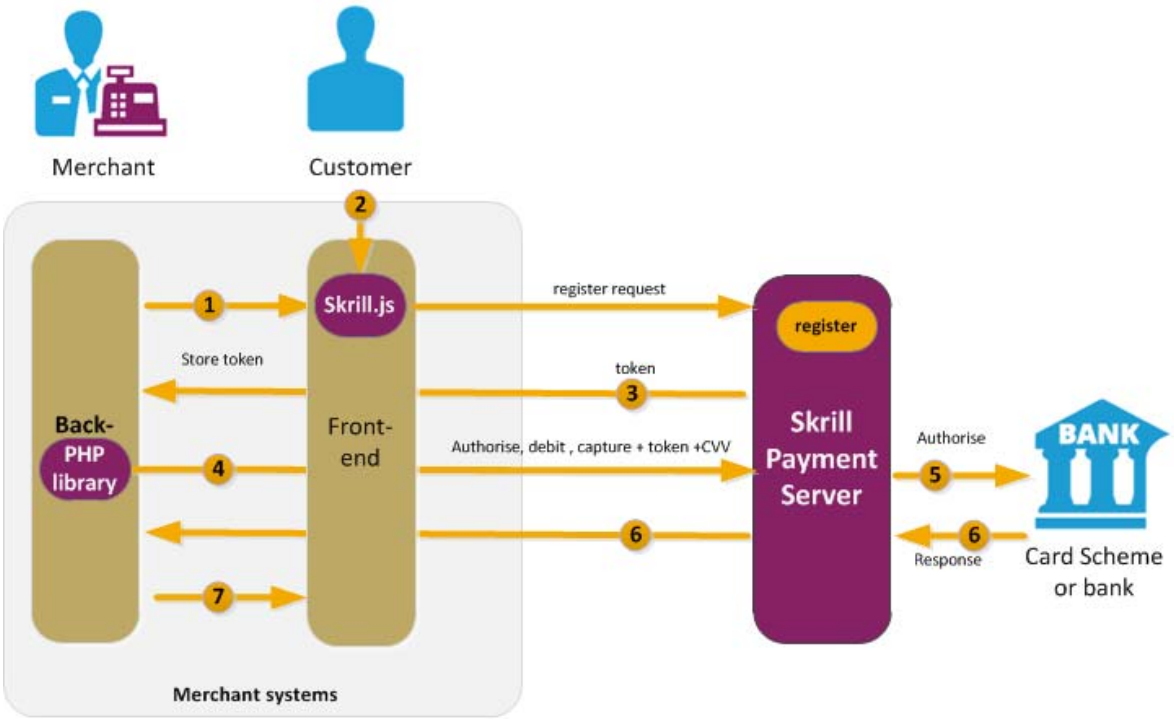

Figure 2-1: Using the Skrill PHP library with Skrill.js (non-recurring payment)

1. When the customer is ready to check out their order on your website, you can provide an HTML payment form on your front-end servers or other means where customers can enter their payment details.
2. The customer enters their payment details and clicks a **Submit** or **Pay** button. Your web server passes this information through to Skrill.js, which sends it to the Skrill Payment Platform.

Note: Card details must not be stored on your systems, unless you are PCI compliant.

3. The Skrill Payment Platform stores the card details and returns a payment token, which you can store for future use.
4. Your back-end systems send an authorisation, debit or credit request using the PHP library.
5. The Skrill Payment Platform requests authorisation for the payment from the card scheme or bank. (Payment is requested via your Acquirer.)

Note: At this stage, Cardholder authentication such as 3D Secure may be requested.

6. A Successful or Failed transaction response is returned to your systems, together with the transaction details. You can store details such as the payment token (***register*** requests) or unique reference ID in your back-end systems for use with future transactions.
7. Your systems can use the transaction response to provide feedback to the customer or initiate any other required action (for example, return the customer to a payment successful page or request an alternative payment method for a failed transaction).

2.1.2. Using the PHP library on its own

The figure below provides an overview of the payment process using the Skrill PHP library on its own, without any other integration method. The figure describes an initial Pre-authorisation, Debit or Register request using the Skrill PHP library. Note that you must be PCI compliant if you are collecting customer card details.

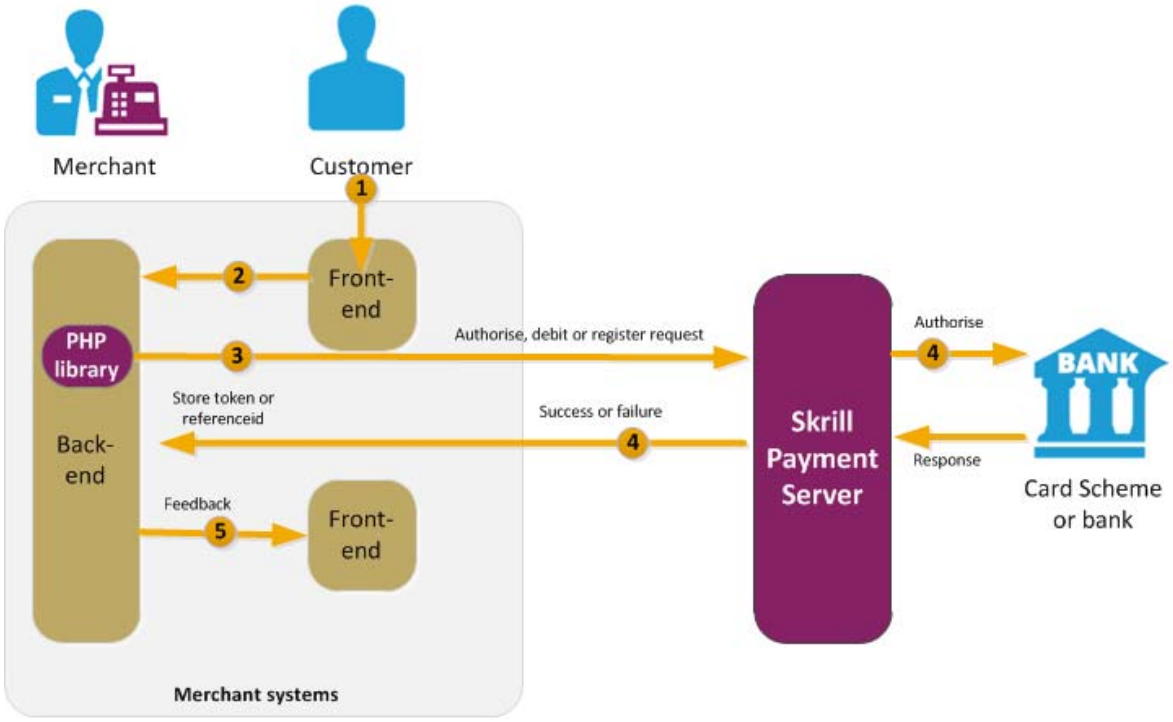

Figure 2-2: Preauthorisation, debit or register flow using the Skrill PHP library

1. When the customer is ready to check out their order on your website, you can provide an HTML payment form on your front-end servers or other means where customers can enter their payment details.
2. The customer enters their payment details and clicks a **Submit** or **Pay** button. Your web servers pass this information through to your PHP application, which will use the Skrill PHP library to send it to the Skrill Payment Platform in JSON format.
3. The Skrill Payment Platform requests authorisation for the payment from the card scheme or bank. (Payment is requested via your Acquirer.)

Note: At this stage, Cardholder authentication such as 3D Secure may be requested.

---

4. A Successful or Failed transaction response is returned to your systems, together with the transaction details. You can store details such as the payment token (***register*** requests) or unique reference ID in your back-end systems for use with future transactions.
5. Your systems can use the transaction response to provide feedback to the customer or initiate any other required action (for example, return the customer to a payment successful page or request an alternative payment method for a failed transaction).

2.1.2.1. Cardholder authentication

Skrill supports cardholder authentication schemes, such as Verified by Visa and MasterCard 3D Secure. To support cardholder authentication, we recommend that you use the PHP library in combination with Skrill.js. For more information, see the ***Skrill.js Guide***.

2.1.3. Security

A direct server-to-server connection to the Skrill Payment Platform over a secure, encrypted SSL connection ensures that transaction details are always kept secure. The HTTPS standard provides website authentication and protects against man-in-the-middle attacks. Since all communication between Skrill and your back-end servers is encrypted, this protects against eavesdropping and tampering with the contents of the JSON request.

In addition, Skrill only accepts transaction requests containing a valid merchant ID and channel.

Note: To ensure security, all production API transaction requests or callback URLs submitted to the Skrill Payment Platform must use the secure HTTPS standard.

2.2. Payment scenarios

This section provides examples of typical payment scenarios, including the possible steps in a typical payment. Two examples are provided: one for ***Immediate payment capture*** and one for ***Delayed payment capture*** .

2.2.1. Immediate payment capture

This scenario is suitable for merchants who take full payment immediately from the customer. Shipment of goods or provision of service is only provided once payment has been confirmed.

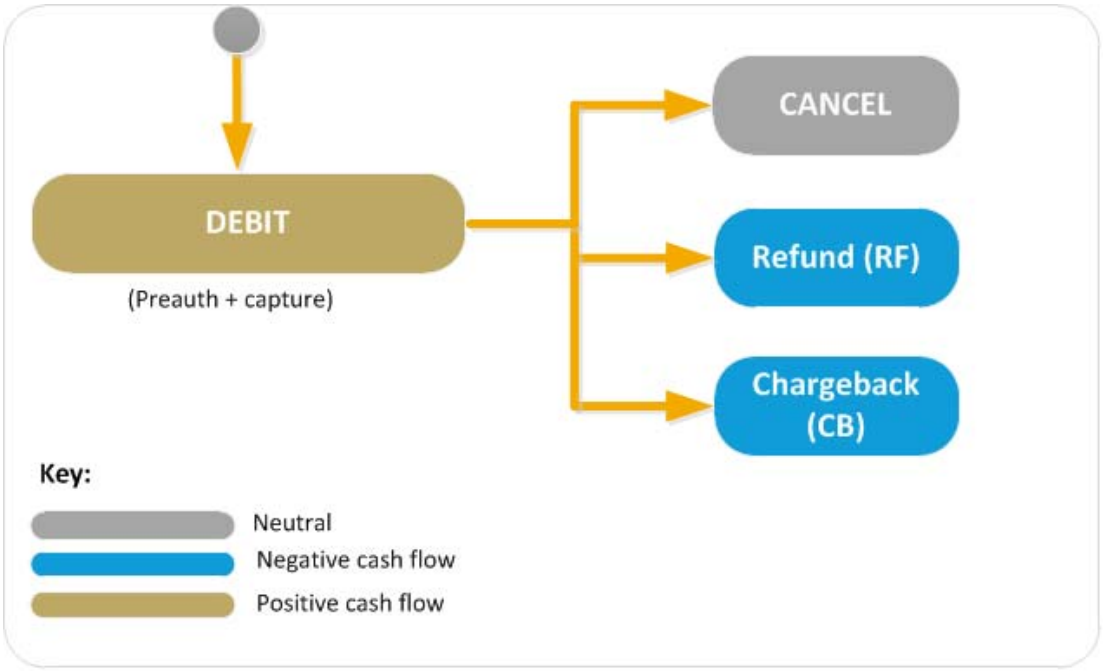

Figure 2-3: Immediate payment capture scenario

- The merchant makes a ***debit*** request, which includes the ***preauthorisation*** and ***capture*** methods in one request. Authorisation is requested from the customer's bank and the funds are immediately captured.
- Once the transaction has been made, the merchant has a short window when they can ***cancel*** the transaction (before the cut-off period when the payment has been deducted from the customer's account).
- After the cut-off period, once the funds have been deducted from the customer's account, the merchant can initiate a ***refund***.
- In the event of a dispute, the customer can contact their bank to request a ***chargeback***, in which case part or all of the funds may be deducted from the merchant's account.

For more information on the different payment request options available via the Skrill Payment Platform, see ***Supported API Call Methods, on page 10.***

2.2.2. Delayed payment capture

This scenario is suitable for merchants who request an initial authorisation for a payment or part payment, but only take full payment after the service has been delivered to the customer. For example, a car hire or holiday booking merchant, where an initial deposit may be taken and the full amount captured at a later date.

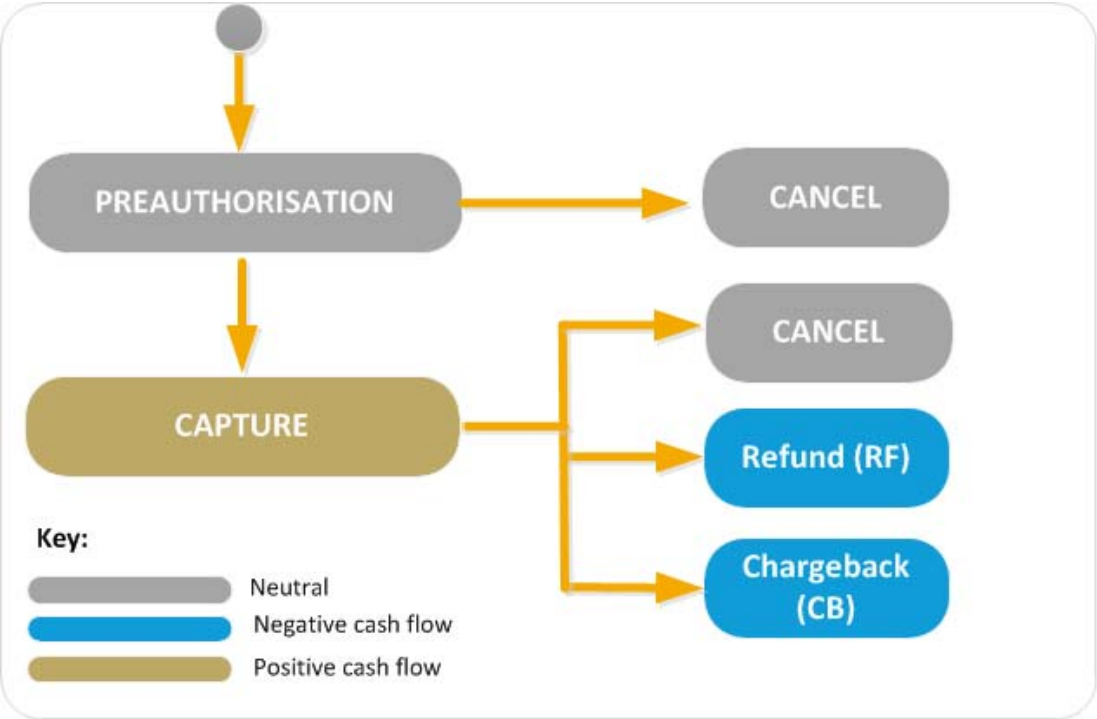

Figure 2-4: Delayed payment capture scenario

- The merchant makes a ***preauthorisation*** request. Authorisation is requested from the customer's bank and the funds are reserved on their account.
- At this stage, the merchant can cancel the ***preauthorisation***.
- When the merchant is ready, they can ***capture*** a payment. Capture can be for a partial amount (e.g., deposit) or for the full amount of the preauthorisation. Merchants can make multiple captures for partial amounts, provided that the sum of all these partial captures does not exceed the amount of the original preauthorisation.
- Once the capture has been made, the merchant has a short window when they can ***cancel*** the transaction (before the cut-off period when the payment has been deducted from the customer's account).
- After the cut-off period, once the funds have been deducted from the customer's account, the merchant can initiate a ***refund***.
- In the event of a dispute, the customer can contact their bank to request a ***chargeback***, in which case part or all of the funds may be deducted from the merchant's account.

For more information on the different payment request options available via the Skrill Payment Platform, see ***Supported API Call Methods, on page 10.***

3. USING THE PHP LIBRARY

3.1. Requirements

Ensure that the following are installed on your web server:

- PHP 5 (5.2.0)
- cURL extension should be enabled
- Copy of the Skrill PHP Library, which can be obtained from the Skrill Merchant Services Team

3.2. Including the PHP Library

To include the PHP library, add the following script to your PHP code:

```
require_once ' <path to file>/lib/SkrillPSP.php';
```

Where ***<path to file>*** is the path to where the Skrill PHP Library is installed on your web server.

3.3. Using the PHP library

Your PHP code should include the following steps:

- ***Prepare the request object***
- ***Set the Skrill API endpoint***
- ***Set the parameters to send to Skrill***
- ***View the JSON request parameters*** (optional)
- ***Send the API request***
- ***Handle the response***

These steps are described in more detail below. To see how this is implemented in working examples, see ***PHP Examples on page 23***.

Alternative payment methods work in a similar way, although some are implemented slightly differently. For details, see ***Alternative Payments on page 37***.

3.3.1. Prepare the request object

Each Skrill API request method is represented in a separate request object. The request object loads the predefined JSON structure and sets the ID member on creation of the request.

The following API request methods are supported:

Table 3-1: Supported API Call Methods

API Method	Description	PHP Request Object
cancel	Cancels a transaction. This is only possible before you have captured the payment or until the preauthorisation expires. It can be for a partial amount.	$cancel = new SkrillPsp_Cancel();
capture	Request capture of a payment that has been authorised. Capture can be for the full amount of the preauthorisation, or for a partial amount.	$capture = new SkrillPsp_Capture();
credit	Request a payment to a customer which is not linked to any existing transaction on the system.	$credit = new SkrillPsp_Credit();
debit	Request immediate authorisation of a card payment and debit of the amount from the cardholder's account.	$debit = new SkrillPsp_Debit();
preauthorisation	Request pre-authorisation for a debit or credit card transaction. A successful preauthorisation reserves the transaction amount on the customer's card. Funds are not taken from the customer's account until you invoke the ***capture*** request.	$preauth = new SkrillPsp_Preauthorization ();
refund	Request a refund on a payment that has been processed, for a partial amount or up to the full amount of the initial payment.	$refund = new SkrillPsp_Refund ();
register	Request a payment token from the Skrill payment platform.	$register = new SkrillPsp_Register ();

Note: For additional request objects available with alternative payment methods, refer to ***Alternative Payments on page 37***.

Request using a payment token

For an ***authorization***, ***credit*** or ***debit*** request on a card that is already registered on the Skrill payment platform, you can provide the payment token returned by Skrill.js as the constructor parameter.

For example:

```
$debit = new SkrillPsp_Debit("58409c4a39a94cf4bd3fe07647fca9fd");
```

Where ***("58409c4a39a94cf4bd3fe07647fca9fd ")*** is the payment token returned by Skrill.js.

Request with credit card data

Note: you need to be PCI-compliant to collect credit card data directly on your website. We recommend that you use Skrill.js to collect the customer's card details and return a payment token, for use with subsequent transactions. Alternatively, you can use the Skrill Web Payment Frontend (WPF) to process debit, credit and preauthorization requests. To connect to the WPF via the PHP library, see ***Web Payment Frontend (WPF) on page 33***.

For a ***debit*** or ***register*** request, where you do not have a Payment token, you do not need to include any constructor parameter in the request object. However, you must then include the following parameters when you set the required parameters to send to Skrill: (***cardholder***, ***number***, ***expiry date*** and ***cvv***).

For example:

```
$debit = new SkrillPsp_Debit();
…
$debit->setTransactionId("Ckdkd12")
             ->setCardholder("John Smith")
             ->setCardNumber("4000000000000051")
             ->setExpiryDate("12/2016")
             ->setCVV(122)
```

Request using a reference ID

For a ***refund***, ***capture*** or ***cancel*** request, you can provide the ***referenceId*** as the constructor parameter.

For example:

```
$capture = new SkrillPsp_Capture("c21a0ef567a14d7bad56582099f9f7c8");
```

Where ***("c21a0ef567a14d7bad56582099f9f7c8")*** is the reference ID created for the original transaction.

Note: The ***referenceId*** parameter is mandatory for a ***refund*** and ***cancel*** request.

