SOFORT
A Klarna Group Company

IDEAL - API DOCUMENTATION

VERSION: 1.0

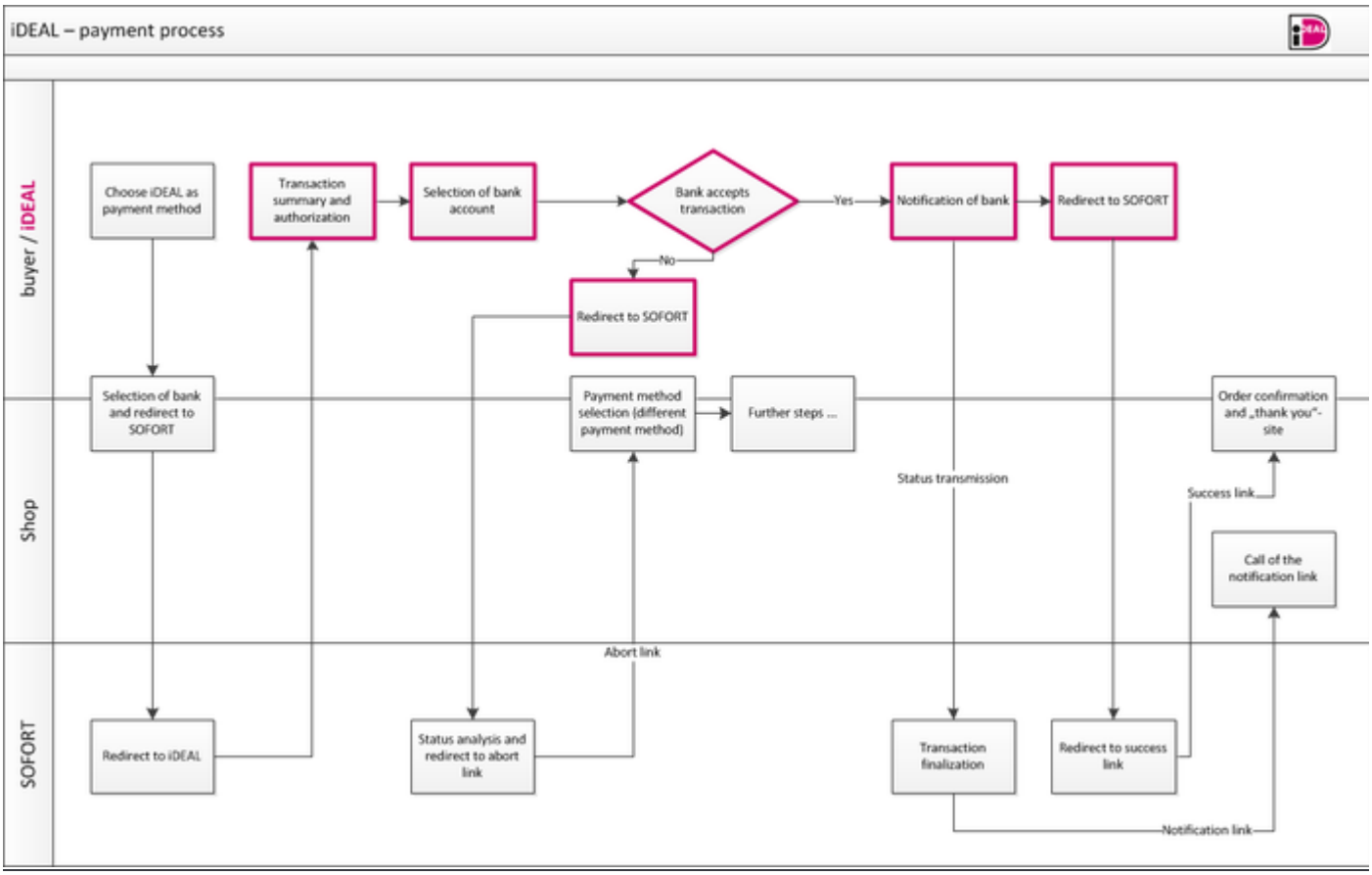
Introduction to iDEAL

The payment method iDEAL is of particular interest to merchants who want to sell their product in the Netherlands. Here, the payment is carried out based on the online banking of the customers in the Netherlands.

Your Dutch customers (end customers) do not have to register for iDEAL: They can make a fast and easy payment using their online banking details. Thanks to real time acknowledgement of the transfer order, you as a merchant can immediately ship the goods or provide services.

From your customer's perspective, an iDEAL transaction consists of the following steps:

- As soon as payment is requested by iDEAL, the customer selects the Dutch bank of his/her current account and is redirected via a SOFORT page to the iDEAL payment form.
- Below the overview of the transfer data (recipient, reason, amount), your customer is requested to authorise the payment.
- The next step is the selection of the account to be used for the payment.
- After a successfully completed transaction, your customer is redirected back to your shop.



IMPORTANT!

