Sips e-payment solution

Technical integration guide

Version 1.4

Table of Contents

1. Preface

1.1. Intended audience

This document is intended for developers and technical profiles implementing Sips.
Installation requires knowledge of at least one programming language, such as PHP, .NET or Java.

1.2. Where to get additional information

For information on the administrative tasks you can perform regarding transactions, see the Full User guide that is available on our website.

1.3. Support

Customer Support is available from 9h00 to 17h00.
Contact support:

E-mail: supportsips-benelux@worldline.com

1.3. Glossary

3-D Secure: Common technological standard (3 Domain Secure) of Visa and MasterCard, set up to make online credit card payments more secure. For commercial reasons, Visa and MasterCard use different brand names: Verified by Visa and MasterCard SecureCode.

Buyer: The buyer is an Internet user who connects to the Merchant's website and pays for a given good or service.

Acquirer: The financial establishment who receives financial information pertaining to a transaction from the acceptor (the merchant, its payment service provider) and enters this information into an exchange system.

Authorization request: Verification of the validity of the cardholder's card with financial institutions. In addition to verifying the card number, this request involves verifying that the card really is a valid payment method, that it has not been stopped and that the amount presented will be cleared.

Capture: Please see *collection.*

(Card) issuer: Institution who has issued an payment card to a cardholder.

Card security code, CVV2, CVC2 or CBN2 (Visual card security code): 3-digit key located on the signature strip on the back of Visa and Mastercard. This adds an additional level of security for remote sales. On American Express cards, the card security code is a 4-digit number (4DBC).

Collection: Collection operation completed in payment of the transactions, meaning the credit/debit of the merchant's account and the debit/credit of the Internet user's account. The capture of a transaction means that it will be collected and will therefore be sent to the bank's merchant processing centre.

Host: Service company which hosts one or more websites on its own servers connected to the Internet. A host occasionally provides website creation/management services.

Internet user: Online Client of the Merchant.

Merchant: Individual person or legal entity who has an online shop. Sips merchants are registered with Worldline and can use the secure online payment service.

merchantID: Unique merchant identifier used by Worldline

Networks (electronic payment): Group of bodies who issue payment methods after having entered into a reciprocal exchange agreement for cardholder (issuer) and merchant (acceptor) movements.

Operations log: Log generally sent to the merchant via email on a daily basis and which contains all the operations made by the merchant on the Sips Office Extranet interface or using the Sips Office Connect connector (reimbursement, validation, cancellation operations etc.) since the previous day's log was sent.

Preproduction: Stage during which the merchants use their production certificate which was given to them when they created their Sips shop. Preproduction tests make it possible to validate that the merchant's contract is operational.

Secret key: Unique value which makes it possible to ensure the confidentiality and integrity of payment via the Internet.

Secure Payment: Transactions booked on the Internet are protected from unauthorized interceptions and also from unauthorized edits and alterations to the original content of messages.

Sips: International multi-channel secure payment solution provided by Worldline.

Transaction log: Log sent to the merchant on a daily basis, generally via email, containing all the transactions made on a given website since the previous day's log was sent.

Transaction reconciliation log: Log which may be sent to the merchant on a daily basis. This log makes it possible to reconcile transactions booked by the merchant in their Sips shop against transactions which were in fact processed by their banking establishment's merchant processing center. This corresponds to what will actually be credited to/debited from their account and alerts the merchant in case of a non-reconciled transaction. This log makes accounting easier for merchants.

transactionReference: Characteristic identifier for each transaction. The merchant can monitor the progress of each transaction using the TREF.

1.4. Acronyms

Acronym	Meaning
CTC	Merchant Processing Centre
CVV	Card verification value (Visa)
CVC	Card validation code (Mastercard)
HTTPS	Hypertext Transfer Protocol Secure
ISO	International Standard Organisation
PAN	Personal Account Number
TREF	TransactionReference

2. Introduction

This document describes how to integrate Sips with the merchant'site (webshop).

The introduction will tell you what you need to know before getting started with integrating Sips with your webshop.

Protocol definition and the provided examples will provide all the information you need to integrate Sips with your site. Testing will provide all relevant information to test your integration properly before going live. If you should have any questions regarding the integration of your webshop with Sips, please do not hesitate to contact the Sips customer support team.