3.3.2. Set the Skrill API endpoint

The next step is to set the Skrill API endpoint. The API endpoint string should point to either the Skrill test or live environment:

- Test environment: https://psp.sandbox.dev.skrillws.net/v1/json/
- Live environment: https://psp.skrill.com/v1/json/

The string must also include the following parameters:

Table 3-2: API endpoint parameters

Parameter	Description	Examples
Merchant ID	A unique merchant identifier on the Skrill payment processing platform.	***zmcdptfj8kv9fwhj***

Table 3-2: API endpoint parameters

Parameter	Description	Examples
Channel ID	Identifier of the channel in which you are processing payments.	***channelid_3d***
Payment type	Identifier of the payment method, such as ***credit card*** or an alternative payment method, such as ***skrillwallet***.	***creditcard***

Your account manager or the Skrill Merchant Services Team can provide you with these details. Contact them to discuss any additional channel requirements.

Example

The example below shows how to set the API endpoint:

```
$debit-> setUri("https://psp.sandbox.dev.skrillws.net/v1/json/3e40a821/
channelid_3d/creditcard");
```

Dynamically setting the API endpoint

The following alternative coding method enables you to dynamically provide and change your ***Merchant ID***, ***Channel ID*** and ***payment method***:

```
$debit->setMerchantUrl("https://psp.sandbox.dev.skrillws.net/v1/json", "3e40a821",
"channelid_3d", "creditcard");
```

3.3.3. Set the parameters to send to Skrill

The next step is to set the payment parameters that you want to send to the Skrill Payment Platform. Each type of request needs to include different parameters. Some parameters are mandatory, while others are optional. For a detailed list of supported parameters, see ***Payment Parameters on page 19.***

There are a number of methods that you can use to set the parameters in your PHP file:

Using the 'SetParameters' method

The code example below uses the ***SetParameters*** method to include the payment parameters in an array:

```
$params = array(
        'identification' => array(
            'transactionid' => 'CRM_F56A',
            'customerid' => '122456'
        ),
        'payment' => array(
            'amount' => '128',
            'currency' => 'USD',
            'descriptor' => 'description'
        ),
        'account' => array(
            'cardholder' => 'Joe Dofof',
            'number' => '4000000000000051',
            'expiry' => '01/2016',
            'cvv' => '123'
        ),
        'customer' => array(
            'name' => array(
```

```
                'title' => 'Mrs',
                'firstname' => 'Sarah',
                'lastname' => 'Roly',
                'company' => 'Cisco Systems'
            ),
            'address' => array(
                'street' => 'Test stree',
                'zip' => '1001',
                'city' => 'Berlin',
                'state' => 'BE',
                'country' => 'DE'
            ),
            'contact' => array(
                'phone' => '  +4545454545 ',
                'mobile' => '',
                'email' => 'info@test',
                'ip' => '122.20.20'
            )
        ),
        'merchant' => array(
            'key' => 'value4545'
        )
    );
    $preauth->setParameters($params);
```

The ***setParameters*** method enables you to set all parameters at once. However, if you do not provide the mandatory parameters for the type of API request type, or you provide invalid data, then an exception is thrown.

With this method you cannot modify or overwrite the predefined JSON-RPC structure for the request, which prevents you from sending incorrect data to the Skrill Payment Platform.

Using the Setter method

The code example below uses the setter method to include the payment parameters:

```
$debit->setTransactionId("Ckdkd12")
             ->setCustomerId("39393993")
             ->setAmount("457")
             ->setCurrency("EUR")
             ->setDescriptor("description")
             ->setCardholder("John Smith")
             ->setCardNumber("4000000000000051")
             ->setExpiryDate("12/2016")
             ->setCVV(122)
             ->setTitle("Mr.")
             ->setFirstName('Ibrahim')
             ->setLastName('Musa')
             ->setCompany('Cicso')
             ->setStreet('Market Street')
             ->setZip('1001')
             ->setCity('London')
             ->setState('London')
             ->setCountry('UK')
             ->setPhone('202020202')
             ->setMobile('+440203030303')
             ->setEmail('nbn313@yahoo.com')
             ->setIpAddress('127.1.0.0');
```

Setter methods support method chaining. With setters, you set the parameters one by one.

Note: For a ***credit*** or ***debit*** request, if you provide a payment token within the constructor, then you do not need to set the Account group parameters (***cardholder***, ***number***, ***ExpiryDate*** and ***cvv***).

3.3.4. View the JSON request parameters

For debugging purposes you can use the ***showJson*** method to view the JSON request parameters and values that are sent to the Skrill Payment Platform. This enables you to check your request either before or after making the call.

For example:

```
echo $debit->showJson();
```

The above method will print the JSON code to your browser window.

3.3.5. Send the API request

The final step is to send the request with the ***makeCall*** method.

For example:

```
$result = $debit->makeCall();
```

Change for other API request methods, e.g., $credit and $refund.

3.3.6. Handle the response

The ***transaction*** response will be either a success object or an error object.

The response object returned for a payment request may be one of the following three types:

- ***SkrillPsp_Response_Success*** – returned on success
- ***SkrillPsp_Response_Error*** – returned on error
- ***SkrillPsp_Response_ErrorLevel*** – returned on error level 1

As an alternative to using ***var_dump*** on the returned response to see the structure of the data, you can use one of the following to check the type of response object:

```
if($result->isSuccess()) {
        // work with success object
        echo $result->getReferenceId();
}
elseif ($result->isError()) {
        // work with error object
        echo $result->getErrorMessage();
}
elseif ($result->isErrorLevel()) {
       // work with errorlevel object
       echo $result->getAdvice();
}
```

- Or -

```
if($result instanceof SkrillPsp_Response_Success) {
        // work with success object
        echo $result->getTimeStamp();
}
elseif ($result instanceof SkrillPsp_Response_Error)
{
       // work with error object
        echo $result->getErrorMessage();
```

```
}
elseif($result instanceof SkrillPsp_Response_ErrorLevel)
{
        // work with errorlevel object
        echo $result->getErrorCode();
}
```

- ***isSuccess*** — returns true only if $result is ***SkrillPsp_Response_Success***
- ***isError***—retruns true only if $result is ***SkrillPsp_Response_Error***
- ***isErrorLevel*** —returns true only if $result is ***SkrillPsp_Response_ErrorLevel***
- ***getJsonResponse*** —retrieves response from server in JSON format. This allows raw json to be seen without php wrapper code. This method is supported by all three response objects no matter success or error.

3.3.6.1. Handling a successful level 1 response (SkrillPsp_Response_Success)

You can use the ***getAccount*** method to retrieve generic response values.

Note: You can safely use these methods directly with other type of Response objects without doing ***isset()*** checks. If the key does not exist, then the value will be returned as null.

Table 3-3: Methods used to retrieve parameters for a successful response

Method	Description of parameter returned
getAccount	The Account group response parameters for the JSON-RPC Response object.
getCode	The code that dictates the status of the transaction.
getId	The request identifier.
getIdentification	The identification response parameters from JSON-RPC Response object.
getLevel	The level response field.
getMessage	The message that indicates the status of the transaction.
getMethod	The payment method.
getPayment	The payment response parameters from JSON-RPC Response object.
getRedirectUr	The Redirect URL in Alternative Payments. If card payments returns null
getReferenceId	A unique reference generated by Skrill's payment platform to identify the transaction or null.
getRequestType	The request type.
getTimeStamp	The time stamp parameter for Alternative payments. For a card payment, this method returns a value of null.
getVersion	The JSON-RPC protocol version.

Retrieving details from a card registration response

The following methods return values from the ***Account*** group for a successful credit/debit card Register response.

Table 3-4: Methods used to retrieve parameters for a successful card registration response

Method	Description of parameter returned
getToken	The generated token, or null if another request type is made.
getBin	The BIN, or null if other request type is made.
getExpiryMonth	The card's Expiry Month or null if another request type is made.
getExpiryYear	The card's Expiry Year or null if other request type is made
getLast	The last 4 digits of the customer's card number, or null if another request type is made.
getMasked	The Masked Card Number or null if other request type is made

Retrieving parameters that are not part of any parameter group

Parameters from the response that are not part of any parameter group can be accessed via public properties. For example:

```
$response->type;
$response->message;
$response->method;
$response->code;
$response->level; Where $response is Skrill_Response_Success object
```

Note: ***SkrillPsp_Response_ErrorLevel*** and ***Skrill_Response_Error*** do not support public properties.

Returning a response array in JSON format

You can use the **getArrayResponse** object to retrieve the full response from the Skrill Payment Platform in JSON format. It is supported by all response objects (Error, ErrorLevel and Success). For example:

```
elseif ($result->isSuccess()) {
        $arr = $result->getArrayResponse();
        foreach($arr as $key => $value) {
                // do some stuff here
        }
}
```

3.3.6.2. Handling an error response (SkrillPsp_Response_ErrorLevel)

The following methods can be used to retrieve details of any error response:

Table 3-5: Methods used to retrieve parameters for an error response

Method	Description of parameter returned
isSuccess	Will always be false for this object.

Table 3-5: Methods used to retrieve parameters for an error response

Method	Description of parameter returned
isError	Will always be true for this object.
isErrorLevel	Will always be true for this object.
getId	The request identifier.
getVersion	The JSON-RPC version.
getErrorLevel	The error level that the response is coming from.
getErrorCode	The response code returned from the acquirer or card issuer.
getMethod	The method parameter of the original request, such as debit or preauthorization.
getType	An abbreviation of the type of request that was invoked.
getErrorMessage	A short description of the error.
getAdvice	The advice for the next step to take.
getIdentification	The identification group provided in the original request.

The $response object for an error response contains the message, code and data (optional) fields.

You can obtain the id and JSON version in the same way as with a success response. The code and message fields can be obtained as follows:

- through the methods getErrorCode and getErrorMessage:

```
echo $result->getErrorMessage();
        echo $result->getErrorCode();
```

- through the getError method, which returns the CODE and MESSAGE members of JSON-RPC Error object:

```
$errorData = $result->getError();
       echo $errorData->code;
       echo $errorData->message;
```

Level, errormessage and advice can be obtained as follows:

- through the getErrorData method which returns the DATA member of JSON-RPC Error object:

```
$data = $result->getErrorData();  and iterate through $data
```

- directly through the getErrorLevel, getAdvice and getErrorDataMessage methods:

```
echo $result->getErrorLevel();
echo $result->getErrorDataMessage();
echo $result->getAdvice();
```

- ***getErrorData*** provides a short description of the error.
- ***getErrorDataMessage*** provides a more detailed description of the error.

The $response object for an error response contains the message, code and data (optional) fields.

3.3.7. Handling responses for the Web Payment Frontend

The Web Payment Frontend (WPF) object returns a string that is the WPF endpoint, with provided parameters encoded and appended to it. For example:

```
http://wpf.dev.skrillws.net/payments/new?payload=<your-encryped-payload>)
```

You can retrieve this using the ***getResult*** method. The value returned from this method is then used to make a WPF call.

4. PAYMENT PARAMETERS

4.1. Payment request

Below is a list of payment parameters that can be included in your request.

Table 4-1: Payment request parameters

Field Name	Description	Required?	Length/ format
transaction_id	A free text field included by you to track the transaction. This should be unique.	Yes	0-256 (alphanumeric)
customer_id	Free text field that enables you to pass through your unique customer identifier.	No	0-256 (alphanumeric)
reference_id	Used to refer to a previous transaction for example, when implementing a capture, refund or a cancel request. This must be the unique ID generated by Skrill to track the transaction.	Conditional (depending on type of transaction)	32 (alphanumeric)
amount	Transaction amount in the specified currency. There is no decimal separator. This number must be at least two digits. The API is built on a cent-based ISO8583 structure that does not cater for decimals (e.g., "." or ","). For example: 10 = ten Euro cents, 100 = 1 Euro, 1000 = 10 Euros.	Yes	2-12 (numeric)
currency	Currency that the transaction is processed in. Currency code is according to the ISO 4217 specification. For example: USD (US Dollars).	Yes	3 (alpha)
descriptor	Enables you to add a description to be displayed on the customer's credit card statement. Note that this descriptor may be truncated by the customer's bank/ card issuer. The descriptor has two lines: the first line is dynamic and the second line is static. Only the Dynamic part is sent in the ***descriptor*** field.	Yes	1-128 (alpha) Line 1: 21 characters Line2: 16 characters
cardholder	Name of the cardholder (as it is displayed on the card).	Optional	4-128 (alpha)
number	The long card number of the credit card or debit card used to make the payment.	Yes	15-16 (numeric)
expiry	The expiry date of the card used.	Yes	7 (alphanumeric)

Table 4-1: Payment request parameters

Field Name	Description	Required?	Length/ format
cvv	The Card Verification Value or CSC (Card Security Code). This is the last 3 digits of the customer's security code on the back on any MasterCard or Visa branded cards or 4 digits for Amex cards. **Note**: the CVV value must not be stored in any capacity.	Yes	3-4 (numeric)
token	Enables you to use a token to refer to an existing card stored on the Skrill system. Can be used for repeat billing transactions.	Yes, if using the Skrill tokenization service	32 (alphanumeric)
language	Language of the payment front-end.	No	2 digit, ISO code
response_url	The URL to which Skrill sends the response data.	Yes	Any length (alphanumeric)
success_url	The URL to which the customer will be redirected to in case of successful payment	Yes	Any length (alphanumeric)
error_url	The URL to which the customer will be redirected to in case of unsuccessful payment	Yes	Any length (alphanumeric)
title	Title of the end customer: Mr, Mrs, Ms, Dipl.-Ing, Dr, Phd, Prof.	Optional	2-6 (alpha)
lastname	The customer's surname	Optional	2-40 (alpha)
firstname	The customer's first name	Optional	2-40 (alpha)
company	The company the customer works for	Optional	2-40 (alphanumeric)
street	The first part of the customer's address, including the street and house number.	Conditional	5-128 (alphanumeric)
zip	The customer's postal/zip code	Conditional	1-10 (alphanumeric)
city	The customer's city	Conditional	2-30 (alpha)
state	The customer's state, county or local region	Conditional	2-30 (Alpha)
country	Country code according to the ISO 3166-1 specification.	Conditional	2 (alpha)
phone	The customer's land line phone number. Must start with a digit or a '+'.	Optional	8-64 (alphanumeric)
mobile	The customer's mobile phone number. Must start with a digit or a '+'.	Optional	8-64 (alphanumeric)

Table 4-1: Payment request parameters

Field Name	Description	Required?	Length/ format
email	The customer's email address	Conditional	6-128 (alphanumeric)
ip	The IP address that was captured and sent through in the request.	Mandatory	15 (alphanumeric)
key0 to key9	There are 10 free fields that allow you to include additional transactional information, which can be used by your back-end systems (e.g., your Order Management System).	No	0-32 (alphanumeric)
value0 to value9	The actual value passed through associated with the key parameters you have defined.	No	0-1024 (alphanumeric)

4.2. Payment Response

The table below describes the attributes that are returned within the response.

Table 4-2: Payment request response fields

Field Name	Description	Length / format
transaction_id	The unique transaction identifier value that was provided in your original transaction request.	0-256 (alphanumeric)
customer_id	The unique customer identifier value that was provided in your original transaction request.	0-256 (alphanumeric)
unique_id	A unique reference generated by the Skrill Payment Platform to identify the transaction.	32 (alpha)
short_id	A short identifier that can also be used as a reference to the transaction.	16-25 (alphanumeric)
amount	Transaction amount in specified currency. There is no decimal separator. Must be least two digits.	2-10 (numeric)
currency	Currency that the transaction is being processed in. Currency code according to the ISO 4217 specification.	3 (alpha)
descriptor	The value you provided in the transaction request to be displayed on the customer's credit card statement.	0-128 (alpha)
token	The token returned if you made a register request to the Skrill Payment Platform to store the customer's card data. You can use this token for future payments from the same cardholder, without requiring them to re-enter their card details.	32 (alphanumeric)
code	The value that dictates the status of the transaction. For example: "***0***"	1-2 (integer)
type	An abbreviation of the type of request that was invoked in the merchant request. For example "***pa***".	2-3 (alpha)

Table 4-2: Payment request response fields

Field Name	Description	Length / format
message	This message indicates the status of the transaction. For example: "***approved***"	128 (alphanumeric)
id	A value of any numeric type, which you can use to match the response with the request that is being replied too.	0-16 (numeric)

4.2.1. Error response

This response is only returned if an error occurs.

Table 4-3: Error response

Field Name	Description	Required?
message	A short description of the error. For example: "***skrillwallet error***".	256 (alphanumeric)
level	This value indicates the level that the response is coming from: 0 = Payment Service Level (ACK response) 1 = Format Error (front-end validation errors) 2 = System Error (configuration/connectivity errors) 3 >= In the event of an error the level value will depict what acquirer the error resides with (NOK response)	1-2 (integer)
error_message	A more detailed description of the error. For example: "***technical error***".	
advice	Provides advice on the next steps to take. For example: "***Please contact technical support***".	256 (alphanumeric)
id	A value of any numeric type, which you can use to match the response with the request that is being replied too.	0-16 (numeric)

5. PHP EXAMPLES

This section contains examples of common API call requests.

5.1. preauthorization

Request a ***preauthorization*** of an amount on a credit or debit card. If you do not have a payment token, the card details must be passed in the request.

```
<?php

// include library
require_once '/lib/SkrillPSP.php';
try {

// Create request object
    $preauth = new SkrillPsp_Preauthorization ("c21a0ef567a14d7bad56582099f9f7c8");

// Set API endpoint
    $preauth->setMerchantUrl ("https://psp.sandbox.dev.skrillws.net/v1/json",
"3e40a821", "channelid_3d", "creditcard");

// Set required parameters
    $params = array(
        'identification' => array(
            'transactionid' => 'CRM_F56A',
            'customerid' => '122456'
        ),
        'payment' => array(
            'amount' => '128',
            'currency' => 'USD',
            'descriptor' => 'description'
        ),
        'account' => array(
            'cardholder' => 'Joe Dofof',
            'number' => '4000000000000051',
            'expiry' => '01/2016',
            'cvv' => '123'
        ),
        'customer' => array(
            'name' => array(
                'title' => 'Mrs',
                'firstname' => 'Sarah',
                'lastname' => 'Roly',
                'company' => 'Cisco Systems'
            ),
            'address' => array(
                'street' => 'Test stree',
                'zip' => '1001',
                'city' => 'Berlin',
                'state' => 'BE',
                'country' => 'DE'
            ),
            'contact' => array(
                'phone' => ' +4545454545 ',
                'mobile' => '',
                'email' => 'info@test',
                'ip' => '122.20.20'
            )
        ),
```

```
        'merchant' => array(
            'key' => 'value4545'
        )
    );
    $preauth->setParameters($params);

// Check the request
    echo $preauth->showJson();

// send request and save the response
    $result = $preauth->makeCall();

// check response object for success or error
      if($result->isSuccess()) {

      // work with result method and properties
          $identity = $result->getIdentification();
          $payment = $result->getPayment();
          echo $token = $result->getTokenFromAccount();
          echo $result->type;
          echo "<br />";
          echo $result->message;
          echo "<br />";
          echo $result->method;
          echo $result->getCode();
          echo "<br />";
          echo $result->getLevel();
      }
      else {

      // work with error data
         $data = $result->getErrorData();
         $errorData = $result->getError();
         var_dump($data);
         echo $errorData->code;
         echo $errorData->message;
         echo $result->getErrorLevel();
         echo "<br />";
         echo $result->getErrorMessage();
         echo $result->getErrorCode();
         echo $result->getErrorDataMessage();
         echo $result->getId();
         echo "<br />";
         foreach ($data as $val)
         {
          echo "$val<br />";
         }
}
catch (SkrillPsp_Exception $e) {
    echo $e->getMessage();
}
```