- You need a Deutsche Handelsbank account in order to use iDEAL.
- The application must be integrated in the system of the provider in such manner that the URL and the SSL certificate of SOFORT can be recognised and verified for the customer of the provider.
- The standard coding used for all parameters is UTF-8! Please bear this in mind especially for hash computation.
- The account data (account number, IBAN) are transferred as masked entries for the redirection via the success link and the HTTP notification. Sole exception: HTTPS protocol with POST method

Integration steps

The following steps are basically required to integrate iDEAL by SOFORT in your system. A detailed description of the individual steps is provided subsequently:

1. Create a test procject at SOFORT:
    1. Register as a merchant on our website https://sofort.com/register.
    2. Activate the product "iDEAL" in the merchant menu (at "My account > Product activation")
    3. Create a new iDEAL project (at "Projects > New Project > SOFORT Classic Project > Create iDEAL Project").
    4. Since "iDEAL" requires a Deutsche Handelsbank account, you have to select in the next step whether you already have an account or want to open a new account ("I have a Deutsche Handelsbank account" or "I don't have a Deutsche Handelsbank account yet").
    5. Configure your iDEAL Project.
    6. After you have created and configured a project, you will receive an API key as well as a notification password. (The API key can also be found in the left menu at "Services > API Key" and the passwords can be accessed in the iDEAL project at "Extended Settings > Passwords and Hash Algorithm").
    7. Contact SOFORT to configure your created account as Testaccount. (Technical support: integration@sofort.com)
2. The next step is the integration in the shop:
    1. Integrate the iDEAL payment method for selection in the checkout including the bank selection by means of a drop down field.
    2. As soon as a customer requests to pay by iDEAL, redirect the customer to our interface including the respective payment data.
    3. The customer processes the payment on our pages or the iDEAL pages and is subsequently redirected to the success or abort page of your shop which you have defined.
    4. In case of a successful iDEAL transaction (i.e. the transfer was placed by your customer) and in case of further status changes, you will automatically be notified of this transaction by SOFORT. Subsequently, you can immediately initiate further steps such as ship the goods or activate the online service. The notification can be issued by email or via XML HTTP notification to the notification page of the shop.
    5. The shop system automatically processes the nofitications and changes the order / payment statuses accordingly.

Integration of iDEAL

As a first step, you have to integrate iDEAL in the checkout of your shop to offer your customers iDEAL as a payment method to be selected. If the customer selects iDEAL as a payment method and confirms the order, communication between your shop and our interface (API) is initiated. Before the individual iDEAL process steps and the corresponding API calls are described in detail, several framework conditions are explained that are required for successful execution of iDEAL transfers.

Integration of iDEAL in the checkout

To integrate iDEAL in the checkout, it is necessary to add an additional option for iDEAL to the selection of the payment methods (for example as a new radio button, link, etc. - Details on this topic can be found in the documentation of your shop system). As customers are redirected to the portal of the corresponding Dutch bank when paying with iDEAL by SOFORT, they must already select their bank (ideally from a drop down field) in the selection of payment methods. The content of the drop down field can be retrieved from an XML API.

Communication with the interfaces for iDEAL

The first API call - searching for banks which are available in the checkout for bank selection - is made from your shop by a HTTP call. The API response is formatted in XML.

- (API-Step 0) Retrieval of iDEAL bank information including API response

All other API calls are made from your (shop) system by a HTTP call (either by GET or POST) with name-value-pair (NVP) parameters. The principle of the responses is the same.

Direct communication with our interface by NVP-HTTP messages always follows the same pattern:

- (API-Step 1) Redirection of customer to the iDEAL payment wizard (via SOFORT) and transaction parameter transfer
- (Customerstep) Listing of transfer on iDEAL and redirection to success / abort link
- (API-Step 2) Transaction status change notification

If you use PHP as the programming language for the integration, we provide a code library (SofortLib) to considerably facilitate the implementation in your system. Implementation examples and short instructions on how to use the SofortLib can be found in the Integration Center.

Detailed information on the individual API calls and responses can be found in the section "Overview of parameters".

Retrieval of iDEAL bank information including API response (API step 0)

Since iDEAL payments are processed on the portals of the corresponding Dutch sender bank, the customer must already select his/her bank in the checkout. The selection is subsequently transferred as BIC in the redirection of the customer to SOFORT. For this purpose, the customer is supposed to select his/her bank in a drop down menu on the payment selection page in the shop. For the subsequent call (API step 1), please transfer the correct BIC to us (in the field sender_bank_code).

API call and authentication

To prevent abuse of the interface, authentication must be performed for *every* XML interface call, transferring your customer number as user name and the API key as password. The API key is provided in the merchant menu at *"Services > API key"*. Please note the following issues which are also explained in the overview of parameters in order to use the interface:

- You have to call the correct URL using HTTPS as protocol.
- You have to transfer the correct authentication information. The basic HTTP authentication (RFC 2617) is used for authentication.
- You have to enter the correct content type headers.
- Your call must be sent by HTTP POST.