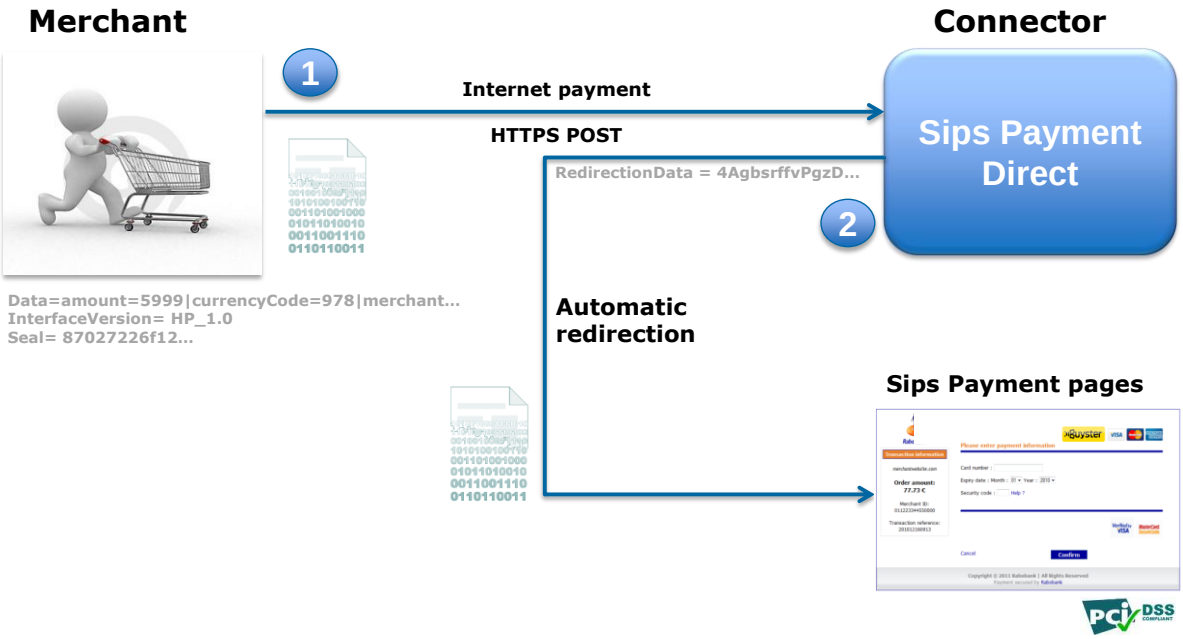
2.1. Purpose of the document

The purpose of the present document is to explain how to implement the Sips Payment Direct solution and how to begin the initial payment tests.

This document is aimed at all Merchants who wish to subscribe to the Sips offer and wish to use a connector based on HTTP(s) POST exchanges between Merchant websites and the Sips servers, while using Sips Payment Direct as a gateway. It is aimed at the Merchants technical team, not the business team.

This connector endeavors to be as Plug & Play as possible for the Merchant.

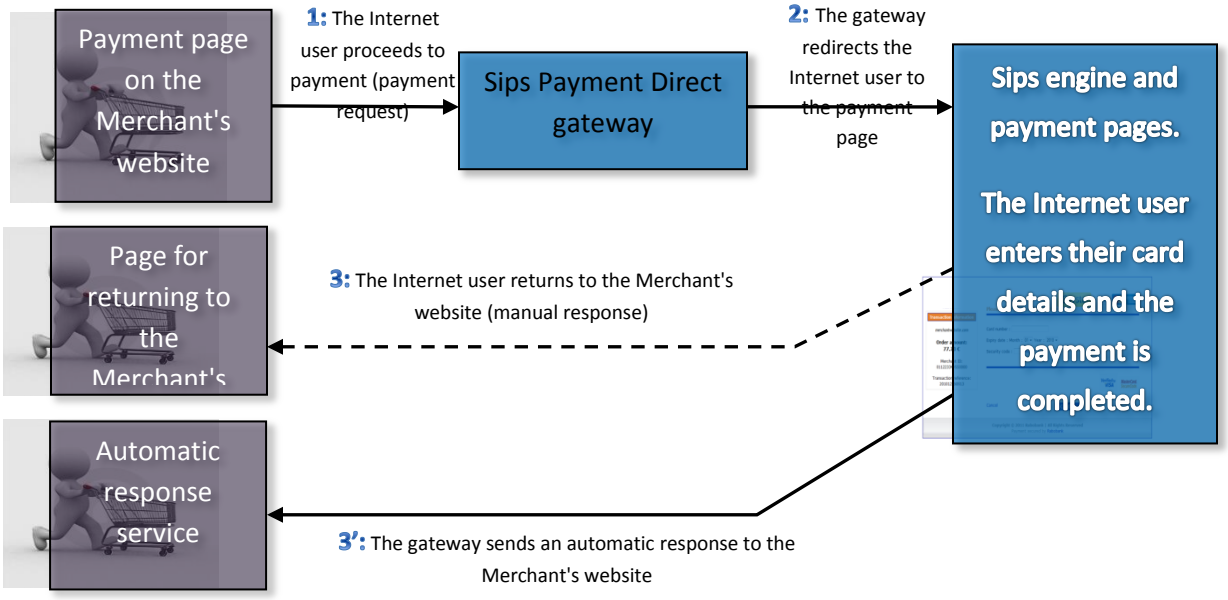
2.2. General principle



1. When the Internet user confirms their basket, they are redirected towards the Sips Payment servers. The payment request is then checked and, if valid, it is encrypted (named RedirectionData in the system).
2. The Internet user is then automatically redirected towards the Sips Payment pages, with the encrypted request. This request is decrypted and the Sips Payment page invites the Internet user to enter the information for their method of payment.