5.2. debit

Request a ***debit*** on a credit or debit card. The amount is authorised and captured in a single request. If you do not have a payment token, the card details must be passed in the request.

```
<?php

// include library
require_once '/lib/SkrillPSP.php';
try {

// Create request object
    $debit = new SkrillPsp_Debit();

// Set API endpoint
    $debit->setMerchantUrl("https://psp.sandbox.dev.skrillws.net/v1/json",
"3e40a821", "channelid_3d", "creditcard");

// Set required parameters
    $params = array(
        'identification' => array(
            'transactionid' => 'CRM_F56A',
            'customerid' => '122456'
        ),
        'payment' => array(
            'amount' => '128',
            'currency' => 'USD',
            'descriptor' => 'description'
        ),
        'account' => array(
            'cardholder' => 'Joe Dofof',
            'number' => '4000000000000051',
            'expiry' => '01/2016',
            'cvv' => '123'
        ),
        'customer' => array(
            'name' => array(
                'title' => 'Mrs',
                'firstname' => 'Sarah',
                'lastname' => 'Roly',
                'company' => 'Cisco Systems'
            ),
            'address' => array(
                'street' => 'Test stree',
                'zip' => '1001',
                'city' => 'Berlin',
                'state' => 'BE',
                'country' => 'DE'
            ),
            'contact' => array(
                'phone' => ' +4545454545 ',
                'mobile' => '',
                'email' => 'info@test',
                'ip' => '122.20.20'
            )
        ),
        'merchant' => array(
            'key' => 'value4545'
        )
    );
    $debit->setParameters($params);

// Check the request
    echo $debit->showJson();
```

```
// send request and save the response
    $result = $debit->makeCall();

// check response object for success or error
       if($result->isSuccess()) {

       // work with result method and properties
           $identity = $result->getIdentification();
           $payment = $result->getPayment();
           echo $token = $result->getTokenFromAccount();
           echo $result->type;
           echo "<br />";
           echo $result->message;
           echo "<br />";
           echo $result->method;
           echo $result->getCode();
           echo "<br />";
           echo $result->getLevel();
       }
       else {

       // work with error data
          $data = $result->getErrorData();
          $errorData = $result->getError();
          var_dump($data);
          echo $errorData->code;
          echo $errorData->message;
          echo $result->getErrorLevel();
          echo "<br />";
          echo $result->getErrorMessage();
          echo $result->getErrorCode();
          echo $result->getErrorDataMessage();
          echo $result->getId();
          echo "<br />";
          foreach ($data as $val)
          {
           echo "$val<br />";
          }
}
catch (SkrillPsp_Exception $e) {
    echo $e->getMessage();
}
```

5.3. capture

Request a ***capture*** of a preauthorised amount, which can be partial or up to the full amount of the original preauthorization. The transaction reference ID is required.

```
<?php

// include library
require_once '/lib/SkrillPSP.php';
try {

// Create request object
    $capture = new SkrillPsp_Capture("c21a0ef567a14d7bad56582099f9f7c8");

// Set API endpoint
    $capture->setMerchantUrl("https://psp.sandbox.dev.skrillws.net/v1/json",
"3e40a821", "channelid_3d", "creditcard");

// Set required parameters
    $params = array(
        'identification' => array(
            'transactionid' => '32160904e6ea4a50bf964e5e095f5692',
            'customerid' => '122456'
        ),
```

```
        'payment' => array(
            'amount' => '122',
            'currency' => 'EUR',
            'descriptor' => 'item description'
        )
    );
    $capture->setParameters($params);

// Check the request
    echo $capture->showJson();

// send request and save the response
    $result = $capture->makeCall();

// check response object for success or error
      if($result->isSuccess()) {

      // work with result method and properties
          $identity = $result->getIdentification();
          $payment = $result->getPayment();
          echo $token = $result->getTokenFromAccount();
          echo $result->type;
          echo "<br />";
          echo $result->message;
          echo "<br />";
          echo $result->method;
          echo $result->getCode();
          echo "<br />";
          echo $result->getLevel();
      }
      else {

      // work with error data
         $data = $result->getErrorData();
         $errorData = $result->getError();
         var_dump($data);
         echo $errorData->code;
         echo $errorData->message;
         echo $result->getErrorLevel();
         echo "<br />";
         echo $result->getErrorMessage();
         echo $result->getErrorCode();
         echo $result->getErrorDataMessage();
         echo $result->getId();
         echo "<br />";
         foreach ($data as $val)
         {
          echo "$val<br />";
         }
}
catch (SkrillPsp_Exception $e) {
    echo $e->getMessage();
}
```

5.4. credit

Request a ***credit*** payment to cardholder, which is not linked to any transaction and can be for any amount. If you do not have a payment token, the card details must be passed in the request.

```
<?php
// include library
require_once '/lib/SkrillPSP.php';
try {

// Create request object
    $credit = new SkrillPsp_Credit("c21a0ef567a14d7bad56582099f9f7c8");

// Set API endpoint
    $register->setMerchantUrl("https://psp.sandbox.dev.skrillws.net/v1/json",
"3e40a821", "channelid_3d", "creditcard");

// Set required parameters
    $params = array(
        'identification' => array(
            'transactionid' => 'CRM_F56A',
            'customerid' => '122456'
        ),
        'payment' => array(
            'amount' => '122',
            'currency' => 'EUR',
            'descriptor' => 'item description'
        )
    );
    $credit ->setParameters($params);

// Check the request
    echo $credit ->showJson();

// send request and save the response
      $result = $credit ->makeCall();

// check response object for success or error
       if($result->isSuccess()) {

       // work with result method and properties
           $identity = $result->getIdentification();
           $payment = $result->getPayment();
           echo $token = $result->getTokenFromAccount();
           echo $result->type;
           echo "<br />";
           echo $result->message;
           echo "<br />";
           echo $result->method;
           echo $result->getCode();
           echo "<br />";
           echo $result->getLevel();
       }
       else {

       // work with error data
          $data = $result->getErrorData();
          $errorData = $result->getError();
          var_dump($data);
          echo $errorData->code;
          echo $errorData->message;
          echo $result->getErrorLevel();
          echo "<br />";
          echo $result->getErrorMessage();
          echo $result->getErrorCode();
          echo $result->getErrorDataMessage();
          echo $result->getId();
          echo "<br />";
          foreach ($data as $val)
```

```
        {
         echo "$val<br />";
        }
}
catch (SkrillPsp_Exception $e) {
    echo $e->getMessage();
}
```

5.5. cancel

Request to ***cancel*** a previous transaction. This can only be done up to a certain cut-off point, *before* the transaction has been sent to the acquirer. The transaction reference ID is required.

```
<?php

// include library
require_once '/lib/SkrillPSP.php';
try {

// Create request object
    $capture = new SkrillPsp_Capture("c21a0ef567a14d7bad56582099f9f7c8");

// Set API endpoint
    $capture->setMerchantUrl("https://psp.sandbox.dev.skrillws.net/v1/
json","3e40a821","channelid_3d","creditcard");

// Set required parameters
    $params = array(
        'identification' => array(
            'transactionid' => '32160904e6ea4a50bf964e5e095f5692',
            'customerid' => '122456'
        ),
        'payment' => array(
            'amount' => '122',
            'currency' => 'EUR',
            'descriptor' => 'item description'
        )
    );
    $capture->setParameters($params);

// Check the request
    echo $capture->showJson();

// send request and save the response
    $result = $capture->makeCall();

// check response object for success or error
       if($result->isSuccess()) {

       // work with result method and properties
           $identity = $result->getIdentification();
           $payment = $result->getPayment();
           echo $token = $result->getTokenFromAccount();
           echo $result->type;
           echo "<br />";
           echo $result->message;
           echo "<br />";
           echo $result->method;
           echo $result->getCode();
           echo "<br />";
           echo $result->getLevel();
       }
       else {

       // work with error data
          $data = $result->getErrorData();
          $errorData = $result->getError();
```

```
         var_dump($data);
         echo $errorData->code;
         echo $errorData->message;
         echo $result->getErrorLevel();
         echo "<br />";
         echo $result->getErrorMessage();
         echo $result->getErrorCode();
         echo $result->getErrorDataMessage();
         echo $result->getId();
         echo "<br />";
         foreach ($data as $val)
         {
          echo "$val<br />";
         }
}
catch (SkrillPsp_Exception $e) {
    echo $e->getMessage();
}
```

5.6. register

Request to ***register*** a credit or debit card. Skrill returns a payment token, which can be used in subsequent transactions with this card. You will need to pass the CVV when using the token in subsequent transactions.

```
<?php

// include library
require_once '/lib/SkrillPSP.php';
try {

// Create request object
    $register = new SkrillPsp_Register();

// Set API endpoint
    $register->setMerchantUrl("https://psp.sandbox.dev.skrillws.net/v1/
json","3e40a821","channelid_3d","creditcard");

// Set required parameters
    $register->setCardNumber("4000000000000002")
             ->setExpiryDate("12/2016")
             ->setCVV("111");

// Check the request
    echo $register->showJson();

// send request and save the response
    $result = $register->makeCall();

// check response object for success or error
       if($result->isSuccess()) {

       // work with result method and properties
           $identity = $result->getIdentification();
           $payment = $result->getPayment();
           echo $token = $result->getTokenFromAccount();
           echo $result->type;
           echo "<br />";
           echo $result->message;
           echo "<br />";
           echo $result->method;
           echo $result->getCode();
           echo "<br />";
           echo $result->getLevel();
       }
       else {
```

```
      // work with error data
         $data = $result->getErrorData();
         $errorData = $result->getError();
         var_dump($data);
         echo $errorData->code;
         echo $errorData->message;
         echo $result->getErrorLevel();
         echo "<br />";
         echo $result->getErrorMessage();
         echo $result->getErrorCode();
         echo $result->getErrorDataMessage();
         echo $result->getId();
         echo "<br />";
         foreach ($data as $val)
         {
          echo "$val<br />";
         }
}
catch (SkrillPsp_Exception $e) {
    echo $e->getMessage();
}
```

5.7. refund

Request to ***refund*** a previous amount. This can be for a partial amount or the full amount of the original transaction. The transaction reference ID is required.

```
<?php

// include library

require_once '/lib/SkrillPSP.php';
try {

// Create request object
    $refund = new SkrillPsp_Refund("c21a0ef567a14d7bad56582099f9f7c8");

// Set API endpoint
    $register->setMerchantUrl("https://psp.sandbox.dev.skrillws.net/v1/json",
"3e40a821", "channelid_3d", "creditcard");

// Set required parameters
    $params = array(
        'identification' => array(
            'transactionid' => 'CRM_F56A',
            'customerid' => '122456'
        ),
        'payment' => array(
            'amount' => '122',
            'currency' => 'EUR',
            'descriptor' => 'item description'
        )
    );
    $refund->setParameters($params);

// Check the request
    echo $refund->showJson();

// send request and save the response
    $result = $refund->makeCall();

// check response object for success or error
       if($result->isSuccess()) {

       // work with result method and properties
           $identity = $result->getIdentification();
           $payment = $result->getPayment();
```

```
            echo $token = $result->getTokenFromAccount();
            echo $result->type;
            echo "<br />";
            echo $result->message;
            echo "<br />";
            echo $result->method;
            echo $result->getCode();
            echo "<br />";
            echo $result->getLevel();
        }
        else {

        // work with error data
           $data = $result->getErrorData();
           $errorData = $result->getError();
           var_dump($data);
           echo $errorData->code;
           echo $errorData->message;
           echo $result->getErrorLevel();
           echo "<br />";
           echo $result->getErrorMessage();
           echo $result->getErrorCode();
           echo $result->getErrorDataMessage();
           echo $result->getId();
           echo "<br />";
           foreach ($data as $val)
           {
            echo "$val<br />";
           }
}
catch (SkrillPsp_Exception $e) {
    echo $e->getMessage();
}
```

6. WEB PAYMENT FRONTEND (WPF)

The Web Payment Frontend (WPF) is a Skrill hosted payment page, where Skrill requests card and payment details for the customer. You can integrate directly to it via the PHP library.

Below is an example of a request to the Web Payment Frontend (WPF). Parameters unique to the WPF are highlighted.

```
<?php
require_once 'lib/SkrillPSP.php';

try {
	$params = array(
			"amount" => 1000,
			"merchant_id" => "3e40a821",
			"customer" => array(
					"firstname" => "John",
					"country" => "DE",
					"email" => "wpf@skrill.com",
					"ip" => ($_SERVER['REMOTE_ADDR'])
			),
			"method" => "debit",
			"currency" => "USD",
			"theme"=>"skeuo",
			"descriptor" => "wpf test",
			"transaction_id" => "FOO",
			"channel_id" => "channelid_3d",
			"success_url" => "http://merchant.com/success.php",
			"response_url" => "http://merchant.com/response.php",
			"error_url" => "http://merchant.com/error.php"
	);

// Create WPF object
	$obj = new WPF_Prepare();

// Set service endpoint
	$obj->setUri("https://wpf.dev.skrillws.net/3e40a821/payments/new?payload");

// Set request parameters
	$obj->setParameters("B[zcj2AGQmL@3k", $params);

// Get the result - wpf endpoint with provided parameters appended to it
	echo $obj->getResult();

}
catch (Exception $e) {
	echo $e->getMessage();
}
```

Below is a summary of the main parameters that can be included in your request to the WPF. For further details about using the Web Payment Frontend, refer to the ***Web Payment Frontend Guide***.

6.1. Customer details

You can pass details of the customer through to the WPF.

Table 6-1: Payment parameters – customer details

Field Name	Description	Status*	Length/ format
title	Title of the end customer: Mr, Mrs, Ms, DPLING, DR, PHD, PROF.	Optional	2-6 (alpha)
lastname	The customer's last name	Optional	2-40 (alpha)
firstname	The customer's first name	Optional	2-40 (alpha)
company	The company the customer works for	Optional	2-40 (alphanumeric)
street	The customer's first line of their address	Conditional	5-128 (alphanumeric)
zip	The customer's postal/zip code	Conditional	1-10 (alphanumeric)
city	The Customer's city	Conditional	2-30 (alpha)
state	The customer's state, county or local region	Conditional	2-30 (Alpha)
country	Country code according to the ISO 3166-1 specification.	Conditional	2 (alpha)
phone	The customer's land line phone number. Must start with a digit or a '+'.	Optional	8-64 (alphanumeric)
mobile	The customer's mobile phone number. Must start with a digit or a '+'.	Optional	8-64 (alphanumeric)
email	The customer's email address.	Mandatory	6-128 (alphanumeric)
ip	The IP address that was captured and sent through in the request.	Mandatory	15 (alphanumeric)
id	Free text field that enables you to pass through your unique customer identifier.	Optional	0-256 (alphanumeric)

* **Optional** – indicates a non-mandatory parameter. **Conditional** indicates a parameter that may be mandatory, if it is used by any of the risk checks performed (for example, values are used for AVS checks).

Tip: If you are not sure whether a conditional parameter may be required, check with your account manager.

6.2. Payment and configuration details

These are details of the payment, plus other internal configuration parameters that indicate to the WPF how to handle the payment request. These parameters are hidden from the customer and are not displayed on the Skrill payment form.

Table 6-2: Hidden payment parameters

Parameter	Description	Status	Value/Length
transaction_id	A free text field included by you to track the transaction. This should be unique.	Mandatory	0-256 (alphanumeric)
checkout_label	Text to be displayed on the **Submit** button on the payment page. For example: **Pay Now**. If not provided, the default 'Submit' text is displayed.	Optional	Text; up to 25 characters
amount	Transaction amount in the specified currency. There is no decimal separator. This number must be at least two digits. The API is built on a cent-based ISO8583 structure that does not cater for decimals (e.g., "**.**" or "**,**"). For example: 10 = ten Euro cents, 100 = 1 Euro, 1000 = 10 Euros.	Mandatory	2-12 (numeric)
currency	Currency that the transaction is processed in. Currency code is according to the ISO 4217 specification. For example: USD (US Dollars).	Mandatory	3 (alpha)
method	Defines the type of payment request you want to use, such as ***preauthorisation*** or ***debit***.	Mandatory	Alpha
descriptor	Enables you to add a description to be displayed on the customer's bank statement. Note that this descriptor may be truncated by the customer's bank.	Optional	1-128 (alpha)
channel_id	Your unique channel ID for the transaction.	Mandatory	Alphanumeric
theme	Enables you to select one of the pre-defined style sheets that define the look and feel of the payment page.	Optional	Alpha
locale	Code indicating the language/country in which the payment page is to be displayed. For example, ***en-GB*** for British English and de-DE for **German**.	Optional	5 digit code
target	Specifies the frame where URL should be rendered in.	Optional	Alpha
required_ attributes	Includes a list of parameters that must be completed by the customer. The WPF will validate these fields on the form.	Optional	List of attributes
editable_ attributes	Includes a list of parameters whose values can be edited by the customer.	Optional	List of attributes

Table 6-2: Hidden payment parameters

Parameter	Description	Status	Value/Length
noneditable_ attribute_style	Defines the behaviour of non-editable attributes. Options include: • **hidden** – do not display these fields on the form. • **none** – display these fields as normal input fields, but will be ignored. Can be used to apply custom styles. • **disabled** – this option is disabled. i.e., will be displayed but not editable.	Optional	List of attributes
success_url	Page where the customer is redirected if the transaction is successful.	Mandatory	Must be a valid URL on your website
error_url	Page where the customer is redirected if the transaction fails.	Mandatory	Must be a valid URL on your website
response_url	Page where you are notified of the result of the transaction. **Note**: this must be in HTTPS format. Your server should respond to a POST to this URL by returning ***HTTP 200*** with '***OK***' in the message body.	Mandatory	Must be a valid URL on your website
stylesheet_url	Link to a style sheet which defines your own look and feel for the payment page. If this field is left blank, Skrill uses the default style sheet. **Note**: This must be in CSS format and available via a secure (HTTPS) URL.	Optional	Must link to a valid CSS file.
javascript_url	Link to a JavaScript file on your website which defines the behaviour of the payment page. If this is not included, Skrill uses the default JavaScript file. **Note**: This must be in JS format and available via a secure (HTTPS) URL.	Optional	Must link to a valid JavaScript file.

7. ALTERNATIVE PAYMENTS

This section provides details of how to integrate local/alternative payment methods using the PHP library.

The table below describes the alternative payment methods supported via the PHP Library. For information on additional payment methods available through Skrill, refer to the ***Alternative Payments Guide***.

Table 7-1: Supported alternative payment methods

Payment Method	Description
iDEAL	Payment method that enables customers to make secure online bank transfers, without sharing card information with the merchant. It is one of the most popular payment methods in the Netherlands.
Paysafecard	Payment method enabling customers to pay online using prepaid vouchers. Vouchers are for fixed denominations and can be combined.
Skrill Direct	Skrill Direct is an instant online payment method that currently supports over 100 banks and reaches more than 110 million European consumers, who can directly pay from their online banking account.
Skrill Wallet	Enables customers to pay using their bank account, or debit and credit cards without having to enter any sensitive details online Customers also have access to a wide range of alternative payment methods
Yandex	Yandex.money is a popular eWallet and payment service, offered by Yandex, Russia's primary search engine, which is used by over 56 million users worldwide.

Each of these payment methods is described in further detail below.

7.1. iDEAL

iDEAL is a payment method that enables customers to make secure online bank transfers, without sharing card information with the merchant. It is one of the most popular payment methods in the Netherlands. iDEAL supports the following request methods:

- ***createDirectory*** – use this method to obtain a list of banks supporting iDEAL.
- ***createPreAuthorization*** – use this method to request authorization for debiting of funds from the customer's online bank account.

Note: Check with your account manager regarding the availability of this service.

7.1.1. Merchant website requirements

All Skrill merchants offering the iDEAL payment method on their website must comply with stringent requirements, which are mandated by iDEAL. For details, refer to the ***Skrill iDEAL Guide***.

7.1.2. Directory (iDEAL)

```
<?php
require_once 'lib/SkrillPSP.php';

try {

// Create iDEAL request object
       $obj = new SkrillPsp_iDEAL();

// Set service endpoint
       $obj->setUri("https://psp.sandbox.dev.skrillws.net/v1/json/3e40a821/
testchannel_ideal/ideal");

// Display the request in JSON format
       echo $obj->showJsonDirectory();

// Make iDEAL directory request
       $result = $obj->createDirectory();

// Handle the response
       if($result->isSuccess()) {
              $issuerid = $result->directory;
       }
       elseif ($result->isError()) {
              // error
       }
       elseif ($result->isErrorLevel()) {
             // error level
       }
}
catch (Exception $e) {
       echo $e->getMessage();
}
```

Below are further details of the request objects and parameters used with this payment method.