The interface is called via the following URL:

```
https://www.sofort.com/payment/ideal/banks
```

The response is formatted in XML and might look as follows:

```
<?xml version="1.0" encoding="UTF-8" ?>
<ideal>
    <banks>
        <bank>
            <code>ABNANL2A</code>
            <name>ABN Amro</name>
        </bank>
        <bank>
            <code>FRBKNL2L</code>
            <name>Friesland Bank</name>
        </bank>
        ...
    <banks>
</ideal>
```

Redirection of customer to the iDEAL payment wizard (via SOFORT) and transaction parameter transfer (API step 1)

Various parameters must be transferred to our API for the call. Some parameters are mandatory, some are optional. To make sure that this data cannot be changed on its way from the shop system to SOFORT, parts of the message will be hashed with the project password and transmitted as a hash value. Details on the use of the individual parameters and on the hash value calculation are listed in the overview of parameters.

The wizard is called via the following URL:

```
https://www.sofort.com/payment/ideal
```

Please implement the call only via this URL and transfer the required parameters (see section "Overview of parameters"), otherwise an error message will appear. You may transfer the parameters by GET or POST. For GET, add the variables to the link specified above; for POST, the parameters are transferred in a form in the body of the request.

Example of an interface call with a hash value and user_variable (POST method):

```
<form method="post" action="https://www.sofort.com/payment/ideal">
    <input type="hidden" name="user_id" value="12345" />
    <input type="hidden" name="project_id" value="654321" />
    <input type="hidden" name="amount" value="30.00" />
    <input type="hidden" name="reason_1" value="Bestellnummer 1" />
    <input type="hidden" name="sender_bank_code" value="ABNANL2A" />
    <input type="hidden" name="sender_country_id" value="NL" />
    <input type="hidden" name="user_variable_0" value="Ihr Wert" />
    <input type="hidden" name="hash" value="7aa872ed86b411654478d95c4adfefd09dfaf75a" />
    <input type="submit" value="Mit iDEAL bezahlen" />
</form>
```

Example of an interface call as a link (GET method):

```
https://www.sofort.com/payment/ideal?user_id=12345&project_id=654321&amount=30.00&
     reason_1=Bestellnummer%201&sender_bank_code=ABNANL2A&sender_country_id=NL&
     user_variable_0=Ihr%20Wert&hash=7aa872ed86b411654478d95c4adfefd09dfaf75a
```

Listing of transfer on iDEAL and redirection to success / abort link (customer step)

In the next step, your customer conducts the transfer in iDEAL.

After successful execution of transfer by iDEAL, your customer will be redirected to the success link stored with the project. Ideally, the order will be confirmed in a final summary including the successful execution of the iDEAL transfer. In the background, your shop system should meanwhile have received a confirmation of the transaction (see API steps 2), emptied the shopping cart, and saved the order with the corresponding status.

In case the iDEAL transfer has not been carried out successfully, your customer will be redirected to the abort link. This link should ideally redirect the customer to the selection of payment methods with a corresponding error message with the shopping cart still being filled.

Transaction status change notification (API step 2)

After successful execution of an iDEAL transfer, you will be notified of the executed transaction while the customer is redirected to the success page. This is done by calling the notification URL stored with the project (either as HTTP POST or HTTP GET). This notification includes the transaction ID of the transaction whose status changed. To make sure that this data cannot be changed on its way from the shop system to SOFORT, parts of the message will be hashed with the project password and transmitted as a hash value. This hash value must be verified in the shop system before the order status can be changed. Details on the individual parameters and on the hash value calculation are listed in the overview of parameters.

NOTE: The hash calculation for the notification considerably differs from the hash calculation for our payment wizard call (API step 1).

For the input check, a hash value is created from a few parameters with the project password, as little information on the payment process is available at this point. For the hash calculation for the notification however all information on the payment is available and is therefore included in the hash value calculation in connection with the notification password (see section overview of parameters). Furthermore, we automatically transfer all parameters required for hash calculation and the hash values are compared in your script.

An example of an interface notification (POST method) might look as follows:

```
transaction=12345-654321-51E811F3-BA51&user_id=12345&project_id=654321&sender_holder=Max%20Mustermann&
     sender_account_number=&sender_bank_name=ABN+Amro&sender_bank_bic=RABXXXXX&
     sender_iban=NL17RXXXXXXXXXX12&sender_country_id=NL&recipient_holder=Gert%20Schopper&
     recipient_account_number=99XXXXXX99&recipient_bank_code=700XXXXX&recipient_bank_name=SOFORT+BanX&
     recipient_bank_bic=DEKTXXXXXX&recipient_iban=DE71700XXXXXXXXXXX99&recipient_country_id=DE&
     amount=30.00&currency_id=EUR&reason_1=Bestellnummer+1&reason_2=&user_variable_0=Ihr+Wert&
     user_variable_1=&user_variable_2=&user_variable_3=&user_variable_4=&user_variable_5=&
     created=2013-07-18+18%3A04%3A13&status=received&status_reason=credited&
     status_modified=2013-07-18+18%3A04%3A13&hash=e1f3c94a897025a5d9e6e41c33a32e6056ec981a
```

You will receive such a notification of every transaction status change.

Status messages

The notification will include the current transaction status.