Request objects used with iDEAL CreatDirectory

Request Object	Description
new SkrillPsp_iDEAL()	Creates the iDEAL request object.
setUri()	Sets the API endpoint for this payment method. You must include your merchant ID and channel ID.
showJsonDirectory()	Displays the list of returned banks in JSON format.
createDirectory();	Invokes the ***CreatDirectory*** method request to iDEAL.

7.1.2.1. Response example

The example below shows the response returned from Skrill. You can use the issuer list to provide a list of iDEAL options to the customer. The issuer ID of the bank selected must then be passed to Skrill in your payment request.


```
{
   "result":    {
      "method": "ideal",
      "type": "directory",
      "level": 0,
      "code": 0,
      "directory":       [
                  [
            "Nederland",
            "Issuer Simulation V3 - ING",
            "INGBNL2A"
         ],
                  [
            "Nederland",
            "Issuer Simulation V3 - RABO",
            "RABONL2U"
         ]
      ]
   },
   "id": 1,
   "jsonrpc": "2.0"
}
```

7.1.3. Preauthorization (iDEAL)

This request obtains authorization for the online payment. Below is an example of the request.

```
<?php
require_once 'lib/SkrillPSP.php';

try {

// Create iDEAL request object
       $obj = new SkrillPsp_iDEAL();

// Set service endpoint
       $obj->setUri("https://psp.sandbox.dev.skrillws.net/v1/json/3e40a821/
testchannel_ideal/ideal");

// Set request parameters

$obj->setParameters(array(
        'identification' => array(
            'transactionid' => 'MerchntAsignedID',
            'customerid' => 'CustomerID'
        ),
        'payment' => array(
            'amount' => '107',
            'currency' => 'eur',
            'descriptor' => 'comment'
        ),
        'account' => array(
            'issuerid' => 'INGBNL2A'
        ),
        'frontend' => array(
            'language' => 'NL',
            'responseurl' => 'https://merchant.com/response.php',
            'successurl' => 'https://merchant.com/success.php',
            'errorurl' => 'https:// merchant.com/error.php'
        ),
        'customer' => array(
            'name' => array(
                'firstname' => 'Tom',
                'company' => 'Skrill LTD',
                'lastname' => 'Jones',
                'salutation' => 'Mr'
            ),
            'address' => array(
                'street' => '42 Street',
                'zip' => '6778',
                'city' => 'Amsterdam',
                'country' => 'NL'
            ),
            'contact' => array(
                'email' => 'inf@fi.nl',
                'ip' => '127.0.0.15'
            )
        )
    ));
// Display the request in JSON format
       echo $obj->showJson();

// Make the iDEAL preauthorization request
//This can include both createDirectory and createPreAuthorization requests
       // $dir = $obj->createDirectory();
       $result = $obj->createPreAuthorization();

// check response object for success or error
       if($result->isSuccess()) {

       // work with result method and properties
           $identity = $result->getIdentification();
           $payment = $result->getPayment();
           echo $token = $result->getTokenFromAccount();
```

```
            echo $result->type;
            echo "<br />";
            echo $result->message;
            echo "<br />";
            echo $result->method;
            echo $result->getCode();
            echo "<br />";
            echo $result->getLevel();
        }
        else {

        // work with error data
           $data = $result->getErrorData();
           $errorData = $result->getError();
           var_dump($data);
           echo $errorData->code;
           echo $errorData->message;
           echo $result->getErrorLevel();
           echo "<br />";
           echo $result->getErrorMessage();
           echo $result->getErrorCode();
           echo $result->getErrorDataMessage();
           echo $result->getId();
           echo "<br />";
           foreach ($data as $val)
           {
            echo "$val<br />";
           }
}
catch (SkrillPsp_Exception $e) {
    echo $e->getMessage();
}
?>
```

Below are further details of the request objects and parameters used with this payment method.

Table 7-2: Request objects used with iDEAL Preauthorization

Request Object	Description
new SkrillPsp_iDEAL()	Creates the iDEAL request object.
setUri()	Sets the API endpoint for this payment method. You must include your merchant ID and channel ID.
setParameters	Sets the parameters to be included in the preauthorization request. A description of the mandatory parameters is included below.
showJson ()	Displays the list of returned response in JSON format.
createPreAuthorization()	Invokes the preauthorization method request to iDEAL.

Table 7-3: Parameters used with iDEAL Preauthorization

Field Name	Description	Required	Length / format
transaction_id	A free text field included by you to track the transaction. This should be unique.	Yes	0-256 (alphanumeric)
amount	Transaction amount in the specified currency.	Yes	2-12 (string or integer, no decimals)
currency	Currency identifier (Alpha-3 ISO 4217 code)	Yes	3 (alpha)

Table 7-3: Parameters used with iDEAL Preauthorization

Field Name	Description	Required	Length / format
descriptor	Comment or description of the transaction which appears on the customer's bank statement.	Yes	1-256 (alpha)
Issuer_id	The issuer ID of the bank that is providing the customer online payment through iDEAL. This is returned in response to the ***directoryRequest***.	Yes	Alphanumeric
response_url	The URL to which the transaction response data is sent. **Note**: this must be in HTTPS format. Your server should respond to a POST to this URL by returning ***HTTP 200*** with '***OK***' in the message body.	Yes	Any length (alphanumeric)
success_url	The URL to which the customer will be redirected to in case of successful payment	Yes	Any length (alphanumeric)
error_url	The URL to which the payer will be redirected to in case of unsuccessful payment	Yes	Any length (alphanumeric)

7.1.3.1. Response example

```
{
    "result": {
        "identification": {
            "transactionid": "MerchantAssignedID",
            "uniqueid": "e8166b8c52704389ba9b2fa78839aed9",
            "shortid": "7292.4803.9309",
            "customerid": "CustomerID"
        },
        "method": "ideal",
        "type": "preauthorization",
        "level": 0,
        "code": 0,
        "message": "waiting",
        "processing": {
            "timestamp": "2014-01-16T11:19:24+00:00",
            "redirecturl": "https://idealtest.secure-ing.com/ideal/
issuerSim.do?trxid=0050000080688526&ideal=prob"
        },
        "merchant": {
            "key0": "Value0",
            "key1": "Value1",
            "key2": "Value2"
        }
    },
    "id": 1,
    "jsonrpc": "2.0"
}
```

You can use the ***redirecturl*** provided in the response to redirect the customer to the iDEAL payment page, where they can complete their payment.

7.2. Paysafecard

Payment method enabling customers to pay online using prepaid vouchers. Vouchers are for fixed denominations and can be combined.

7.2.1. Preauthorization (Paysafecard)

This request obtains authorization for the online voucher payment. Below is an example of the request.

```
<?php
require_once 'lib/SkrillPSP.php';

try {

// Create the Paysafecard request object
       $pay = new SkrillPsp_PaySafeCardPA();


// Set service endpoint
       $pay->setUri("https://psp.sandbox.dev.skrillws.net/v1/json/3e40a821/
channelid_psc_psp/paysafecard");

// Set request parameters
$pay->setParameters(array(
        'identification' => array(
            'transactionid' => '',
            'customerid' => ''
        ),
        'payment' => array(
            'amount' => '56',
            'currency' => 'eur'
        ),
        'account' => array(
            'country_restriction' => '',
            'kyclevel' => ''
        ),
        'frontend' => array(
            'responseurl' => 'http://merchant.com/response.php',
            'successurl' => 'http://merchant.com/success.php',
            'errorurl' => 'http://merchant.com/error.php'
        ),
        'customer' => array(
            'name' => array(
                'salutation' => 'Mr',
                'title' => 'Dr',
                'given' => '',
                'family' => '',
                'company' => ''
            ),
            'address' => array(
                'street' => '12 Street,
                'zip' => '1234',
                'city' => Berlin',
                'state' => 'Berlin',
                'country' => 'DE'
            ),
            'contact' => array(
                'email' => 'in@in.de',
                'ip' => '123.255.255.255'
            )
        )
    ));

// Displays the request in JSON format
       echo $pay->showJson();

// Make Paysafecard request
```

```
      $result = $pay->makeCall();

// check response object for success or error
      if($result->isSuccess()) {

      // work with result method and properties
          $identity = $result->getIdentification();
          $payment = $result->getPayment();
          echo $token = $result->getTokenFromAccount();
          echo $result->type;
          echo "<br />";
          echo $result->message;
          echo "<br />";
          echo $result->method;
          echo $result->getCode();
          echo "<br />";
          echo $result->getLevel();
      }
      else {

      // work with error data
         $data = $result->getErrorData();
         $errorData = $result->getError();
         var_dump($data);
         echo $errorData->code;
         echo $errorData->message;
         echo $result->getErrorLevel();
         echo "<br />";
         echo $result->getErrorMessage();
         echo $result->getErrorCode();
         echo $result->getErrorDataMessage();
         echo $result->getId();
         echo "<br />";
         foreach ($data as $val)
         {
          echo "$val<br />";
         }
}
catch (SkrillPsp_Exception $e) {
    echo $e->getMessage();
}
```

Below are further details of the request objects used with this payment method.

Table 7-4: Request objects used with Paysafecard preauthorization

Request Object	Description
new SkrillPsp_PaysafecardPA()	Creates the Paysafecard request object.
setUri()	Sets the API endpoint for this payment method. You must include your merchant ID and channel ID.
setParameters	Sets the parameters to be included in the preauthorization request. A description of the mandatory parameters is included below.
showJson ()	Displays the list of returned response in JSON format.
makeCall()	Invokes the request to Paysafecard.

Table 7-5: Parameters used with Paysafecard preauthorization

Field Name	Description	Required	Length / format
transaction_id	A free text field included by you to track the transaction. This should be unique.	Yes	0-256 (alphanumeric)
amount	Transaction amount in the specified currency. There is no decimal separator. This number must be at least two digits. The API is built on a cent-based ISO8583 structure that does not cater for decimals (e.g., "**.**" or "**,**"). For example: 10 = ten Euro cents, 100 = 1 Euro, 1000 = 10 Euros.	Yes	2-12 (string or integer)
currency	Currency that the transaction is processed in. Currency code is according to the ISO 4217 specification.	Yes	3 (alpha)
locale	Locale of the payment page (e.g., "en", "de",).	No	2 (alpha)
country	Lists the countries where Paysafecard is supported (e.g., DE, FR, GR, AT).	No	2 letter ISO country code
country_ restriction	Lists any countries you want to be excluded.	No	2 letter ISO country code
minimum_age	Restricts the Paysafecard user account to users of this age or above.	No	2 letter number.
kyc_level	Defines the level of customer verification required. Options are: • **simple** – text and email • **document** – proof of identity and address • **postident** – Germany only, full proof of identity, plus pick up form at post office	No	Alpha
response_url	Enables the response data to be redirected to Skrill. **Note**: this must be in HTTPS format. Your server should respond to a POST to this URL by returning ***HTTP 200*** with '***OK***' in the message body.	Yes	Any length (alphanumeric)
success_url	The URL to which the customer will be redirected to for a successful payment.	Yes	Any length (alphanumeric)
error_url	The URL to which the payer will be redirected to in case of unsuccessful payment.	Yes	Any length (alphanumeric)

7.2.1.1. Skrill response

Below is an example of the response to the ***preauthorization*** request.

```
{
    "result": {
        "identification": {
            "transactionid": "MerchantAssignedID",
            "uniqueid": "df62b42b0a4a41f5abfe271ea53170f1",
            "shortid": "3731.6399.6317",
            "customerid": "customerid 12345"
        },
        "method": "paysafecard",
        "type": "preauthorization",
        "level": 0,
        "code": 0,
        "message": "new",
        "processing": {
            "timestamp": "2014-01-16T12:36:17+00:00",
            "redirecturl": "https://customer.test.at.paysafecard.com/psccustomer/
GetCustomerPanelServlet?mid=1000004553&mtid=373163996317&amount=1.00&currency=EUR"
        },
        "merchant": {
            "key0": "Value0",
            "key1": "Value1",
            "key2": "Value2"
        }
    },
    "id": 1,
    "jsonrpc": "2.0"
}
```

Table 7-6: Paysafecard response parameters

Parameter	Description	Length / format
transaction_id	The unique value you provided in the request.	0-256 (alphanumeric)
unique_id	This value is a reference that is generated by Skrill's payment platform to uniquely identify the transaction.	32 (alpha)
method	This value indicates the method used in the original transaction request. For example: "***paysafecard***".	Any length
type	An abbreviation on the type of request that was invoked in your request. For example: "***debit***"	2-3 (alpha)
alias	The type of request that was invoked. For example: "***preauthorization***"	Any length
level	This value depicts the level that the response ID is coming from: 0 = Payment Service Level (ACK response) 1 = Format Error (frontend validation errors) 2 = System Error (configuration/connectivity errors) 3 >= In the event of an error the level value will depict what acquirer the error resides with (NOK response)	1-2 (numeric)
code	A value that indicates the status of the transaction. For example: "***0***"	1-2 (numeric)
message	A message that describes the status of the transaction. For example: "***new***".	128 (alphanumeric)

Table 7-6: Paysafecard response parameters

Parameter	Description	Length / format
timestamp	The date and time stamp of the transaction in the following format: *YYYY-MM-DD HH:MM:SS*	19 (alphanumeric)
redirect_url	Used in the return response following the direct payment request. Your web servers should use this response to redirect the customer to the Paysafecard website.	No length restriction (alphanumeric)

7.2.2. Testing

You can use the following details to test your Paysafecard transactions.

Table 7-7: Paysafecard PINs

PIN	Amount	Currency
8888888888804380	500	eur
8888888883004381	500	skk
8888888883104381	10000	skk
30000000000043805	100	eur
5000000000004380	51.13	eur

Test login credentials

- User: ***skrilltest01***
- Password: ***Paysafe1!***

7.3. Skrill Direct

Skrill Direct is an instant online payment method that enables customers to pay directly from their online banking account.

7.3.1. Preauthorization (Skrill Direct)

This request obtains authorization for the online bank transfer. Below is an example of the request.

```
<?php
require_once 'lib/SkrillPSP.php';

try {

// Create SkrillDirect request object
       $skrill = new SkrillPsp_SkrillDirect();


// Set service endpoint
       $skrill->setUri("https://psp.sandbox.dev.skrillws.net/v1/json/3e40a821/
channelid_skrilldirect/skrilldirect");

// Set request parameters

$skrill->setParameters(array(
        'payment' => array(
            'amount' => '112',
            'currency' => 'EUR',
            'descriptor' => 'comment',
            'recipient' => ''
        ),
        'account' => array(
            'holder' => 'L. Manelli',
            'accountnumber' => '797979',
            'routingnumber' => '6767676'
        ),
        'frontend' => array(
            'responseurl' => 'http://merchant.com/response.php',
            'successurl' => 'http://merchant.com/success.php',
            'errorurl' => 'http://merchant.com/error.php'
        ),
        'customer' => array(
            'name' => array(
                'firstname' => Lisa',
                'lastname' => Manelli'
            ),
            'address' => array(
                'street' => 54 Road',
                'zip' => '34343',
                'city' => 'Berlin',
                'country' => 'DE'
            ),
            'contact' => array(
                'email' => 'in@in.de',
                'ip' => '124.20.2.2'
            )
        )
    ));

// Display the request in JSON format
       echo $skrill->showJson();


// Make SkrillDirect request
       $result = $skrill->makeCall();

// check response object for success or error
       if($result->isSuccess()) {
```

```
      // work with result method and properties
          $identity = $result->getIdentification();
          $payment = $result->getPayment();
          echo $token = $result->getTokenFromAccount();
          echo $result->type;
          echo "<br />";
          echo $result->message;
          echo "<br />";
          echo $result->method;
          echo $result->getCode();
          echo "<br />";
          echo $result->getLevel();
      }
      else {

      // work with error data
         $data = $result->getErrorData();
         $errorData = $result->getError();
         var_dump($data);
         echo $errorData->code;
         echo $errorData->message;
         echo $result->getErrorLevel();
         echo "<br />";
         echo $result->getErrorMessage();
         echo $result->getErrorCode();
         echo $result->getErrorDataMessage();
         echo $result->getId();
         echo "<br />";
         foreach ($data as $val)
         {
          echo "$val<br />";
         }
}
catch (SkrillPsp_Exception $e) {
    echo $e->getMessage();
}
```

Below are further details of the request objects used with this payment method..

Table 7-8: Request objects used with Skrill Direct preauthorization

Request Object	Description
new SkrillPsp_SkrillDirect()	Creates the Skrill Direct request object.
setUri()	Sets the API endpoint for this payment method. You must include your merchant ID and channel ID.
setParameters	Sets the parameters to be included in the preauthorization request. A description of the mandatory parameters is included below.
showJson ()	Displays the list of returned response in JSON format.
makeCall()	Invokes the request to Skrill Direct.

Table 7-9: Skrill Direct payment parameters

Field Name	Description	Required	Length / format
transaction_id	A free text field included by you to track the transaction. This should be unique.	Yes	0-256 (alphanumeric)

Table 7-9: Skrill Direct payment parameters

Field Name	Description	Required	Length / format
amount	Transaction amount in the specified currency.	Yes	2-12 (string or integer)
currency	Currency identifier (Alpha-3 ISO 4217 code).	Yes	3 (alpha)
descriptor	Comment or description of the transaction.	Yes	1-256 (alpha)
recipient	You can add an additional merchant description here.	No	1-256 (Alphanumeric)
account_ holder	The account holder's name.	Yes	(alphanumeric)
Account_ number	The customer's account number.	Yes	(numeric)
routing_ number	The customer's routing number or sort code.	Yes	6 digits (numeric)
country	The bank country.	Yes	2 digit ISO code
response_url	Enables the response data to be redirected to Skrill. **Note**: this must be in HTTPS format. Your server should respond to a POST to this URL by returning ***HTTP 200*** with '***OK***' in the message body.	Yes	Any length (alphanumeric)
success_url	The URL to which the customer will be redirected to in case of successful payment.	Yes	Any length (alphanumeric)
error_url	The URL to which the payer will be redirected to in case of unsuccessful payment.	Yes	Any length (alphanumeric)

7.3.1.1. Response example

The example below shows the response returned from Skrill.

```
{
   "result":    {
      "identification":       {
         "transactionid": "transactionID",
         "uniqueid": "f29331af11914ab297677de3b3a47404",
         "shortid": "1379.4044.4186",
         "customerid": "customerID"
      },
      "method": "skrilldirect",
      "type": "preauthorization",
      "level": 0,
      "code": 0,
      "message": "waiting",
      "processing":       {
         "timestamp": "2013-09-17T07:45:26+00:00",
         "redirecturl": "https://payment.onlinebanktransfer.com/secure/payment/
?ngpID=1000001523607&hash=Uk1PcXdvYkRyc0s5U2NPT01CRld3NXhPQ1NuQ29jTyt3NzFmdzZVMA=="
      }
   },
   "id": 1,
```

```
   "jsonrpc": "2.0"
}
```

7.3.2. Testing

You can use the following parameters to test your Skrill Direct integration:

Table 7-10:

Account/routing number	PIN	TIN
56341200	123456	111111

7.4. Skrill Wallet

The Skrill Digital Wallet enables customers to pay using their bank account, debit and credit cards, without having to enter any sensitive details online. Customers also have access to a wide range of alternative payment methods.