A transaction will not be created with SOFORT until the customer has successfully executed the iDEAL payment. Since it may take some time to confirm the payment by iDEAL, the status "pending" may be transferred. As soon as the payment by iDEAL has been confirmed, the status changes to "received". If payment by iDEAL cannot be confirmed or if other errors are reported by iDEAL, the status "loss" is transferred. In case of a (partial) refund of the transferred amount, the status "refunded" may be active furthermore.

Detailed information on possible statuses and their descriptions can be found in the overview of parameters.

Error messages

In case the user and project ID have not been transferred correctly or not at all, the end customer is redirected to a SOFORT error page. In all other cases, if validation has failed, the end customer is redirected to the abort page of the shop to which the corresponding error codes are transferred as HTTP GET parameters.

The individual error codes and their descriptions are listed in the overview of errors in the overview of parameters. An example of an error page call with an error code might look as follows:

```
http://www.mein-shop.de/cancel.php?error_codes=7008,7014
```

Testing

You can activate the test mode for your project in the merchant area (under "My projects > Select project > Base settings> Test mode") to test the correct integration and the functionality of the HTPP(S) notification.

As soon as the test mode has been activated, the following iDEAL status messages can be activated by the following amounts:

- 1 EUR => Successful payment
- 2 EUR => Payment aborted
- 3 EUR => Expired payment (errorCode 6001)
- 4 EUR => Pending payment
- 5 EUR => Payment error (errorCode 6000)

The amounts with the corresponding status messages are also listed in a drop down list which is displayed in the SOFORT merchant area with activated test mode.

NOTE!

Real transactions cannot be carried out if test mode is active.

Functional test

In order to fully test the functionality of the iDEAL integration, please carry out a test transfer directly in your system. Please follow these steps:

1. Activate test mode (see above)
2. Place an order in your system and select iDEAL as payment method
3. Transfer test data for the individual cases
4. Check correct redirection
    - Is the customer redirected to the confirmation page after the transaction?
    - Are all notifications received correctly?
    - Is the order status set correctly and is the order created correctly?

System test (payment wizard)

In the merchant area, the tab "Test project" provides a test opportunity in your iDEAL project with SOFORT. You can specify different parameters to issue the HTML code for calling the interface and the string for hash generation. Additionally, you can call the payment wizard and go through all steps of the transfer.

This way of testing is not appropriate to simulate the entire process.

Overview of parameters

If you do not use the code library (SofortLib), you will find the XML API documentation here to retrieve bank information and the name-value-pair (NVP) interface documentation.

Description of the table columns of the subsequent tables

- **Parameter:** Name of parameter which is transferred.
- **Mandatory:** states whether a parameter must be transferred (Yes) or not (No)
- **Type (length):** specifies the data type of the parameter content; if this is a string, the maximum permissible number of characters is stated
- **Description:** detailed description of the use of the parameter

(API step 0) Retrieval of iDEAL bank information including API response

The bank information is retrieved by calling the following URL via the SOFORT interface:

```
https://www.sofort.com/payment/ideal/banks
```

The basic HTTP authentication (RFC 2617) is used for authentication. Please enter your customer number as user name (for example 12345) and your API key as password (for example a12b34cd567890123e456f7890123456), separated by ":" and coded by Base64 (base64(12345:a12b34cd567890123e456f7890123456)).

For every call, "application/xml" must be entered in the field Content Type and the field Accept in the HTTP header.

An example of a HTTP header might look as follows:

```
Authorization: Basic MTIzNDU6YTEyYjM0Y2Q1Njc4OTAxMjNlNDU2Zjc4OTAxMjM0NTY=
Content-Type: application/xml; charset=UTF-8
Accept: application/xml; charset=UTF-8
```

After successful authentication, the server will confirm your API call with HTTP 200 OK and sends an XML document with the requested information as a response:

Parameter	Number	Type (Length)	Description
ideal	[1]	Container	Identifies the response of the initial request and covers the whole message
banks	[1]	Container	List of banks
bank	[0..n]	Container	Container for a bank
code	[1]	String(255)	The BIC of a bank which must be transferred in API step 1 as "sender_bank_code"
name	[1]	String(255)	Name of bank to be displayed in the drop down field of the payment method selection

Table 1: Parameter definition of the initial call to request bank information (API step 0)

Note: This type of API call applies only to API step 0.

(API step 1) Redirection of customer to the iDEAL payment wizard (via SOFORT) and transaction parameter transfer

The name-value-pair parameters (NVP) described in the following must be transferred by HTTP POST to the following URL in the redirection of the customer to the payment wizard:

```
https://www.sofort.com/payment/ideal
```