7.4.1. Preauthorization (Skrill Wallet)

This request obtains authorization for the Skrill Wallet payment. Below is an example of the request. Parameters unique to Skrill Wallet are highlighted.

```
<?php
require_once 'lib/SkrillPSP.php';

try {

// Create SkrillWallet PA request object
       $obj = new SkrillPsp_SkrillWalletPreauthorization();

// Set service endpoint
       $obj->setUri("https://psp.sandbox.dev.skrillws.net/v1/json/3e40a821/
channelid_skrillwallet/skrillwallet");

//Set request parameters
       $obj->setParameters(array(
         'identification' => array(
             'transactionid' => 'RT126',
             'customerid' => 'TYY66'
         ),
         'frontend' => array(
             'responseurl' => 'http://www.merchant.com/response.php',
             'successurl' => 'http://merchant.com/success.php',
             'successurl_url_text' => 'Return to sample merchant',
             'errorurl' => 'http://merchant.com/error.php',
             'language' => 'EN',
             'errorurl_url_target' => '4',
             'new_window_redirect' => '1',
             'ext_ref_id' => '12345',
             'confirmation_note' => Confirmation comments',
             'logo_url' => 'http://merchant.com/logo.gif',
             'rid' => '2323',
             'detail_description1' => 'test description',
             'detail_text1' => 'test details
         ),
         'payment' => array(
             'amount' => '500',
             'currency' => 'EUR',
             'descriptor' => 'Your order details'
         ),
         'customer' => array(
```

```
            'name' => array(
                'firstname' => 'Nick',
                'lastname' => Smith
            ),
            'address' => array(
                'street' => 21 Lane',
                'zip' => '45454',
                'city' => 'Birmingham',
                'country' => 'UK'
            ),
            'contact' => array(
                'email' => 'mb654@abv.bg',
                'ip' => '124.0.0.12'
            )
        )
    ));
// Display the request in JSON format
       echo $obj->showJson();

// Make SkrillWallet PA request
        $result = $obj->makeCall();

// check response object for success or error
       if($result->isSuccess()) {

       // work with result method and properties
           $identity = $result->getIdentification();
           $payment = $result->getPayment();
           echo $token = $result->getTokenFromAccount();
           echo $result->type;
           echo "<br />";
           echo $result->message;
           echo "<br />";
           echo $result->method;
           echo $result->getCode();
           echo "<br />";
           echo $result->getLevel();
       }
       else {

       // work with error data
          $data = $result->getErrorData();
          $errorData = $result->getError();
          var_dump($data);
          echo $errorData->code;
          echo $errorData->message;
          echo $result->getErrorLevel();
          echo "<br />";
          echo $result->getErrorMessage();
          echo $result->getErrorCode();
          echo $result->getErrorDataMessage();
          echo $result->getId();
          echo "<br />";
          foreach ($data as $val)
          {
           echo "$val<br />";
          }
}
catch (SkrillPsp_Exception $e) {
    echo $e->getMessage();
}
```

Below are further details of the request objects used with this payment method..

Table 7-11: Request objects used with Skrill Wallet preauthorization

Request Object	Description
new SkrillPsp_SkrillWalletPreauthorization()	Creates the Skrill Wallet request object.
setUri()	Sets the API endpoint for this payment method. You must include your merchant ID and channel ID.
setParameters	Sets the parameters to be included in the preauthorization request. A description of the mandatory parameters is included below.
showJson ()	Displays the list of returned response in JSON format.
makeCall()	Invokes the request to Skrill Wallet.

Table 7-12: Skrill Wallet preauthorization parameters

Field Name	Description	Required	Length / format
transaction_id	A free text field included by you to track the transaction. This should be unique.	Yes	0-256 (alphanumeric)
amount	Transaction amount in the specified currency. There is no decimal separator. This number must be at least two digits. The API is built on a cent-based ISO8583 structure that does not cater for decimals (e.g., "**.**" or "**,**"). For example: 10 = ten Euro cents, 100 = 1 Euro, 1000 = 10 Euros.	Yes	2-12 (string or integer)
currency	Currency that the transaction is processed in. Currency code is according to the ISO 4217 specification.	Yes	3 (alpha)
descriptor	Comment to the invoice. Enables you to add a description to be displayed on the customer's bank statement.	Yes	1-256 (alpha)
account_holder	Skrill Wallet account holder name.	Yes	50 (alphanumeric)
recipient_description	A description to be shown on the Skrill Gateway page. If no value is submitted, the ***pay_to_email*** value is shown as the recipient of the payment.	No	30 (alphanumeric)
response_url	The URL to which Skrill sends the response data.	Yes	Any length (alphanumeric)
success_url	The URL to which the customer will be redirected to in case of successful payment.	Yes	Any length (alphanumeric)

Table 7-12: Skrill Wallet preauthorization parameters

Field Name	Description	Required	Length / format
error_url	The URL to which the customer will be redirected to in case of unsuccessful payment	Yes	Any length alphanumeric)
error_url_url_target	The target in which the return_url value is displayed upon an error. Default value is 1. 1 = '_top' 2 = '_parent' 3 = '_self' 4= '_blank'		
new_window_redirect	You can redirect customers a new window instead of in the same browser window (e.g., for online bank transfer payment methods).	No	0 (default) or 1(new window).
language	2-letter code of the language used for Skrill's pages. Can be any of EN, DE, ES, FR, IT, PL, GR RO, RU, TR, BG, CN, CZ, NL, DA, SV or FI.	Yes	2-letter code
confirmation_note	This enables you to display a confirmation message or other details at the end of the payment process. Line breaks *** *** can be used for longer messages.	No	240 characters (alphanumeric)
logo_url	The URL of the logo which you would like to appear at the top of the Skrill page. The logo must be accessible via HTTPS or it will not be shown. For best results use logos with dimensions up to 200px in width and 50px in height.	No	240 characters (alphanumeric)
rid	You can pass a unique referral ID or email of an affiliate from which the customer is referred. The rid value must be included within the actual payment request.	No	100 characters (alphanumeric)
ext_ref_id	You can pass additional identifier in this field in order to track your affiliates. You must inform your account manager about the exact value that will be submitted so that affiliates can be tracked.	No	100 characters (alphanumeric)
payment_methods	A comma-separated list of payment method codes, indicating the payment methods to be presented to the customer. If included, the customer is presented with the selected payment methods and their corresponding logos.	No	3-digit code.

Table 7-12: Skrill Wallet preauthorization parameters

Field Name	Description	Required	Length / format
detail_description	Enables you to add additional details about the product in the 'More information' section in the header of the Skrill Gateway page.	No	240 characters (alphanumeric)
detail_text	Shown next to the ***detail_description*** and is also shown to the customer in their Skrill Digital Wallet account history.	No	240 characters (alphanumeric)

7.4.1.1. Preauthorisation response

Below is an example of the preauthorization response.

```
{
   "result":    {
      "identification":       {
         "transactionid": "MerchantAssignedID",
         "uniqueid": "83ed01cce74ac43061b5ab3c91e51dad",
         "shortid": "0019.6099.9707"
      },
      "level": "14",
      "code": "0",
      "method": "skrillwallet",
      "type": "preauthorization",
      "message": "new",
      "processing":       {
         "timestamp": "2013-03-12 15:54:11",
         "redirecturl": "https://www.skrill.com/app/
payment.pl?sid=6e1ba0c61c1e547bcd4cc37b2c567ccc"
      }
   },
   "id": 1,
   "jsonrpc": "2.0"
}
```

7.4.2. Skrill Wallet refund

This request can be used to issue a refund for a Skrill Wallet payment. Below is an example of the request. Parameters unique to Skrill Wallet refunds are highlighted.

```
?php
require_once 'lib/SkrillPSP.php';
try {

// Create Skrill Wallet Refund request object
   $refund = new SkrillPsp_SkrillWalletRefund("83ed01cce74ac43061b5ab3c91e51dad");

// Set service endpoint
       $refund->setUri("https://psp.sandbox.dev.skrillws.net/v1/json/3e40a821/
        channelid_skrillwallet/skrillwallet");

//Set request parameters
           $refund->setParameters(array(
         'identification' => array(
             'transactionid' => 'RT126'
         ),
         'payment' => array(
             'amount' => '600',
             'currency' => 'eur',
             'descriptor' => 'wallet refund'
         )
     ));
```

```
// Display the request in JSON format
       echo $refund->showJson();

// Make Skrill Wallet Refund call
       $result = $refund->makeCall();

// check response object for success or error
       if($result->isSuccess()) {

       // work with result method and properties
           $identity = $result->getIdentification();
           $payment = $result->getPayment();
           echo $token = $result->getTokenFromAccount();
           echo $result->type;
           echo "<br />";
           echo $result->message;
           echo "<br />";
           echo $result->method;
           echo $result->getCode();
           echo "<br />";
           echo $result->getLevel();
       }
       else {

       // work with error data
          $data = $result->getErrorData();
          $errorData = $result->getError();
          var_dump($data);
          echo $errorData->code;
          echo $errorData->message;
          echo $result->getErrorLevel();
          echo "<br />";
          echo $result->getErrorMessage();
          echo $result->getErrorCode();
          echo $result->getErrorDataMessage();
          echo $result->getId();
          echo "<br />";
          foreach ($data as $val)
          {
           echo "$val<br />";
          }
}
catch (SkrillPsp_Exception $e) {
    echo $e->getMessage();
}
```

Below are further details of the request objects and parameters used with this payment method..

Table 7-13: Request objects used with Skrill Wallet refund

Request Object	Description
new SkrillPsp_SkrillWalletRefund()	Creates the Skrill Wallet refund request object. Your argument should include the unique ID for returned with the original transaction. For example: ***new SkrillPsp_SkrillWalletRefund ("83ed01cce74ac43061b5ab3c91e51dad");***
setUri()	Sets the API endpoint for this payment method. You must include your merchant ID and channel ID.
setParameters	Sets the parameters to be included in the preauthorization request. A description of the mandatory parameters is included below.
showJson ()	Displays the list of returned response in JSON format.

Table 7-13: Request objects used with Skrill Wallet refund

makeCall()	Invokes the refund request to Skrill Wallet.

Table 7-14: Skrill Wallet refund parameters

Field Name	Description	Required	Length / format
channel_id	Identifier of the channel in which you are processing payments.	Yes (if not globally defined)	1-128 (alphanumeric)
transaction_id	A free text field included by you to track the transaction. This should be unique.	Yes	0-256 (alphanumeric)
reference_id	Used to refer to a previous transaction. This must be the unique ID generated by Skrill to track the transaction.	Yes	32 (alphanumeric)
amount	Refund amount in the specified currency. There is no decimal separator. This number must be at least two digits.	Yes	2-12 (string or integer)
currency	Currency that the refund is processed in. Currency code is according to the ISO 4217 specification.	Yes	3 (alpha)
descriptor	Enables you to add a description to be displayed on the customer's bank statement.	Yes	1-256 (alpha)

7.4.2.1. Refund response

Below is an example of the response to the Skrill Wallet refund request.

```
{
   "result":    {
      "identification":       {
         "transactionid": "MerchantAssignedID",
         "uniqueid": "0a8480bc0b8646ec8c1e6ccbd7ca71d5",
         "shortid": "1322.1463.4193"
      },
      "method": "skrillwallet",
      "type": "refund",
      "level": 0,
      "code": 0,
      "message": "approved",
      "clearing":       {
         "amount": "08",
         "currency": "EUR",
         "descriptor": "wallet refund"
      },
      "processing": {"timestamp": "2013-11-25T15:00:04+00:00"}
   },
   "id": 1,
   "jsonrpc": "2.0"
}
```

7.4.3. Skrill Wallet credit (Send Money)

This request can be used to send money or make a payout to the customer's Skrill Wallet account. Below is an example of the request. Parameters unique to a Skrill Wallet credit are highlighted.

```
<?php
require_once 'lib/SkrillPSP.php';

try {

// Create SkrillWallet SendMoney request object
       $send = new SkrillPsp_SkrillWalletCredit();

// Set service endpoint
       $send->setUri('https://psp.sandbox.dev.skrillws.net/v1/json/3e40a821/
channelid_skrillwallet/skrillwallet');

//Set request parameters
$send->setParameters(array(
        'identification' => array(
            'transactionid' => 'TY6788',
            'customerid' => '123456'
        ),
        'payment' => array(
            'amount' => '203',
            'currency' => 'eur',
            'subject' => 'Your payment,
            'note' => details'
        ),
        'customer' => array(
            'contact' => array(
                'email' => 'nb@di.nf',
                'ip' => '127.0.0.1'
            )
        )
    ));

// Display the request in JSON format
       echo $send->showJson();

// Make Skrill Wallet Credit request
       $result = $send->makeCall();

// check response object for success or error
       if($result->isSuccess()) {

       // work with result method and properties
           $identity = $result->getIdentification();
           $payment = $result->getPayment();
           echo $token = $result->getTokenFromAccount();
           echo $result->type;
           echo "<br />";
           echo $result->message;
           echo "<br />";
           echo $result->method;
           echo $result->getCode();
           echo "<br />";
           echo $result->getLevel();
       }
       else {

       // work with error data
          $data = $result->getErrorData();
          $errorData = $result->getError();
          var_dump($data);
          echo $errorData->code;
          echo $errorData->message;
          echo $result->getErrorLevel();
          echo "<br />";
          echo $result->getErrorMessage();
```

```
          echo $result->getErrorCode();
          echo $result->getErrorDataMessage();
          echo $result->getId();
          echo "<br />";
          foreach ($data as $val)
          {
           echo "$val<br />";
          }
}
catch (SkrillPsp_Exception $e) {
    echo $e->getMessage();
}
```

Below are further details of the request objects and parameters used with this payment method.

Table 7-15: Skrill Wallet Credit parameters

Field Name	Description	Required	Length / format
transaction_id	A free text field included by you to track the transaction. This should be unique.	Yes	0-256 (alphanumeric)
amount	The amount to send to the customer. The minimum amount is "***01***".	Yes	2-12 (string or integer)
currency	The currency of the transaction. The amount will be sent in this currency. Provide the 3-digit 3 ISO 4217 currency code.	Yes	3-digit currency code
descriptor	You can add a brief subject for the credit request. This is displayed in the email subject field of the message sent to the customer.	Yes	16-25 (alphanumeric)
note	You can add a more detailed description of the transaction, which is displayed in the body of the message sent to the customer.	Yes	
firstname	Customer's first name.	No	2-40 (alpha)
lastname	The customer's surname.	No	2-40 (alpha)
IP	The IP address of the customer.	No	15 (alphanumeric)
email	The customer or user's email address.	Yes	6-128 (alphanumeric)

7.4.3.1. Credit response

Below is an example of the response to the Skrill Wallet credit request.

```
{
    "result": {
        "identification": {
            "transactionid": "Marcel 0002",
            "uniqueid": "f29c7caca06e40c88ef67337e0e6a9c9",
            "shortid": "5067.9576.6150",
            "customerid": "customerid 12345"
        },
        "method": "skrillwallet",
        "type": "credit",
        "level": 0,
        "code": 0,
        "message": "approved",
        "clearing": {
```

```
            "amount": "1.20",
            "currency": "EUR"
        },
        "processing": {
            "timestamp": "2013-11-27T10:00:36+00:00"
        }
    },
    "id": 1,
    "jsonrpc": "2.0"
}
```

7.4.3.2. Credit response for non-registered user

Below is an example of the response to the credit request.

```
{
    "result": {
        "identification": {
            "transactionid": "Marcel 0002",
            "uniqueid": "0ac8ab8492814a6caf1816efbca8e4a6",
            "shortid": "2681.8077.0807",
            "customerid": "customerid 12345"
        },
        "method": "skrillwallet",
        "type": "credit",
        "level": 1,
        "code": 518,
        "message": " skrill Wallet send money status PENDING",
        "advice": " Declined - beneficiary's email address is not registered"
    },
    "id": 1,
    "jsonrpc": "2.0"
}
```

7.4.4. Skrill 1-Tap register request

Below is an example of the preauthorization request to set up a 1-Tap payment. You must be enabled for one-tap to use this service.

```
<?php
require_once 'lib/SkrillPSP.php';

try {
    // Create OneTap Register object
    $one = new OneTap_Register();

    // Set service endpoint
    $one->setUri('https://psp.sandbox.dev.skrillws.net/v1/json/3e40a821/
channelid_skrillwallet/skrillwallet');

    // Set request parameters
    $one->setParameters(array(
        'identification' => array(
            'transactionid' => 'RT5',
            'customerid' => 'TYy6868'
        ),
        'payment' => array(
            'amount' => '1984',
            'currency' => 'eur',
            'descriptor' => 'descriptor line',
            'payment_methods' => array(
                'SFT',
                'LSR'
            )
        ),
        'frontend' => array(
            'amount_details' => array(
                array(
                    "86",
                    "amount1 desc1"
                ),
                array(
                    "678",
                    "amount2 desc"
                )
            ),
            'responseurl' => 'https://merchant.com/response.php',
            'successurl' => array(
                'url' => 'http:// merchant.com/response.php',
                'text' => 'Return to sample merchant',
                'target' => '4'
            ),
            'errorurl' => array(
                'url' => 'http:// merchant.com/response.php',
                'target' => '4'
            ),
            'new_window_redirect' => "1",
            'language' => 'EN',
            'hide_login' => '1',
            'confirmation_note' => 'Thanks for shopping with us',
            'logo_url' => 'https:// merchant.com /logo.png',
            'rid' => '123445',
            'ext_ref_id' => 'AffiliateName',
            'detail_descriptions' => array(
                array(
                    "detail description",
                    "detail text"
                ),
                array(
                    "detail description",
                    "detail text"
                )
            )
        ),
```

```
        'customer' => array(
            'name' => array(
                "salutation" => "Mr",
                "title" => "Mr",
                "firstname" => "John",
                "lastname" => "Doe",
                "company" => "Skrill"
            ),
            'address' => array(
                "street" => "Karl-Liebknecht-Strasse 5",
                "zip" => "10117",
                "city" => "Berlin",
                "state" => "BE",
                "contry" => "UK"
            ),
            'contact' => array(
                'phone' => '+49302341423',
                'mobile' => '+49 172 931 44 01',
                'email' => 'mb654@abv.bg',
                'ip' => '127.0.0.1'
            )
        ),
        'merchant' => array(
            'product' => 'Nokia'
        )
    ));

    // Allows you to view request in raw json format
    echo $one->showJson();

    // Make OneTap Register request
    $result = $one->makeCall();

    // check response object for success or error
       if($result->isSuccess()) {

       // work with result method and properties
           $identity = $result->getIdentification();
           $payment = $result->getPayment();
           echo $token = $result->getTokenFromAccount();
           echo $result->type;
           echo "<br />";
           echo $result->message;
           echo "<br />";
           echo $result->method;
           echo $result->getCode();
           echo "<br />";
           echo $result->getLevel();
       }
       else {

       // work with error data
          $data = $result->getErrorData();
          $errorData = $result->getError();
          var_dump($data);
          echo $errorData->code;
          echo $errorData->message;
          echo $result->getErrorLevel();
          echo "<br />";
          echo $result->getErrorMessage();
          echo $result->getErrorCode();
          echo $result->getErrorDataMessage();
          echo $result->getId();
          echo "<br />";
          foreach ($data as $val)
          {
           echo "$val<br />";
          }
}
catch (SkrillPsp_Exception $e) {
    echo $e->getMessage();
```

```
}
```

Skrill 1-Tap parameters

The table below lists the parameters that should be included in the 1-Tap request.

Table 7-16: Skrill Wallet 1-Tap parameters

Field Name	Description	Required	Length / format
remote_method	The type of API request. For example, 'onetap'.	Yes	String
amount	The amount to send to the customer. The minimum amount is "***01***".	Yes	2-12 (string or integer)
currency	The currency of the transaction. The amount will be sent in this currency. Provide the 3-digit 3 ISO 4217 currency code.	Yes	3-digit currency code (upper or lower-case)
ondemand_ max_amount	Maximum amount for future payments that will be debited from the customer's account.	Yes	9
ondemand_ note	Text shown to the customer on the confirmation page as the reason for the Skrill 1-Tap payment.	Yes	1000
responseurl	The URL to which Skrill sends the response data. **Note**: this must be in HTTPS format. Your server should respond to a POST to this URL by returning ***HTTP 200*** with '***OK***' in the message body.	Yes	Any
successurl	The URL to which the customer will be redirected to for a successful payment. You can include a text description and the URL target value, defining the area on your website where the response is displayed. 1 = '_top', 2 = '_parent' 3 = '_self', 4= '_blank'	Yes	Any
errorurl	The URL to which the customer will be redirected for an unsuccessful payment. You can include a URL target value.	Yes	Any
language	2-letter code of the language used for the payment pages. Can be any of EN, DE, ES, FR, IT, PL, GR RO, RU, TR, CN, BG, CZ, NL, DA, SV or FI.	Yes	2
email	The customer or user's email address.	Yes	6-128 (alphanumeric)

Notes

In addition to the above parameters, you can add an amount and text description to your request by including the following parameters:

- amount_details

- detail_descriptions

7.4.4.1. Skrill 1-Tap response

You can use the reference ID returned in the response as the payment token for processing future transactions for this cardholder.


```
{
   "result":    {
      "identification":       {
         "transactionid": "c16c3378c6c34ed1be1c692b4d087bc0",
         "uniqueid": "4707179ff2ca4a32bef006ac32ce8a27",
         "shortid": "8663.2679.2135"
      },
      "method": "skrillwallet",
      "type": "onetap",
      "level": 0,
      "code": 0,
      "message": "waiting",
      "processing":       {
         "timestamp": "2013-12-20T11:27:38+00:00",
         "redirecturl": "https://www.moneybookers.com/app/
payment.pl?sid=e758cb49b2ec9b40464156b3b54bbe8f"
      }
   },
   "id": 1,
   "jsonrpc": "2.0"
}
```

7.4.5. Using the Skrill 1-Tap token

A payment token is provided in the transaction details returned to your **responseurl** page after a successful 1-Tap request. You can use the token to process further payments from the customer, as shown in the example below.

```
<?php
try {
    // Create OneTap debit object
    $onetap = new OneTap_Debit("b316cb82490044979e8e19345e2bce24");

    // Set service endpoint
    $onetap->setUri("https://psp.sandbox.dev.skrillws.net/v1/json/3e40a821/
channelid_skrillwallet/skrillwallet");

    // Set request parameters
    $onetap->setParameters(array(
        'identification' => array(
            'transactionid' => 'Tyjyj23',
            'customerid' => 'yuyuuy'
        ),
        'payment' => array(
            'amount' => '567',
            'currency' => 'eur',
            'descriptor' => 'dkdkdk'
        ),
        'account' => array(
            'token' => 'b316cb82490044979e8e19345e2bce24'
        )
    ));

    // Allows you to view request in raw json format
    echo $one->showJson();

    // Make OneTap Debit request
    $result = $onetap->makeCall();

    // check response object for success or error
       if($result->isSuccess()) {

       // work with result method and properties
           $identity = $result->getIdentification();
           $payment = $result->getPayment();
           echo $token = $result->getTokenFromAccount();
           echo $result->type;
           echo "<br />";
           echo $result->message;
           echo "<br />";
           echo $result->method;
           echo $result->getCode();
           echo "<br />";
           echo $result->getLevel();
       }
       else {

       // work with error data
          $data = $result->getErrorData();
          $errorData = $result->getError();
          var_dump($data);
          echo $errorData->code;
          echo $errorData->message;
          echo $result->getErrorLevel();
          echo "<br />";
          echo $result->getErrorMessage();
          echo $result->getErrorCode();
          echo $result->getErrorDataMessage();
          echo $result->getId();
          echo "<br />";
          foreach ($data as $val)
          {
```

```
          echo "$val<br />";
         }
}
catch (SkrillPsp_Exception $e) {
    echo $e->getMessage();
}
```

7.4.5.1. Skrill 1-Tap token response

```
{
   "result":    {
      "identification":       {
         "transactionid": "MerchantAssignedID",
         "uniqueid": "8e707e07e447468eaa78fd30c25daae4",
         "shortid": "3916.1769.4186",
         "customerid": "customerid 12345",
         "referenceid": "b316cb82490044979e8e19345e2bce24"
      },
      "method": "skrillwallet",
      "type": "onetap",
      "level": 0,
      "code": 0,
      "message": "approved",
      "account": {"token": "b316cb82490044979e8e19345e2bce24"},
      "clearing":       {
         "amount": "10.80",
         "currency": "EUR"
      },
      "processing": {"timestamp": "2013-12-20T11:28:55+00:00"}
   },
   "id": 1,
   "jsonrpc": "2.0"
}
```

7.4.6. Skrill 1-Tap Status request

This option enables you to check the status of a Skrill 1-Tap request. You will need to include the one-tap token in your request, as shown below.

```
<?php
require_once 'lib/SkrillPSP.php';

try {
    // Create Onetap Status object
    $tab = new OneTap_Status("5504eaee2ef54e369ba620cb716426f1");

    // Set service endpoint
    $tab->setUri("https://psp.sandbox.dev.skrillws.net/v1/json/3e40a821/
channelid_skrillwallet/skrillwallet");

    // Set request parameters
    $tab->setParameters(array(
        'identification' => array(
            'transactionid' => 'TYYY',
            'customerid' => 'HJJJ67'
        ),
        'account' => array(
            'token' => 'fkfkfkfk'
        )
    ));

    // Allows you to view request in raw json format
    echo $tab->showJson();

    // Make OneTap Status request
    $result = $tab->makeCall();

    // check response object for success or error
       if($result->isSuccess()) {
```

```
      // work with result method and properties
          $identity = $result->getIdentification();
          $payment = $result->getPayment();
          echo $token = $result->getTokenFromAccount();
          echo $result->type;
          echo "<br />";
          echo $result->message;
          echo "<br />";
          echo $result->method;
          echo $result->getCode();
          echo "<br />";
          echo $result->getLevel();
      }
      else {

      // work with error data
         $data = $result->getErrorData();
         $errorData = $result->getError();
         var_dump($data);
         echo $errorData->code;
         echo $errorData->message;
         echo $result->getErrorLevel();
         echo "<br />";
         echo $result->getErrorMessage();
         echo $result->getErrorCode();
         echo $result->getErrorDataMessage();
         echo $result->getId();
         echo "<br />";
         foreach ($data as $val)
         {
          echo "$val<br />";
         }
}
catch (SkrillPsp_Exception $e) {
    echo $e->getMessage();
}
```

7.4.6.1. Skrill 1-Tap Status response

Below is an example of the response to a 1-Tap status request.

```
{
   "result":    {
      "identification":       {
         "transactionid": "c16c3378c6c34ed1be1c692b4d087bc0",
         "uniqueid": "de3df200e5a441d3b0089897ac66257d",
         "shortid": "6380.6105.7980"
      },
      "method": "skrillwallet",
      "type": "status",
      "level": 0,
      "code": 0,
      "message": "cancelled",
      "account": {"token": "b316cb82490044979e8e19345e2bce24"},
      "processing":       {
         "last_execution": " Last execution date: 20-12-2013\n",
         "timestamp": "2013-12-20T11:37:48+00:00"
      }
   },
   "id": 1,
   "jsonrpc": "2.0"
}
```

7.4.7. Skrill 1-Tap Cancel request

This option enables you to cancel the Skrill 1-Tap token set up in the Payment Gateway. You will need to include the one-tap token in your request, as shown below.

```
try {
    // Create OneTap cancel object
    $onetap = new OneTap_Cancel("5504eaee2ef54e369ba620cb716426f1");

    // Set service endpoint
    $onetap->setUri("https://psp.sandbox.dev.skrillws.net/v1/json/3e40a821/
channelid_skrillwallet/skrillwallet");

    // Set request parameters
    $onetap->setParameters(array(
        'identification' => array(
            'transactionid' => 'Tyjyj23',
            'customerid' => 'yuyuuy'
        ),
        'account' => array(
            'token' => 'fjfjfj'
        )
    ));

    // Allows you to view request in raw json format
    echo $one->showJson();

    // Make OneTap Cancel request
    $result = $onetap->makeCall();
    // check response object for success or error
       if($result->isSuccess()) {

       // work with result method and properties
           $identity = $result->getIdentification();
           $payment = $result->getPayment();
           echo $token = $result->getTokenFromAccount();
           echo $result->type;
           echo "<br />";
           echo $result->message;
           echo "<br />";
           echo $result->method;
           echo $result->getCode();
           echo "<br />";
           echo $result->getLevel();
       }
       else {

       // work with error data
          $data = $result->getErrorData();
          $errorData = $result->getError();
          var_dump($data);
          echo $errorData->code;
          echo $errorData->message;
          echo $result->getErrorLevel();
          echo "<br />";
          echo $result->getErrorMessage();
          echo $result->getErrorCode();
          echo $result->getErrorDataMessage();
          echo $result->getId();
          echo "<br />";
          foreach ($data as $val)
          {
           echo "$val<br />";
          }
}
catch (SkrillPsp_Exception $e) {
    echo $e->getMessage();
}
```

7.4.7.1. Skrill 1-Tap Cancel response

Below is an example of the response to a 1-Tap cancel request.


```
{
   "result":    {
      "identification":       {
         "transactionid": "c16c3378c6c34ed1be1c692b4d087bc0",
         "uniqueid": "e896f1bcc61f4dfeabe223eb719e1255",
         "shortid": "6876.3019.6103"
      },
      "method": "skrillwallet",
      "type": "cancel",
      "level": 0,
      "code": 0,
      "message": "approved",
      "account": {"token": "b316cb82490044979e8e19345e2bce24"},
      "processing": {"timestamp": "2013-12-20T11:29:22+00:00"}
   },
   "id": 1,
   "jsonrpc": "2.0"
}
```

7.4.8. Skrill Wallet Payment methods

The following payment methods are supported on Skrill Wallet:

Table 7-17: Skrill Wallet payment method codes

Payment Method	Value	Supported Countries
Wallet	WLT	ALL
Credit/Debit Cards		
All Card Types	ACC	ALL
Visa	VSA	ALL
MasterCard	MSC	ALL
Visa Delta/Debit	VSD	United Kingdom
Visa Electron	VSE	ALL
Maestro	MAE	United Kingdom, Spain & Austria
American Express	AMX	ALL
Diners	DIN	ALL
JCB	JCB	ALL
Carte Bleue	GCB	France
Dankort	DNK	Denmark
PostePay	PSP	Italy
CartaSi	CSI	Italy
Instant Banking Options		
Skrill Direct (Online Bank Transfer)	OBT	Germany, United Kingdom, France, Italy, Spain, Hungary and Austria
Giropay	GIR	Germany
Direct Debit / ELV	DID	Germany
Sofortueberweisung	SFT	Germany, Austria, Belgium, Netherlands, Switzerland & United Kingdom
eNETS	ENT	Singapore
Nordea Solo	EBT	Sweden
Nordea Solo	SO2	Finland
iDEAL	IDL	Netherlands
EPS (Netpay)	NPY	Austria
POLi	PLI	Australia
All Polish Banks	PWY	Poland
ING Bank Śląski	PWY5	Poland
PKO BP (PKO Inteligo)	PWY6	Poland

Table 7-17: Skrill Wallet payment method codes

Payment Method	Value	Supported Countries
Multibank (Multitransfer)	PWY7	Poland
Lukas Bank	PWY14	Poland
Bank BPH	PWY15	Poland
InvestBank	PWY17	Poland
PeKaO S.A.	PWY18	Poland
Citibank handlowy	PWY19	Poland
Bank Zachodni WBK (Przelew24)	PWY20	Poland
BGŻ	PWY21	Poland
Millenium	PWY22	Poland
mBank (mTransfer)	PWY25	Poland
Płacę z Inteligo	PWY26	Poland
Bank Ochrony Środowiska	PWY28	Poland
Nordea	PWY32	Poland
Fortis Bank	PWY33	Poland
Deutsche Bank PBC S.A.	PWY36	Poland
ePay.bg	EPY	Bulgaria

7.5. Yandex

Yandex.money is a popular eWallet and payment service, offered by Yandex, Russia's primary search engine, which is used by over 56 million users worldwide. Customers can make online payments in real-time, from their eWallet account, without paying a commission.

7.5.1. Preauthorization (Yandex)

This request obtains authorization for the online payment. Below is an example of the request.

```
<?php
require_once 'lib/SkrillPSP.php';
try {
    // Create Yandex request object
    $yandex = new SkrillPsp_Yandex();

    // Set service endpoint
    $yandex->setUri ("https://psp.sandbox.dev.skrillws.net/v1/json/3e40a821/
testchannel_yandex/yandex");

    // Set request parameters
    $yandex->setParameters(array(
        'identification' => array(
            'transactionid' => 'TY',
            'customerid' => '5666'
        ),
        'payment' => array(
            'amount' => '4444',
            'currency' => 'EUR',
            'country' => 'RU',
            'descriptor' => 'Description'
        ),
        'frontend' => array(
            'language' => 'EN',
            'responseurl' => 'https://merchant.com/wpf_response.php',
            'successurl' => 'https://merchant.com/success_url.php',
            'errorurl' => 'https:// merchant.com/error_url.php'
        ),
        'customer' => array(
            'contact' => array(
                'email' => 'test@dir.bg',
                'ip' => '129.0.0.10'
            )
        )
    ));

    // View request in JSON format
    echo $yandex->showJson();

    // Make Yandex request
    $result = $yandex->makeCall();

    // check response object for success or error
       if($result->isSuccess()) {

       // work with result method and properties
           $identity = $result->getIdentification();
           $payment = $result->getPayment();
           echo $token = $result->getTokenFromAccount();
           echo $result->type;
           echo "<br />";
           echo $result->message;
           echo "<br />";
           echo $result->method;
           echo $result->getCode();
           echo "<br />";
           echo $result->getLevel();
       }
```

```
      else {

      // work with error data
         $data = $result->getErrorData();
         $errorData = $result->getError();
         var_dump($data);
         echo $errorData->code;
         echo $errorData->message;
         echo $result->getErrorLevel();
         echo "<br />";
         echo $result->getErrorMessage();
         echo $result->getErrorCode();
         echo $result->getErrorDataMessage();
         echo $result->getId();
         echo "<br />";
         foreach ($data as $val)
         {
          echo "$val<br />";
         }
}
catch (SkrillPsp_Exception $e) {
    echo $e->getMessage();
}
```

Below are further details of the parameters used with this payment method.

Table 7-18: Parameters used with Yandex

Field Name	Description	Required	Length / format
transaction_id	A free text field included by you to track the transaction. This should be unique.	Yes	0-256 (alphanumeric)
amount	Transaction amount in the specified currency. There is no decimal separator. This number must be at least two digits. The API is built on a cent-based ISO8583 structure that does not cater for decimals (e.g., "**.**" or "**,**"). For example: 10 = ten Euro cents, 100 = 1 Euro, 1000 = 10 Euros.	Yes	2-12 (string or integer)
currency	Currency that the transaction is processed in. Currency code is according to the ISO 4217 specification.	Yes	3 (alpha)
response_url	The URL to which the transaction response data is sent. **Note**: this must be in HTTPS format. Your server should respond to a POST to this URL by returning ***HTTP 200*** with '***OK***' in the message body.	Yes	Any length (alphanumeric)
success_url	The URL to which the customer will be redirected to for a successful payment.	Yes	Any length (alphanumeric)
error_url	The URL to which the payer will be redirected to in case of unsuccessful payment.	Yes	Any length (alphanumeric)

7.5.1.1. Response example

Below is a description of the parameters returned in a response to a Yandex request.

```
{
    "id": 1,
    "jsonrpc": "2.0",
    "result": {
        "code": "0",
        "identification": {
            "shortid": "0039.0344.9374",
            "transactionid": "MerchantAssignedID",
            "uniqueid": "d0a56d001568dde31f7a28275604ee99"
        },
        "level": "16",
        "message": "new",
        "method": "webmoney",
        "processing": {
            "redirecturl": "http://test.transfers.com?tokenId=625ad496-6a21-4918-
91a4-e2fbfb270d28",
            "timestamp": "2013-04-11T13:13:02+01:00"
        },
        "type": "pa"
    }
}
```

You can use the ***redirecturl*** provided in the response to redirect the customer to the Yandex payment page, where they can complete their payment.

8. TESTING YOUR CONNECTION

8.1. Using test data

See below for test card details that you can use to process test transactions.

8.2.1. Non-3D Secure transactions

Table 8-1: Visa test card details

Test card values	Expected Results	Merchant Action
PAN = 4000000000000051 Expiry = 01/2016 CVV = 123	Successful token generation	Store and use the token

Table 8-2: MasterCard test card details

Test card values	Expected Results	Merchant Action
PAN = 5200000000000056 Expiry = 01/2016 CVV = 123	Successful token generation	Store and use the token

8.3.1. 3D Secure transactions

Table 8-3: Visa test card details

Test card values	Expected Results	Merchant Action	Liability shift
PAN = 4000000000000002 Expiry = 01/2016 CVV = 123	3D Secure window displayed. Successful token generation upon completion of 3D Secure.	Merchant should append the CAVV and token values to the authorisation request.	Yes

Table 8-4: MasterCard test card details

Test card values	Expected Results	Merchant Action	Liability shift
PAN = 5200000000000007 Expiry = 01/2016 CVV = 123	3D Secure window displayed. Successful token generation upon completion of 3D Secure.	Merchant should append the CAVV and token values to the authorisation request.	Yes

8.1. Testing error handling

Refer to the table below for a list of level 4 error messages for simulating 3 card payments. You can trigger these errors by including the code in the payment **amount** for the transaction on the test environment. (Note that any amount value from 0 to 99 will trigger the corresponding error message). For example:

```
"payment": {
            "amount": "01",
            "currency": "eur",
            "descriptor": "order number"
        },
```

Table 8-5: Level 4 error messages

Code	Message	Advice
01	Refer to card issuer	You should contact the issuer before the transaction can be approved.
02	Refer to card issuer special conditions	Special conditions may apply to this card. You should contact the issuer before the transaction can be approved.
03	Invalid merchant	You have not provided a valid merchant ID. Please contact your account manager or merchant services.
04	Pick up card	The card number has been listed on the Warning Bulletin List for reasons of counterfeit fraud or other.
05	Generic authorization decline	The transaction was declined by the card issuer without a specific reason. The customer should contact their bank for clarification.
06	Error	The banking network returned a General error with no further explanation.
07	Pick up card special conditions	The card number has been listed on the Warning Bulletin List. The merchant may receive reward money by capturing the card.
08	Honour with identification	The purchase can proceed, but we advise you to request some form of identification from the customer.
10	Approval for partial amount	The transaction has been approved for a partial amount.
11	Approved VIP	The cardholder is enrolled in the VIP programme.
12	Invalid transaction	The bank received an invalid transaction code. The transaction request presented is not supported or is not valid for the card number.
13	Invalid amount	The purchase amount was less than the minimum or greater than the maximum amount allowed for this card.
14	Invalid card number	The card issuer indicates that this card is not valid.
15	No such issuer	The issuing bank's identification number or BIN is invalid. Please ask the customer to check the card number or use an alternative payment method.

Table 8-5: Level 4 error messages

Code	Message	Advice
17	Customer cancellation	The customer cancelled this transaction.
19	Re-enter transaction	Enter the transaction details again and resend the transaction.
21	Cancel unsuccessful	The transaction cancel request was declined by the bank. This could be because the time limit for a cancel has expired.
25	Unable to locate record in file	The record could not be located.
27	File update field edit error	There was an error when updating the file.
28	File temporarily not available for update	This file is currently not available. Try again later.
30	Message format error	Error in the communication between the bank and Skrill. Please contact your account manager or merchant services.
32	Partial cancel	Only a partial cancel is possible. Please try to reverse a small amount.
33	Expired card pickup	The card has expired. The customer should pay using an alternative method.
38	Pin tries exceeded	The number of PIN tries has been exceeded. The customer should pay using an alternative method.
40	Function not supported	This option is not supported. Contact merchant services for details.
41	Lost card	This card has been reported as lost.
43	Stolen card	This card has been reported as stolen.
51	Insufficient funds	The transaction amount is over the customer's credit limit or the customer has insufficient funds in their account for this transaction.
52	No checking account	Not used
53	No saving account	Not used
54	Expired card	The card has expired or the expiration date is invalid.
55	Invalid pin	The PIN provided is invalid.
56	Cannot process	The banking network could not process this payment. No reason has been provided.
57	Transaction not permitted by cardholder	The cardholder is not permitted to make this transaction. The customer should contact their bank for details.
58	Transaction not permitted by acquirer	The banking network does not allow this type of transaction. Please contact your account manager or merchant services.
61	Exceeds approval amount	The cardholder has requested a withdrawal amount in excess of their daily limit.