Parameters

Parameter	Mandatory	Type (Length)	Description
user_id	Yes	Integer	Customer number of merchant
project_id	Yes	Integer	Project number of iDEAL project
sender_holder	No	String (27)	Account holder (customer)
sender_account_number	No	String (30)	Account number of customer's account
sender_bank_code	Yes	String (30)	BIC of your customer's account (see API step 0)
sender_country_id	Yes	String (2)	Country code of customer (NL)
amount	Yes	Double (8,2)	The amount to be transferred (minimum 0.10 EURO, important for test orders). No thousands separators, e.g. 1010,50 EURO
reason_1	Yes	String (27)	Reason in line 1 (a maximum of 27 characters). The reason should have different assignment characteristics for each order (e.g. order number, order date) and is therefore unambiguous. Please observe the notes at the end of the table as regards the characters to be used.
reason_2	No	String (27)	Reason for line 2 (for further assignments on the order; a maximum of 27 characters) Please observe the notes at the end of the table as regards the characters to be used.
user_variable_0	No	String (255)	Customer variable 0; for free use
user_variable_1	No	String (255)	Customer variable 1; for free use
user_variable_2	No	String (255)	Customer variable 2; for free use

user_variable_3	No	String (255)	Customer variable 3; for free use
user_variable_4	No	String (255)	Customer variable 4; for free use
user_variable_5	No	String (255)	Customer variable 5; for free use
hash	Yes	String (>=32)	Hash value for the input check (for further details, see below)
language_id	*No*	*String (2); **	*Use this parameter to determine the language of the error message; the following values are possible: NL, DE, EN, FR, ES, IT, PL*
interface_timeout	*No*	*Integer; **	*Time period in seconds to complete the transaction on the payment wizard. Values between 180 and 900 are possible.*
interface_version	*No*	*String (20); **	*Interface name, shop system name and interface version number in abbreviated form (e.g. sofort-ideal_xtc_v4.0); parameter is only relevant for shop system provider; please follow the example*

Table 2: Parameter definition of the redirection of customer to the payment wizard (API step 1)

*NOTE: This parameter is not included in the hash calculation.

We accept the contents in the parameters customer variable 0 to 5 in the API call and return them to you at the end of the payment process in the notification or (if requested) in the abort or success link.

NOTE:
Only the following characters are allowed in the parameters reason_1 and reason_2: '0-9', 'a-z', 'A-Z', ' ', '+', ',', '-', '.'. Umlauts are replaced, e.g. ä -> ae. Other characters will be removed for the display on our payment page and for notifications. Therefore, a modified string for reason_1 and reason_2 may be transmitted for redirections (success and abort link and notifications). This means for the input check (i.e. for hash calculation) in API step 1 that the check hash at SOFORT is calculated before the reason is converted; therefore this must also be carried out for hash creation in the shop system.

Furthermore, it should be remembered that the combination of both reason lines must not be longer than 32 characters and a longer text will be cut off. As both reason lines are added and separated by a space by SOFORT, 31 characters remain to be filled. Please bear this in mind for hash value calculation (you may only view the complete reason lines in your SOFORT merchant menu).

Hash calculation for the redirection to iDEAL (for API step 1)

Calculation of the hash value in **PHP** when calling the API:

```
<?php
    $data = array( '12345',        // user_id
                   '54321',        // project_id
                   '',             // sender_holder
                   '',             // sender_account_number
                   '21',           // sender_bank_code
                   'NL',           // sender_country_id
                   '30.00',        // amount
                   'Verwendung', // reason_1
                   '',             // reason_2
                   '',             // user_variable_0
                   '',             // user_variable_1
                   '',             // user_variable_2
                   '',             // user_variable_3
                   '',             // user_variable_4
                   '',             // user_variable_5
                   'geheim'        // project_password
                 );
    $data_implode = implode('|', $data);
    $hash = sha1($data_implode);
    echo $hash;
?>
```

Calculation of the hash value in **Java** when calling the API:

```
import java.security.MessageDigest;
public class HashCalculation {
    public String user_id = "";
    public String project_id = "";
    public String sender_holder = "";
    public String sender_account_number = "";
    public String sender_bank_code = "";
```

```
    public String sender_country_id = "";
    public String amount = "";
    public String reason_1 = "";
    public String reason_2 = "";
    public String user_variable_0 = "";
    public String user_variable_1 = "";
    public String user_variable_2 = "";
    public String user_variable_3 = "";
    public String user_variable_4 = "";
    public String user_variable_5 = "";
    public String project_password = "";
    public String hash() {
        String result = "";
        String input = this.user_id + "|" + this.project_id + "|" + this.sender_holder + "|" +
this.sender_account_number +
                    "|" + this.sender_bank_code + "|" + this.sender_country_id + "|" + this.amount + "|" +
this.reason_1 +
                    "|" + this.reason_2 + "|" + this.user_variable_0 + "|" + this.user_variable_1 + "|" +
this.user_variable_2 +
                    "|" + this.user_variable_3 + "|" + this.user_variable_4 + "|" + this.user_variable_5 + "|" +
this.project_password;
        StringBuffer result = new StringBuffer();
        try {
            MessageDigest messageDigest = MessageDigest.getInstance("MD5");
            byte[] md5 = messageDigest.digest(input.getBytes("UTF-8"));
            for(int i = 0; i < md5.length; i++) {
                result.append( Integer.toHexString((0xF0 & md5[i]) >> 4);
                result.append( Integer.toHexString((0x0F & md5[i])));
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        return result.toString();
    }
    public static void main(String[] args) {
        HashCalculation calculation = new HashCalculation();
        calculation.user_id = "12345";
        calculation.project_id = "54321";
        calculation.sender_bank_code = "21";
        calculation.sender_country_id = "NL";
        calculation.amount = "30.00";
        calculation.reason_1 = "Verwendung 1";
        calculation.project_password = "geheim";
```

```
        System.out.println(calculation.hash());
    }
}
```