Table 8-5: Level 4 error messages

Code	Message	Advice
62	Restricted card	The card has been restricted by the card issuer. The customer should contact their bank for details.
63	Security violation	The card has been restricted.
65	Exceeds withdrawal frequency limit	The allowed number of daily transactions has been exceeded.
66	Card acceptor call acquirer decline	Contact merchant services for details.
67	Hard capture	Not used
68	Response received too late	The transaction has timed out because the response was received too late.
69	Bad close	Not used
70	Card already active	Not used
71	Card not a gift card	Not used
72	Card already closed	Not used
73	Over maximum balance	Not used
74	Invalid activate gift card	Not used
75	Exceeds pin tries	The allowed number of pin entries has been exceed
76	Late cancel	The credit account does not exist or is not associated with the card number presented
77	Cancel does not match original transaction	The cancel request does not match the details of the original transaction.
78	Invalid account specified (general)	The associated card number account is invalid or does not exist.
79	Already cancelled.	Already cancelled.
80	Processor link out of service	Not used.
81	Pin key sync error	Not used.
82	Invalid CVV	Not used.
83	Unable to verify pin	Not used.
85	No reason to decline	Not used.
87	Network Unavailable	Banking network is unavailable
88	Card record not available	Not used
91	Issuer switch inoperative	The bank is not available to authorise this transaction.
92	Cannot reach network	The banking network is currently not available. Please try again later. If the problem persists

Table 8-5: Level 4 error messages

Code	Message	Advice
93	Illegal transaction	This transaction is not allowed.
94	Duplicate transaction	A duplicate transaction has been made. Some banks decline transactions if the same transaction details are sent more than once in a very short time period.
95	Reconciliation error	Reconciliation error.
96	System error	The banking network is currently not available. Please try again later. If the problem persists contact merchant services.
97	No response from host	Not Used
98	Duplicate transaction	A duplicate transaction has been made. Some banks decline transactions if the same transaction details are sent more than once in a very short time period.
99	Credit transaction can be performed as debit	Not Used

9. APPENDICES

9.1. Response codes

9.2.1. Level 1: Format error (front-end validation errors)

Table 9-1: Format validation errors returned by the Skrill Payment Platform

Code	Message	Advice
1	Card number is missing	Please provide a card number.
2	Cardholder name is missing	Please enter the name displayed on the credit card.
3	Expiry date is missing	Please provide an expiry date.
4	CVV is missing	Please provide the Card Verification Value (CVV). This is last 3 digits on the back of Visa and MasterCard cards, or last 4 digits for an Amex card.
5	Expiry date is missing	Please provide a valid expiry date.
6	Amount is missing	Please provide an amount.
7	Currency is missing	Please provide a currency.
8	Descriptor is missing	Please provide a short description of the transaction, to appear on the cardholder's statement.
9	Response URL is missing	Please provide a response URL.
10	Success URL is missing	Please provide a success URL.
11	Error URL is missing	Please provide an error URL.
12	Email is missing	Please provide an email address.
13	IP is missing	Please provide an IP address.
14	Reference ID is missing	Please provide a reference ID.
15	Token ID is missing	Please provide a token.
16	PARes is missing	Please provide a Payer Authentication Response (PARes) value.
17	MD is missing	Please provide a Merchant Data (MD) value.
18	CAVV is missing	Please provide a Cardholder Authentication Verification Value (CAVV).
19	XID is missing	Please provide a transaction identifier (XID).
20-21	Not in use	
22	Routing number/Sort Code is missing	Please provide the customer's bank routing transit number or sort code.

Table 9-1: Format validation errors returned by the Skrill Payment Platform

Code	Message	Advice
23	Country is missing	Please provide an ISO 2-digit country code.
24	Account number is missing	Please provide the customer's bank account number.
25	PIN is missing	Please provide a PIN.
26	QIWI user ID is missing	Please provide the QIWI user ID. This is the customer's telephone number.
27	Detailed description is missing	Please provide the detailed description.
28	Detailed text is missing	Please provide the detailed text.
29	Boleto Fiscal ID is missing	Please provide a Boleto Customer Fiscal Number.
30	Receiving currency is missing	Please provide a receiving currency.
31	Boleto country is missing	Please provide a country.
32	WebMoney country is missing	Please provide a country.
33	Frontend parameters are missing	Please provide frontend parameters such as Response URL, success URL and error URL.
34	Session ID is missing	Please provide a session ID.
35	iDEAL issuer ID is missing	Please provide the iDEAL issuer ID returned by the Directory request.
36	MultiBanco country is missing	Please provide a country.
37	Neosurf country is missing	Please provide a country.
38	Lobanet country is missing	Please provide a country.
39	Usage is missing	Please provide a usage.
40	Salutation is missing	Please provide a salutation.
41	Title is missing	Please provide a title.
42	First name is missing	Please provide a first name.
43	Surname is missing	Please provide a surname.
45	Street is missing	Please provide a street.
46	City is missing	Please provide a city.
47	State is missing	Please provide a state.

Table 9-1: Format validation errors returned by the Skrill Payment Platform

Code	Message	Advice
48	Phone number is missing	Please provide a phone number.
49	Mobile number is missing	Please provide a mobile number.
50	Not in use	
51	Card number is invalid	Please provide a valid credit or debit card number.
52	Expiry date is invalid	Please provide a valid expiry date. This is a number in the format DD/YYYY.
53	CVV is invalid	Please provide a valid Card Verification Value (CVV). This is last 3 digits on the back of Visa and MasterCard cards, or last 4 digits for an Amex card.
54	Amount is invalid	Please provide a valid amount. This must be a number between 2-12 characters, with a minimum value of 01. Amounts are cent-based and must not include decimals or commas.
55	Currency is invalid	Please provide a valid 3-digit ISO currency code.
56	IP is invalid	Please provide a valid IP address. This must be a numeric value of 15 characters, in the format: 255.255.255.255
57	Salutation is invalid	Please provide a valid Salutation. This must be an alphabetic value between 2-6 characters.
58	Title is invalid	Please provide a valid title. This must be an alphanumeric value between 1-20 characters.
59	Invalid Skrill Risk score	Invalid Skrill Risk score
60	Skrill bank routing number is invalid	Skrill bank routing number is invalid
61	Token Vault error	Problem processing the payment token. Please try again or contact Merchant Services
62	Invalid language	Invalid language
63	Routing number is invalid	Routing number is invalid
64	Invalid Paysafecard Locale parameter	Please provide a valid Paysafecard Locale value, such as ‘en_en’.
65	Invalid Paysafecard Minage parameter	Please provide a valid Paysafecard minimum age value, such as ‘18’.
66	Invalid Paysafecard KYC parameter	Please provide a valid Paysafecard KYC level, such as ‘SIMPLE’.
67	Invalid QIWI Lifetime parameter	Please provide a valid QIWI lifetime value in minutes, for example, ‘15’.
68	Invalid Yandex Paysource parameter	Please provide a valid Paysource parameter. This must always be ‘Yandex’.

Table 9-1: Format validation errors returned by the Skrill Payment Platform

Code	Message	Advice
69	Invalid Paysafecard country	Please provide a valid 2-digit ISO country code.
70	Invalid URL targets	Please provide a valid URL target.
71	Invalid new window redirect	Please provide a valid new window redirect URL.
72	Not in use	
73	Invalid Hide Login	Please provide a valid Hide Login parameter.
74	Invalid success URL	Please provide a valid success URL.
75	Invalid error URL	Please provide a valid error URL.
76	Invalid response URL	Please provide a valid response URL.
77	Invalid Fiscal ID	Please provide a valid Boleto Customer Fiscal Number.
78	Invalid Receiving currency	Please provide a valid currency.
79	Not in use	
80	Invalid Boleto country	Please provide a valid 2 digit ISO country code.
81	Invalid Webmoney country	Please provide a valid 2 digit ISO country code.
82	Paysafecard amount limit	Please provide a valid amount. This must be a cent-based number between 2-12 characters, without decimals.
83	Invalid Neosurf country	Please provide a valid 2 digit ISO country code.
84	Invalid Multibanco country	Please provide a valid 2 digit ISO country code.
85	Invalid Lobanet country	Please provide a valid 2 digit ISO country code.
86	Transaction ID is invalid	Please provide a valid transaction ID. This must be an alphanumeric value between 0-256 characters.
87	Reference ID is invalid	Please provide a valid reference ID. This must be an alphanumeric value with 32 characters.
88	Customer ID is invalid	Please provide a valid Customer ID. This must be an alphanumeric value with 32 characters.
89	Descriptor is invalid	Please provide a valid Descriptor. This must be an alphanumeric value between 1-255 characters. If displayed on the customer's credit card statement, this may be truncated to the first 18 characters by certain banks.
90	Cardholder name is invalid	Please provide a valid cardholder name. This must be an alphabetic value between 4-128 characters.
91	Expiry date is invalid	Please provide a valid expiry date. This is a number in the format DD/YYYY.

Table 9-1: Format validation errors returned by the Skrill Payment Platform

Code	Message	Advice
92	Descriptor is invalid	Please provide a valid descriptor. This must be an alphanumeric value between 1-255 characters. If displayed on the customer's credit card statement, this may be truncated to the first 18 characters by certain banks.
93	First name is invalid	Please provide a valid first name. This must be an alphabetic value between 2-40 characters.
94	Surname is invalid	Please provide a valid surname. This must be an alphabetic value between 2-40 characters.
95	Company is invalid	Please provide a valid company name. This must be an alphanumeric value between 2-40 characters.
96	Street is invalid	Please provide a valid street. This must be an alphanumeric value between 5-50 characters.
97	State is invalid	Please provide a valid state, region or county. This must be an alphabetic value between 2-8 characters.
98	Title is invalid	Please provide a valid title. This must be an alphanumeric value between 1-20 characters.
99	Phone number is invalid	Please provide a valid phone number. This must be an alphanumeric value between 8-64 characters.
100	Credit card length issue	Please provide a valid credit or debit card number. This should be a number between 15-17 characters long.
101	Cardholder name length issue	Please provide a valid cardholder name.
102	Expiry date length issue	Please provide a valid expiry date. This is a number in the format DD/YYYY.
103	Expiry month length issue	Please provide a valid expiry month. This is a number with 2 characters.
104	Expiry year length issue	Please provide a valid expiry year. This is a number with 4 characters and must be equal to or less than 2016.
105	CVV length issue	Please provide a valid Card Verification Value (CVV). This is last 3 digits on the back of Visa and MasterCard cards, or last 4 digits for an Amex card.
106	Transaction ID length issue	Please provide a valid transaction ID. This must be an alphanumeric value with 32 characters.
107	Customer ID length issue	Please provide a valid customer ID. This must be an alphanumeric value with 32 characters.
108	Amount length issue	Please provide a valid amount. This must be a number between 2-12 characters, with a minimum value of 01.
109	Currency length issue	Please provide a valid 3-digit ISO currency code.
110	Descriptor length issue	Please provide a valid descriptor. This must be an alphanumeric value between 1-255 characters.

Table 9-1: Format validation errors returned by the Skrill Payment Platform

Code	Message	Advice
111	Response URL length issue	Please provide a valid response URL. This must be an alphanumeric value between 1-255 characters.
112	Success URL length issue	Please provide a valid success URL. This must be an alphanumeric value between 1-255 characters.
113	Error URL length issue	Please provide a valid Error URL. This must be an alphanumeric value between 1-255 characters.
114	Email length issue	Please provide a valid email address. This must be an alphanumeric value between 6-128 characters.
115	IP length issue	Please provide a valid IP address.
116	First name length issue	Please provide a valid first name. This must be an alphabetic value between 2-40 characters.
117	Surname length issue	Please provide a valid surname. This must be an alphabetic value between 2-40 characters.
118	Company name length issue	Please provide a valid company name. This must be an alphanumeric value between 2-40 characters.
119	Street length issue	Please provide a valid street. This must be an alphanumeric value between 2-50 characters.
120	Zip/postal code length issue	Please provide a valid Zip/Postcode. This must be an alphanumeric value between 1-10 characters.
121	City length issue	Please provide a valid city. This must be an alphabetic value between 2-30 characters.
122	State length issue	Please provide a valid state. This must be an alphabetic value between 2-8 characters.
123	Country length issue	Please provide a valid ISO 3-digit country code.
124	Phone length issue	Please provide a valid phone number. This must be an alphanumeric value between 8-64 characters.
125	Mobile number length issue	Please provide a valid mobile number. This must be an alphanumeric value between 10-64 characters.
126	Skrill country length issue	Please provide a valid ISO 3-digit country code.
127	MCC length issue	Please provide a valid 4-digit Merchant Category Code (MCC) number.
128	Acquirer length issue	Please provide a valid acquirer.
129	Token length issue	Please provide a valid payment token. This must be an alphanumeric value of 32 characters.
130	MID length issue	Please provide a valid Merchant ID.
131	PARes length issue	Please provide a valid Payer Authentication Response (PARes) value.

Table 9-1: Format validation errors returned by the Skrill Payment Platform

Code	Message	Advice
132	ECI length issue	Please provide a valid Electronic Commerce Indicator (ECI).
133	MD length issue	Please provide a valid Merchant Data (MD) value.
134	Reference ID length issue	Please provide a valid reference ID. This must be an alphanumeric value of 32 characters.
135	CAVV length issue	Please provide a valid Cardholder Authentication Verification Value (CAVV).
136	XID length issue	Please provide a valid transaction identifier (XID).
137	Cardholder length issue	Please provide a valid cardholder name. This is an alphanumeric value between 4-128 characters.
138	Routing length issue	Please provide a valid bank routing number/SORT code.
139	Recipient length issue	Please provide a valid recipient length.
140	Paysafecard Shop ID length issue	Please provide a valid shop ID. This must be an alphanumeric value between 1-60 characters.
141	Paysafecard Shop label length issue	Please provide a valid shop label length. This must be an alphanumeric value between 1-60 characters.
142	Paysafecard Minage length issue	Please provide a valid Paysafecard minimum age value, such as '18'.
143	QIWI Lifetime length issue	Please provide a valid QIWI lifetime value in minutes, for example, '15'.
144-164	Not in use	
165	Card number is too short	Please provide a valid card number. This must be an alphanumeric value between 3-64 characters.
166	Card number is too long	Please provide a valid card number. This must be an alphanumeric value between 3-64 characters.
167	Amount is too short	Please provide a valid amount. This must be a number between 2-12 characters, with a minimum value of 01.
170	Amount is too long	Please provide a valid month. This must be a number between 2-12 characters, with a minimum value of 01.
171	Expiry year too short	Please provide a valid expiry year. This is a number with 4 characters and must be equal to or less than 2016.
172	Expiry year too long	Please provide a valid expiry year. This is a number with 4 characters and must be equal to or less than 2016.
173	CVV is too short	Please provide a valid CVV. This is last 3 digits on the back of Visa and MasterCard cards, or last 4 digits for an Amex card.
174	CVV is too long	Please provide a valid CVV. This is last 3 digits on the back of Visa and MasterCard cards, or last 4 digits for an Amex card.

Table 9-1: Format validation errors returned by the Skrill Payment Platform

Code	Message	Advice
175	Transaction ID is too short	Please provide a valid transaction ID. This must be an alphanumeric value between 0-256 characters.
176	Transaction ID is too long	Please provide a valid transaction ID. This must be an alphanumeric value between 0-256 characters.
177	Reference ID is too short	Please provide a valid reference ID. This must be an alphanumeric value with 32 characters.
178	Reference ID is too long	Please provide a valid reference ID. This must be an alphanumeric value with 32 characters.
179	Customer ID is too short	Please provide a valid Customer ID. This must be an alphanumeric value with 32 characters.
180	Customer ID is too long	Please provide a valid Customer ID. This must be an alphanumeric value with 32 characters.
181	Currency is too short	Please provide a valid 3-digit ISO currency code.
182	Currency is too long	Please provide a valid 3-digit ISO currency code.
183	Descriptor is too short	Please provide a valid Descriptor. This must be an alphanumeric value between 1-255 characters.
184	Descriptor is too long	Please provide a valid Descriptor. This must be an alphanumeric value between 1-255 characters.
185	Cardholder name too short	Please provide a valid cardholder name. This must be an alphabetic value between 4-128 characters.
186	Cardholder name too long	Please provide a valid cardholder name. This must be an alphabetic value between 4-128 characters.
187	Expiry date is too short	Please provide a valid card expiry date. This is a number in the format DD/YYYY.
188	Expiry date is too long	Please provide a valid card expiry date. This is a number in the format DD/YYYY.
189	Descriptor is too short	Please provide a valid Descriptor. This must be an alphanumeric value between 1-255 characters.
190	Descriptor is too long	Please provide a valid Descriptor. This must be an alphanumeric value between 1-255 characters.
191	Salutation is too short	Please provide a valid Salutation. This must be an alphabetic value between 2-6 characters.
192	Salutation is too long	Please provide a valid Salutation. This must be an alphabetic value between 2-6 characters.
193	Title is too short	Please provide a valid Title. This must be an alphanumeric value between 1-20 characters.
194	Title is too long	Please provide a valid Title. This must be an alphanumeric value between 1-20 characters.

Table 9-1: Format validation errors returned by the Skrill Payment Platform

Code	Message	Advice
195	First name is too short	Please provide a valid first name. This must be an alphabetic value between 2-40 characters.
196	First name is too long	Please provide a valid first name. This must be an alphabetic value between 2-40 characters.
197	Surname is too short	Please provide a valid surname. This must be an alphabetic value between 2-40 characters.
198	Surname is too long	Please provide a valid surname. This must be an alphabetic value between 2-40 characters.
199	Company name is too short	Please provide a valid company name. This must be an alphanumeric value between 2-40 characters.
200	Company name is too long	Please provide a valid company name. This must be an alphanumeric value between 2-40 characters.
201	Street is too short	Please provide a valid street. This must be an alphanumeric value between 2-50 characters.
202	Street is too long	Please provide a valid street. This must be an alphanumeric value between 2-50 characters.
203	City is too short	Please provide a valid city. This must be an alphabetic value between 2-30 characters.
204	City is too long	Please provide a valid city. This must be an alphabetic value between 2-30 characters.
205	State is too short	Please provide a valid state. This must be an alphabetic value between 2-8 characters.
206	State is too long	Please provide a valid state. This must be an alphabetic value between 2-8 characters.
209	Phone number is too short	Please provide a valid phone number. This must be an alphanumeric value between 8-64 characters.
210	Phone number is too long	Please provide a valid phone number. This must be an alphanumeric value between 8-64 characters.
211	Mobile number is too short	Please provide a valid mobile number. This must be an alphanumeric value between 10-64 characters.
212	Mobile number is too long	Please provide a valid mobile number. This must be an alphanumeric value between 10-64 characters.
213	Email is too short	Please provide a valid email address. This must be an alphanumeric value between 6-128 characters.
214	Email is too long	Please provide a valid email address. This must be an alphanumeric value between 6-128 characters.
215	IP address is too short	Please provide a valid IP address.
216	IP address is too long	Please provide a valid IP address.
217	XID is too short	Please provide a valid XID.