Calculation of the hash value in **C# / ASP.NET** when calling the API:

```
public string calculateHash() {
    string p = userId.ToString() + "|" + projectId.ToString() + "|" + senderHolder + "|" +
senderAccountNumber + "|" +
            senderBankCode + "|" + senderCountryId + "|" + amount.ToString() + "|" + reason1 + "|" +
reason2 + "|" +
            userVariable0 + "|" + userVariable1 + "|" + userVariable2 + "|" + userVariable3 + "|" +
userVariable4 + "|" +
            userVariable5;
    return System.Web.Security.FormsAuthentication.HashPasswordForStoringInConfigFile(p, "sha1");
}
```

(API step 2) Transaction status change notification

The following describes the parameters which are transferred in a transaction status change notification.

Parameter	Data type	Description
transaction	String (27)	Transaction ID (unique)
user_id	Integer	Customer number of merchant
project_id	Integer	Project number of iDEAL project
sender_holder	String (255)	Sender account holder (customer) (may be returned empty)
sender_account_number	String (30)	Previous account number of customer's bank account (a value is not transferred here)
sender_bank_name	String (255)	Bank name of customer's bank account
sender_bank_bic	String (50)	BIC of customer's bank account (may be returned empty)
sender_iban	String (50)	IBAN of customer's bank account (may be returned empty)
sender_country_id	String (2)	Country code of customer
recipient_holder	String (255)	Recipient account holder (merchant)
recipient_account_number	String (30)	Account number of merchant's bank account
recipient_bank_code	String (30)	Sort code of merchant's bank account
recipient_bank_name	String (255)	Bank name of merchant's bank account

recipient_bank_bic	String (50)	BIC of merchant's bank account
recipient_iban	String (50)	IBAN of merchant's bank account
recipient_country_id	String (2)	Country code of merchant
amount	Double (8, 2)	Amount
currency_id	String (3)	Currency [EUR]
reason_1	String (255)	Reason 1
reason_2	String (255)	Reason 2
user_variable_0	String (255)	Customer variable 0
user_variable_1	String (255)	Customer variable 1
user_variable_2	String (255)	Customer variable 2
user_variable_3	String (255)	Customer variable 3
user_variable_4	String (255)	Customer variable 4
user_variable_5	String (255)	Customer variable 5
created	Datetime	Time of transfer in the format 2013-07-19 09:28:02
status	String (20)	Status of credit transfer (for characteristics see below)
status_reason	*String ()**	Details on the status of the credit transfer
status_modified	Datetime	Time of status change
hash	*String (>=32)**	*Checksum (see next sub-chapter)*
amount_refunded	*Double (8, 2)**	*Refunded amount*
amount_refunded_integer	*Integer**	*Refunded amount in the integer format*

Table 3: Parameter definition of the notification (API step 2)

*NOTE: This parameter is not included in the hash calculation.

Status of iDEAL transactions

The notification will always include the current transaction status. Usually, the transaction can immediately be confirmed by iDEAL. The status 'received' will be displayed and you can ship the goods/release the download. In case iDEAL has not confirmed the transaction immediately, the status 'pending' will be displayed. You should then wait for the final status change:

- pending - received (You can ship the goods.)
- pending - loss (The transaction could not be executed.)

In case of a (partial) refund of the transferred amount, the status "refunded" may be active furthermore.

You will be informed of the status change by email or HTTP notification (if stored with the project).

status	status_reason	Description
pending	not_credited_yet	iDEAL has not confirmed the transaction yet. You will receive a notification when the status changes.
received	credited	The transaction was successfully completed.
loss	not_credited	The transaction could not be executed.
refunded	compensation	The money has been refunded (partial refund).
refunded	refunded	The money has been refunded (full refund of total amount).

Table 4: Status messages for transactions with iDEAL (API step 2)

Hash calculation for the notification (for API step 2)

Hash value calculation in **PHP** for notifications:

```
<?php
    $data = array( 'transaction' => 'TRANSACTION',
                   'user_id' => 'USER_ID',
                   'project_id' => 'PROJECT_ID',
                   'sender_holder' => 'SENDER_HOLDER',
                   'sender_account_number' => 'SENDER_ACCOUNT_NUMBER',
                   'sender_bank_name' => 'SENDER_BANK_NAME',
                   'sender_bank_bic' => 'SENDER_BANK_BIC',
                   'sender_iban' => 'SENDER_IBAN',
                   'sender_country_id' => 'SENDER_COUNTRY_ID',
                   'recipient_holder' => 'RECIPIENT_HOLDER',
                   'recipient_account_number' => 'RECIPIENT_ACCOUNT_NUMBER',
                   'recipient_bank_code' => 'RECIPIENT_BANK_CODE',
                   'recipient_bank_name' => 'RECIPIENT_BANK_NAME',
                   'recipient_bank_bic' => 'RECIPIENT_BANK_BIC',
                   'recipient_iban' => 'RECIPIENT_IBAN',
                   'recipient_country_id' => 'RECIPIENT_COUNTRY_ID',
                   'amount' => 'AMOUNT',
                   'currency_id' => 'CURRENCY_ID',
                   'reason_1' => 'REASON_1',
                   'reason_2' => 'REASON_2',
                   'user_variable_0' => 'USER_VARIABLE_0',
                   'user_variable_1' => 'USER_VARIABLE_1',
                   'user_variable_2' => 'USER_VARIABLE_2',
```

```
                'user_variable_3' => 'USER_VARIABLE_3',
                'user_variable_4' => 'USER_VARIABLE_4',
                'user_variable_5' => 'USER_VARIABLE_5',
                'created' => 'CREATED',
                'status' => 'STATUS',
                'status_modified' => 'STATUS_MODIFIED',
                'notification_password' => 'NOTIFICATION_PASSWORD');
    $data_implode = implode('|', $data);
    $hash = sha1($data_implode);
    echo $hash;
?>
```

Hash value calculation in **Java** for notifications:

```
import java.security.MessageDigest;
public class HashCalculation {
    public String transaction = "";
    public String user_id = "";
    public String project_id = "";
    public String sender_holder = "";
    public String sender_account_number = "";
    public String sender_bank_name = "";
    public String sender_bank_bic = "";
    public String sender_iban = "";
    public String sender_country_id = "";
    public String recipient_holder = "";
    public String recipient_account_number = "";
    public String recipient_bank_code = "";
    public String recipient_bank_name = "";
    public String recipient_bank_bic = "";
    public String recipient_iban = "";
    public String recipient_country_id = "";
    public String amount = "";
    public String currency_id = "";
    public String reason_1 = "";
    public String reason_2 = "";
    public String user_variable_0 = "";
    public String user_variable_1 = "";
    public String user_variable_2 = "";
    public String user_variable_3 = "";
    public String user_variable_4 = "";
    public String user_variable_5 = "";
    public String created = "";
```

```
    public String status = "";
    public String status_modified = "";
    public String notification_password = "";
    public String hash() {
        String result = "";
        String input = this.transaction + "|" + this.user_id + "|" + this.project_id + "|" + this.sender_holder + "|" +
                       this.sender_account_number + "|" + this.sender_bank_name + "|" + this.sender_bank_bic +
"|" +
                       this.sender_iban + "|" + this.sender_country_id + "|" + this.recipient_holder + "|" +
                       this.recipient_account_number + "|" + this.recipient_bank_code + "|" +
this.recipient_bank_name + "|" +
                       this.recipient_bank_bic + "|" + this.recipient_iban + "|" + this.recipient_country_id + "|" +
                       this.amount + "|" + this.currency_id + "|" + this.reason_1 + "|" + this.reason_2 + "|" +
                       this.user_variable_0 + "|" + this.user_variable_1 + "|" + this.user_variable_2 + "|" +
                       this.user_variable_3 + "|" + this.user_variable_4 + "|" + this.user_variable_5 + "|" +
this.created + "|" +
                       this.status + "|" + this.status_modified + "|" + this.notification_password;
        StringBuffer result = new StringBuffer();
        try {
            MessageDigest messageDigest = MessageDigest.getInstance("MD5");
            byte[] md5 = messageDigest.digest(input.getBytes("UTF-8"));

            for(int i = 0; i < md5.length; i++) {
                result.append( Integer.toHexString((0xF0 & md5[i]) >> 4);
                result.append( Integer.toHexString((0x0F & md5[i])));
            }
        } catch (Exception e) {
            e.printStackTrace();
        }
        return result.toString();
    }
}
```

Error codes

The following error codes may occur in the iDEAL payment process and are returned in the replacement parameter -ERROR_CODES- (see next section for replacement parameters):

Code	Message
1000	Invalid request.
1001	Technical error.
6000	An unknown error occured.

6001	Session expired.
7007	Amount required.
7008	Invalid amount.
7009	Reason required.
7010	Invalid sender country id.
7011	Invalid recipient country id.
7012	Invalid sender bank code.
7013	Sender account equals recipient account.
7014	Invalid hash.

Table 5: Possible error codes transferred with the redirection to the abort link

NOTE!
If several errors occur at the same time, you will receive the error codes separated by "," (e.g. error_codes=7012,7013,7014).

Replacement parameters for the return

The following variables can be added to the URLs (success, abort, and notification URL) as parameter values. They will be replaced with the actual values by our system.