Table 9-1: Format validation errors returned by the Skrill Payment Platform

Code	Message	Advice
218	XID is too long	Please provide a valid XID.
219	CAVV is too short	Please provide a valid CAVV.
220	CAVV is too long	Please provide a valid CAVV.
221	ECI is too short	Please provide a valid ECI.
222	ECI is too long	Please provide a valid ECI.
223	Verification type is too short	Please provide a valid Verification type.
224	Verification type is too long	Please provide a valid Verification type.
225	Response URL is too short	Please provide a valid Response URL. This must be an alphanumeric value between 1-255 characters.
226	Response URL is too long	Please provide a valid Response URL. This must be an alphanumeric value between 1-255 characters.
227	Session ID is too short	Please provide a valid Session ID.
228	Session ID is too long	Please provide a valid Session ID.
229	Success URL is too short	Please provide a valid Success URL. This must be an alphanumeric value between 1-255 characters.
230	Success URL is too long	Please provide a valid Success URL. This must be an alphanumeric value between 1-255 characters.
231	Response URL is too short	Please provide a valid Response URL. This must be an alphanumeric value between 1-255 characters.
232	Response URL is too long	Please provide a valid Response URL. This must be an alphanumeric value between 1-255 characters.
233	Country code is too short	Please provide a valid 2-letter Country code.
234	Country code is too long	Please provide a valid 2-letter Country code.
235	Country code is too short	Please provide a valid 3-letter Country code.
236	Country code is too long	Please provide a valid 3-letter Country code.
237	Mobile number is invalid	Please provide a valid mobile number. This must be an alphanumeric value between 10-64 characters.
238	Email address is invalid	Please provide a valid Email address. This must be an alphanumeric value between 6-128 characters.
239	Transaction ID is missing	Please provide a transaction ID.
240	City is invalid	Please provide a valid city. This must be an alphabetic value between 2-30 characters.
241	Invalid store in vault	

Table 9-1: Format validation errors returned by the Skrill Payment Platform

Code	Message	Advice
242-290	Not in use	
291	Reference ID not found	The transaction that is being referenced could not be found. Please enter a valid reference ID.
292	Authentication MD not found	The Merchant Data (MD) value provided for 3D secure could not be found. Please provide a valid MD value.
293	VANTIV MCC not found	Please provide a valid 4-digit Merchant Category Code (MCC) number.
294	Elavon MID not found	The data you have provided could not be resolved to a valid Elavon merchant ID.
295	No liability shift	This transaction was processed, but there is no liability shift and a chargeback is possible.
296	Cardinal failed authentication	
297-299	Not in use	
300	Invalid merchant	
301	Invalid merchant. Too many MIDs	
302	Invalid processing channel	Please provide a valid channel.
303	Invalid processing merchant ID	Please provide a valid merchant ID.
305	Invalid processing brand	Please provide a valid brand.
306	Invalid processing currency	Please provide a valid currency.
307	Invalid processing MCC	Please provide a valid 4-digit Merchant Category Code (MCC) number.
308	Invalid processing country	Please provide a valid ISO 3-digit country code.
309	Payment method not supported	Please provide a valid payment method.
310	Type not supported	
311	JSON payment method not supported	Please provide a valid JSON payment method.
312	Currency not found	Please provide a currency.

9.3.1. Level 2: System error (configuration/connectivity errors)

Table 9-2: System error codes returned by the Skrill Payment Platform

Code	Description
0	Undefined system error
1	Token socket connection error
2	Request method not found
3	Wrong request URL
4	Method not supported
5	Application socket connection error
6	Application error
7	Wrong channel
8	Currency not supported
9	Method not supported
10	Type not supported
11	Brand not supported
12	Pro socket connection error
13	Undefined token connection error
14	Mnesia not running
15	Token not in table etc.
16	MPI connection timeout
20	Refunds not allowed
21	No debit was captured
22	User Not enrolled
23	Card not participating/ authentication unavailable
24	Technical error in 3D system
25	Socket connection error

9.4.1. Level 4: Acquirer reason codes

Table 9-3: Success and failure reason codes returned by the acquirer

Code	Description
00	Approved or completed successfully
03	Invalid merchant number
05	Authorization declined
12	Invalid transaction
13	Invalid amount
14	invalid card
21	No action taken
30	Format Error
33	Card expired
34	Suspicion of Manipulation
40	Requested function not supported
43	Stolen Card, pick up
55	Incorrect personal identification number
56	Card not in authorizer's database
57	Referencing transaction (e.g. , Booking pre-authorization ...) was not carried out with the card which was used for the original transaction.
58	Terminal ID unknown
62	Restricted Card
64	The transaction amount of the referencing transaction is higher than the transaction amount of the original transaction
78	Stop payment order (for forwarding the Visa response code "R0" of the Visa BASE I interface): the transaction was declined or returned because the cardholder requested that payment of a specific recurring or instalment payment transaction be stopped.
79	Revocation of authorization order (for forwarding the Visa response code "R1" of the Visa BASE I interface): the transaction was declined or returned because the cardholder requested that payment of all recurring or instalment payment transactions for a specific merchant account be stopped.
80	Amount no longer available
81	Message-flow error
91	Card issuer temporarily not reachable
92	The card type is not processed by the authorization centre
96	Processing temporarily not possible
97	Security breach - MAC check indicates error condition

10.GLOSSARY

This section provides a description of key terms used in this guide.

Term	Explanation
Acquirer	An acquiring bank (or acquirer) is the bank or financial institution that processes credit and or debit card payments for a merchant. Example: Barclays Merchant Service and European Merchant Services.
Address Verification Service (AVS)	This service validates the billing address provided by the customer on the payment against the billing address registered with the customer's bank or credit card. Typically, only the numbers in the first line of the address and postcode are verified and the result is returned to the merchant. (only available in the US, Canada and UK)
Alternative payment	Local payment method specific to a country or region, or method that is not a traditional international credit or debit card type of payment.
API	Generic term for Application Programme Interface. API's provide a means for developers to communicate with the Skrill Payment Platform.
Authentication	Security checks performed to verify the identity of a customer, merchant, website or entity. For example: Cardholder authentication checks such as 3D Secure and website authentication via SSL certificates.
Authorisation	Also known as Pre-authorisation. Process where the merchant can send a request to the banking network via Skrill to verify that the card is valid and that the funds requested can be taken from the card.
Back-end system	As opposed to a front-end system, a back-end system used internally by Skrill or within the merchant's business. Skrill merchants can also use payment information returned from the Skrill Payment Platform on their own back-end systems, such their customer order management system.
Batch	A group of approved credit card transactions, accumulated during one business day (weekends and official/bank holidays excluded).
Batch Processing	The authorisation of transactions offline when immediate approval is not required. Transactions are collected in a batch and sent as one transmission for Authorisation and/or Settlement.
BIN	Bank Identification Number. A unique series of numbers which identifies an institution in transaction processing. The BIN comprises the first six digits of a standard debit or credit card number.
Browser	Application that enables a customer or merchant to access web pages. Examples include: Internet Explorer, Google Chrome and Mozilla Firefox.
Cancel	Request to cancels a transaction. This is only possible before you have captured the payment or until the preauthorisation expires. It can be for a partial amount.

Term	Explanation
Capture	Process of requesting payment from the customer's account. Only preauthorised transactions can be captured. For some merchants, there may be a delay between the initial authorisation of the transaction and when capture is made. See Delayed capture.
Cardholder	The customer or online shopper whose card is used during an online purchase.
Cardholder authentication	In online transactions, where the customer is not physically present, certain payment methods, such as Visa, MasterCard and Amex, provide an alternative means for verifying the customer's identity. Typically, the customer is asked to provide a unique password or other identifying information during the online transaction. See also 3D Secure and Authentication.
Card Issuer	Bank or card scheme that issues the customer's payment card.
Cardholder Funds Transfer (CFT)	Also called Credit Funds Transfer, is a method of transferring funds which can be up to or more than the original transaction amount. Currently only a VISA card used for a successful gaming deposit is available for CFT payment. See also Credit.
Card scheme	Card association such as Visa, MasterCard, Discover, American Express, Diners Club and JCB.
Card Verification Value (CVV)	CVV is a three digit code printed on the back of Visa, MasterCard and Discover cards and a four digit code printed on the front for American Express. Also known as CVV2, CV2 and CSC (Card Security Code).
Cardholder Funds Transfer (CFT)	Also called Credit Funds Transfer, is a method of transferring funds which can be up to or more than the original transaction amount. Skrill merchants or customers can arrange a CFT by transferring funds from their digital wallet account to a credit card. There are some restrictions. Currently only a VISA card used for a successful gaming deposit is available for CFT payment. See also Credit.
Channel	Channels enable merchants to process payments in different ways. For example, a merchant may be set up with two channels, one which has 3D Secure enabled and the other which has 3D Secure disabled. Some channel options may require a separate merchant account.
Chargeback	The return of funds, previously authorised in a transaction, to a customer, which is initiated by the issuing bank. Also refers to the of a prior outbound transfer. A chargeback applies when customer contacts their Card Issuer to dispute a transaction. In this case, the card issuer or bank may request that the money be paid back directly by Skrill. The merchant may incur an administration cost for Skrill processing the dispute, in addition to any amount eventually credited back to the customer.

Term	Explanation
Credit	Credit can be used by merchants to transfer money to the customer. Also known as Payouts / Payout of Winnings (PoW) and Cardholder Funds Transfer (CFT). The difference to a Refund is that more than the original transaction amount can be transferred back to the customer. Mostly common in the gambling industry. A credit is only allowed on a credit card which has been subject to a charge before.
Credit card	A type of payment card that allows customers to pay for goods and services using funds that are loaned. The loan must be paid back within a specified period. Interest is typically charged on the balance after a grace period (typically 20-55 days). Examples: Visa, MasterCard, Diners and Amex. See also Debit card.
Customer ID	On the Skrill Payment Platform, a merchant may be configured with multiple customer IDs set up for different channels. (Not to be confused with the *customerid* field.)
Customer services team	Skrill team responsible for end-customer support queries. Also referred to as the Merchant Services team.
cURL	cURL is a command line tool for transferring data with URL syntax. Merchants can use cURL to send transactional request data to the Skrill Payment Platform. See also API.
Debit	Process in which the merchant can send a request to the banking network via Skrill for authority to take payment for an online transaction. Debit combines Authorisation and Capture in a single request.
Debit card	A type of payment card that provides customers with instant access to funds in their bank account. In contrast to a Credit card payment, payments using a debit card are immediately taken from the customer's account, instead of being paid back at a later date. So, the customer must have sufficient funds in their account or an agreed overdraft limit to cover the payment. Delayed capture is typically not available for debit cards.
Decline	A response from the Card Issuer denying the use of the card for the attempted transaction. If a request for approval is declined, the merchant must ask the cardholder for another form of payment.

Term	Explanation
Delayed capture	Option that enables the merchant to delay the capture of a payment that has been authorised. Often referred to as Authorisation and Capture. Typically, merchants request authorisation for a payment to ensure that the customer has sufficient funds in their account. The funds are then reserved and cannot be used by the customer for a certain period. Once the goods or services have been delivered, the full amount is taken from the customer's card. Car hire purchases are a good example of delayed capture. The merchant typically requests a Pre-authorisation on the card, but full authorisation and capture only occurs once the vehicle has been returned.
Descriptor	A field descriptor, used in different contexts on the Skrill Payment Platform to enable merchants to pass descriptive information to the system.
Digital Wallet	Provides the electronic equivalent of a physical wallet that can hold a number of cards and payment methods, which are linked to the wallet. Typically, customers log in to their wallet account during an online transaction and authorise the payment using one of the payment methods linked to their wallet account.
Financial Conduct Authority (FCA)	Organisation that took over the role of the Financial Services Authority (FSA) in regulating the activities of financial organisations operating in the UK.
Hosted payment page	Also known as Web Payment Front-end (WPF) or Hosted Gateway. This option provides a secure payment page hosted by Skrill, where customers can enter card and other payment details. It is suitable for merchants who want a simple integration and/or who are not PCI compliant.
HTML POST	Integration method where the merchant sends details to the Skrill Payment Platform using a standard HTML form that posts this information in the HTML header.
In-store	Refers to payment made by a customer who is physically present at the merchant's store. Often referred to as *brick-and-mortar* or customer present payments.
International Bank Account Number (IBAN)	Originally adopted by the European Committee for Banking Standards (ECBS), and later adopted as an international standard under ISO 13616:1997. The current standard is ISO 13616:2007, which indicates SWIFT as the formal registrar. Initially developed to facilitate payments within the European Union, it has now also been implemented by most European countries and many other countries, especially in the Middle East and in the Caribbean.

Term	Explanation
Integration	Process undertaken by merchants to ensure that their website or shopping cart can connect to and communicate with Skrill's payment processing systems.
Integration methods	The connection options provided to enable a merchant to integrate with the Skrill Payment Platform. For example, JSON and Web Payment Front-end (WPF).
ISO country codes	3-digit country code of the International Standards organisation (ISO) that identifies the country. For example, GBR for United Kingdom. ISO country codes also exist in a 2-digit format.
ISO currency codes	3-digit currency code of the International Standards Organisation (ISO) that identifies the currency. For example, GBP for British Pound.
JavaScript	JavaScript is a dynamic scripting language used primarily in client-side web browser applications.
KYC	Acronym for Know your Customer. Refers to due diligence procedure in the financial services industry to verify the identity of the customer, and is conducted by Skrill for merchants and Skrill digital wallet customers who reach over 2,500 Euro in processed transactions.
JQuery	jQuery is a multi-browser JavaScript library designed to simplify the client-side scripting of HTML.
JSON	JavaScript Object Notation is an open-source data communication standard support by the Skrill Payment Platform. Merchants can communicate with the platform using JSON.
Liability Shift	For transactions where the customer is authenticated via 3D Secure, the merchant may be protected from Chargeback involving identity fraud.
Mail Order Telephone Order (MOTO)	Solution for enabling the merchant's customer services team to take payments over the telephone or from mail order.
MasterCard SecureCode®	Cardholder authentication method used by MasterCard to verify the identity of a customer during an online transaction. See also 3D Secure.
MD5	A widely used hash algorithm, which can be used for securely encrypting information sent over the internet. MD5 produces a 128-bit (16-byte) hash value. The purpose of the field is to ensure the integrity of the data posted back to the merchants' server.
Merchant	Skrill customer (legal or natural person) using their Skrill solution to receive payments for products or services they provide.
Merchant Agreement	The Skrill Merchant Agreement is the agreement governing the legal relationship between Skrill and a merchant using the Skrill payment processing services for commercial purposes. It consists of the Skrill Merchant Terms and Conditions, plus cover pages, schedules and the Skrill Account Terms of Use.
Merchant ID (MID)	A merchant identifier, provided by the Acquirer, used to uniquely identify a merchant within the banking network when a transaction is processed.

Term	Explanation
Merchant Portal	Online portal used for account administration and viewing transactions by merchants.
Merchant Services team	Skrill team responsible for providing technical and service support to merchants.
My Account	Customer account administration portal that enables viewing of transactions and transferring funds.
On-boarding process	Process of signing up and verifying a merchant. This involves a number of teams in Skrill, including sales and risk and compliance. On the payment processing side, this process is coordinated by a dedicated on-boarding team.
Online Bank Transfer	A payment method enabling customers to transfer funds from their bank accounts to their Skrill account in real-time. See also Skrill Direct.
Payment option or method	The payment method used by the customer, such as debit card, credit card and bank transfer. Note that in the payments industry, the terms payment method, payment option and payment type are often used interchangeably. See also Alternative payments.
Payment brand	Term used on the Merchant Portal to describe the payment brand connected to a payment method. For example, Visa and MasterCard are payment brands available for credit and debit cards payment methods.
Payment token	Token returned from the Skrill Payment Platform. The token enables repeat transactions on a card, without requiring the customer to re-enter their card details again.
Payment type	The type of payment service that was carried out against the payment method used. For example, Pre-authorisation and Capture.
Payouts / Payout of Winnings (PoW)	Payouts refer to payments made by merchants to customers, typically used in the online gambling industry. Skrill offers a payout product that enables merchants to make automated payments from their digital wallet account to their customers' digital wallet account. This can be used for paying employees or payment of winnings to customers. See also Cardholder Funds Transfer (CFT) which provides an alternative means of making payouts.
Payment	Unique financial record on the system. A payment may consist of multiple Transactions.
Payment page	Page used to collected payment method details from the customer during an online transaction. This page can be provided by the merchant or hosted by Skrill. See also Hosted payment page.
Payment processing platform	System used for the processing of eCommerce transactions.
Payment Service Provider (PSP)	Service provider enabling merchants to accept eCommerce payments.

Term	Explanation
PCI Compliance	The Payment Card Industry Data Security Standard (PCI DSS) is a security standard for organisations that handle cardholder information. Merchants who collect and store cardholder information must be PCI compliant.
PCI compliant	Merchants who collect and store cardholder information must comply with minimum standards to ensure that sensitive customer financial information is stored securely. The standard covers both systems and business processes in place within an organisation.
PCI Level 1 certified	Merchants fall under four categories of PCI compliant, depending on the number of transactions they process each year, and whether those transactions are performed from an In-store location or over the Internet. Level 1 is the highest level of compliance, typically demonstrated by large financial organisations and payment service providers such as Skrill.
Pending transaction	A transaction in which the payment system is waiting for a confirmation, an input or customer action. For example, 3D Secure, redirect payment method or manual bank transfer.
POST method	A request method supported by the HTTP protocol, which enables data to be sent to a web server. It is often used when uploading a file or submitting a completed web form. As part of a POST request, data of any type can be sent to the server in a message body. A header field in the POST request usually indicates the message body's Internet media type.
Pre-authorisation	A request made by Skrill to the Card Issuer to confirm that the customer has sufficient funds in their account for a credit card transaction. The funds are typically reserved, so that they will be available when the Debit or Capture request is made.
Primary account number (PAN)	Payment card number of 14 or 16 digits embossed on a bank or credit card and encoded in the card's magnetic strip. The PAN identifies the issuer of the card and the account, and includes a check digit as an authentication device.
Refund	Option to pay money back to a customer, which can be done using the API or Merchant Portal. The refund has to be referenced to the original payment and can only be up to that amount. See also Credit. Skrill enables partial or full amount refunds.
Register	Type of API call, where a request is made for a payment token from the Skrill payment platform. Skrill registers the card details and returns a payment token, which can be used for subsequent transactions with this card.

Term	Explanation
Real-time	An event that occurs instantly or within a short period, such as seconds or minutes. For a real-time transaction, the customer, merchant or Skrill receive a response to the transaction request while the customer is still online.
Reason code	Every transaction has a reason code, which indicates the status of the transaction. Skrill receives a variety of reason codes from the Card Issuer, bank or scheme authorising the transaction and consolidates these before providing them to merchants.
Reserve	A portion of the merchant's transaction funds or balance temporarily held in their account and which cannot be withdrawn. The reserve functions to protect Skrill from costs and charges such as unpaid transactions, Chargebacks or inaccurately executed transactions.
Settlement	When funds authorised by the Card Issuer or customer's bank are transferred to the Skrill account. Depending on the type of payment method used, settlement may be between one to fourteen days or more after the transaction. Once funds are settled into the merchant's wallet, merchants can transfer the balance (minus any Reserve) from their Skrill account to their local bank account.
Transaction	Each financial interaction with the Skrill Payment Platform is referred to as a transaction. A transaction is linked to a Payment.
XML integration	Integration method enabling merchants to connect to the payment gateway using XML.
3D Secure	Term for cardholder authentication, offered by schemes such as Visa, MasterCard and Amex to verify the identity of a customer during an online transaction.
Skrill Digital Wallet	Skrill's Digital Wallet, enabling customers to link cards and pay directly from their wallet account using cards or bank transfer. Up to 4 payment cards and 10 bank accounts can be linked to a wallet account.
Skrill Direct	Skrill's Online Bank Transfer product, which can be used with local bank in the UK, France, Germany, Italy, Spain and Austria.
Skrill Quick Checkout	Skrill product, related to the Skrill Digital Wallet, which enables customers to bypass the Skrill registration details page and simply confirm and pay. Quick Checkout uses the eCommerce platform for processing payments.
Skrill Payment Platform	Skrill's system for the processing of eCommerce payments.
SSL	Secure socket layer. An industry standard encryption method that allows for secure transmission of data between customers and merchant sites.

Term	Explanation
Tokenization	Feature that enables repeat payments without requiring the customer to provide their card details again for subsequent payments. When the merchant submits the register request, the Skrill Payment Platform stores the customer's card details securely and generates a Payment token, which is then sent to the merchant. The merchant can use the token to take repeat payments in the future, without needing to submit the credit card or bank account details again.
Transaction	Each financial interaction with the Skrill Payment Platform is referred to as a transaction. Transactions are linked to payments.
Transaction ID	Unique ID assigned to a transaction by the Skrill Payment Platform.
Transaction status	Each transaction on the Skrill Payment Platform is given a status. This includes: *processed, pending, temporary, scheduled, cancelled, failed, chargeback* and *successful*.
Uptime	Measure of the time a computer or service has been working and available. Uptime is the opposite of downtime. It is often used as a measure of reliability or stability.
Verified by Visa® (VbV)	Cardholder authentication method used by Visa to verify the identity of a customer during an online transaction. See also 3D Secure.
Viewpoint	Internal Skrill system used for setting up merchants on the Skrill Payment Platform. Viewpoint also provides a conduit for transaction and settlement data.
Web Payment Front-end (WPF)	The Web Payment Front-end enables merchants to connect to Skrill using a Hosted payment page where customers can enter their payment details.

Index