Parameter	Meaning
-USER_VARIABLE_0-	Customer variable 0 - 5
-USER_VARIABLE_1-	
-USER_VARIABLE_2-	
-USER_VARIABLE_3-	
-USER_VARIABLE_4-	
-USER_VARIABLE_5-	
-USER_VARIABLE_0_URLENCODE-	Customer variables 0 - 5
-USER_VARIABLE_1_URLENCODE-	
-USER_VARIABLE_2_URLENCODE-	

-USER_VARIABLE_3_URLENCODE-	
-USER_VARIABLE_4_URLENCODE-	
-USER_VARIABLE_5_URLENCODE-	
-USER_VARIABLE_0_HASH_PASS-	Customer variable 0 - 5, encoded in the selected algorithm; calculated as follows: "selected algorithm", e.g. SHA1(customer variable project password)
-USER_VARIABLE_1_HASH_PASS-	
-USER_VARIABLE_2_HASH_PASS-	
-USER_VARIABLE_3_HASH_PASS-	
-USER_VARIABLE_4_HASH_PASS-	
-USER_VARIABLE_5_HASH_PASS-	
-USER_ID-	Your customer number
-PROJECT_ID-	Your project number
-TRANSACTION-	*Transaction ID (generated in every wizard call)**
-CURRENCY_ID-	Transaction currency (EUR)
-SENDER_HOLDER-	Name of sender
-SENDER_ACCOUNT_NUMBER-	Account number of sender account
-SENDER_ACCOUNT_NUMBER_XXX-	Account number : Last 3 digits replaced by XXX
-SENDER_BANK_NAME-	Bank name of sender account
-SENDER_BANK_BIC-	BIC of sender account
-SENDER_IBAN-	IBAN of sender account
-SENDER_COUNTRY_ID-	Country code of sender account (NL)
-SENDER_CITY-	City of account holder
-SENDER_CITY_URLENCODE-	City of account holder, URL-encoded
-RECIPIENT_HOLDER-	Name of recipient's account
-RECIPIENT_ACCOUNT_NUMBER-	Account number of recipient
-RECIPIENT_ACCOUNT_NUMBER_XXX-	Account number of recipient: Last 3 digits replaced by XXX
-RECIPIENT_BANK_CODE-	Sort code of recipient's account
-RECIPIENT_BANK_NAME-	Bank name of recipient's account
-RECIPIENT_BANK_BIC-	BIC of recipient's account
-RECIPIENT_IBAN-	IBAN of recipient's account
-RECIPIENT_COUNTRY_ID-	Country code of recipient's account (DE)

-AMOUNT-	Invoice amount
-AMOUNT_INTEGER-	Invoice amount as an integer (whole numbers)
-REASON_1-	Reason line 1
-REASON_2-	Reason line 2
-DATE-	Date of transaction
-TIME-	Time of transaction
-TIMESTAMP-	Time stamp of transaction
-SENDER_HOLDER_URLENCODE-	Name of sender, URL-encoded
-SENDER_BANK_NAME_URLENCODE-	Bank name of sender, URL-encoded
-RECIPIENT_HOLDER_URLENCODE-	Name of recipient, UTL-encoded
-RECIPIENT_BANK_NAME_URLENCODE-	Bank name of recipient, UTL-encoded
-REASON_1_URLENCODE-	Reason (line 1), URL-encoded
-REASON_2_URLENCODE-	Reason (line 2), URL-encoded
-ERROR_CODES-	Error messages (for detailed table, see return parameters -ERROR_CODES-)
-FEES-	Cost of transaction
-STATUS-	Current status of transaction
-STATUS_MODIFIED-	Time of status change
-STATUS_REASON-	Reason for status change
-AMOUNT_REFUNDED-	*Refunded amount***
-AMOUNT_REFUNDED_INTEGER-	*Refunded amount in the integer format***

Table 6: Replacement parameters to be used for the success, abort, and notification link

* Cannot be used for the abort link.
** Please bear in mind that this parameter can only be used in a HTTP notification with a payment status and is only replaced when you receive a notification of a cancelled transaction.

NOTE:

Please observe the order for the USER_VARIABLES. If you for example transfer the success link in -USER_VARIABLE_1- which includes -USER_VARIABLE_0- as a replacement parameter, the replacement would fail.

Example

Success link with dynamic URL (USER_VARIABLES must be transferred in the call):

```
http://-USER_VARIABLE_0-/index.php?meineSession=-USER_VARIABLE_1-&transID=-TRANSACTION-
```

The merchant must then enter this link as a success link in his/her project settings in the SOFORT merchant menu.

Support & Contact

The SOFORT team will be available if you need help.

You may either send us an email at service@sofort.com or fill out our support form at https://www.sofort.com/eng-DE/general/verkaeufer/kontakt/.

We are also glad to assist you in case of technical issues:

Technical support:
Phone: +49 89 20 20 889-400
Email: integration@sofort.com

Business hours:
Monday to Thursday: 8:30 a.m. to 6:00 p.m.
Friday: 8:30 a.m. to 5:00 p.m.

Legal Notice

SOFORT GmbH
Fußbergstraße 1
82131 Gauting
Germany

Support for customers
Phone: +49 89 20 20 889-0

Support for merchants
Phone: +49 89 20 20 889-500
Fax: +49 89 20 20 889 -599

info@sofort.com
www.sofort.com

Directors

Marc Berg, Dr. Jens Lütcke, Georg Schardt

External Privacy Policy Officer
Mr. Tobias Kohl, LL.M. (lawyer)
For privacy questions please contact us at: datenschutz@sofort.com

Registered at the District Court Munich
HRB 218675
VAT-ID: DE248376956