2.3. Payment flow

There are three flows to implement between the Merchant's website and the payment server in order to integrate the solution.



Step 1: Once the Internet user proceeds to the payment stage, a payment request must be sent to the Sips Payment Direct gateway. Worldline gives the Merchant the gateway URL. The best method of managing this call is to send a form as a POST HTTPS, but any other solution which could send a POST HTTPS request would also work.

Step 2: The Sips Payment Direct gateway will redirect the calling application to the Sips payment pages. The Internet user must enter the details for the payment method so that the Sips payment server can process the transaction. It is worth noting that payment details can be entered directly on the server which issued the payment method (for example: Credit transfer or PayPal). At the end of the payment process, regardless of whether it was successful or not, two responses are created and sent to the response URLs provided during flow 1.

There are two separate notification procedures:

- **Step 3:** *Manual responses* are sent as HTTP(S) POSTs by the payment server to the normal response *normal response* URL provided during the payment request when the Internet user clicks on "Return to shop" on the payment page. This is why the *normal response* URL is also the destination page where the Internet user is redirected at the end of payment. There is no guarantee that the Internet user will click on this link, consequently there is no guarantee that the *manual response* will be received.
- **Step 3':** *Automatic responses* are sent separately to manual responses. They also use HTTP(s) POSTs requests sent by Sips payment servers but via the *automatic response* URL, in this case

provided during the payment request. This means that the Merchant will always receive this response as soon as the payment is completed on the Sips payment pages.

2.4. Security

Sips is PCI DSS-compliant (Payment Card Industry Data Security Standard). This means that the response message contains no customer information such as bank account or credit card number, PAN number or other customer information. Instead, a unique transaction reference number (transactionReference) is used to match the response message to the order in the webshop and the appropriate customer. We also recommend using an order number as an extra idenfication field in the payment request (orderID).

2.5. Secret Key

The payment request and the response message between the webshop and Sips are exchanged securely, thanks to the use of a secret key. You can find the Sips secret key on the downloadsite: https://download.sips-atos.com.

After the signed Sips contract is received by Worldline, the technical contact person will receive the username for the downloadsite by e-mail. The password will be sent to the contract requester. For more information about the downloadsite, please consult the user manual.

For the test environment you do not need your own secret key to install and test. In this environment you can use the general test merchant ID and its corresponding secret key. See chapter testing for more information.

3. Description of the protocol

3.1. Post fields

3 mandatory fields are provided during the payment requests and responses.

Data	Contains all the information related to the transaction, gathered in a character chain as described in section Data field
InterfaceVersion	Version of the connector interface.
Seal	Used to validate the integrity of the data exchanged. The Seal field is calculated using the Data field and the secret key field, as described in section Seal field

An additional optional field is available:

Encode Provides the coding used in the Data field, as described in section Encode field

3.1.1. Data field

The Data field is built according to the following format:

<field name>=<value name>|<field name>=<value name>|<field name>=<value name> etc.

All the fields necessary for the transaction (please see details in the data dictionary) must be present in this character chain. The order of the fields is irrelevant.

Example of a payment request:

```
amount=55|currencyCode=978|merchantId=011223744550001|normalReturnUrl=http://www.normalreturnurl.com|transactionReference=534654|keyVersion=1
```

3.1.2. Seal field

The value of the Seal field is built as follows:

- Concatenation of the Data field and the secret key (encoded if the encoding option is used. See section Encode field)
- Obtaining the UTF-8 encoding of the data for the previous result
- SHA256 encrypting of the bytes obtained

This procedure can be summarized as follows:

```
SHA256( UTF-8(Data+secretKey ) )
```

3.1.3. Encode field

In the event that special characters appear in the Data field, the value of this field must be encoded.

Two encoding formats can be used: base64 or base64Url.

It is worth noting that, as the calculation of the signature is made in the Data field, when encoding is applied, it is the encoded value for the Data field which is used for this calculation.

4. How to make a payment

4.1. Payment request

The payment request is an HTTP POST call to the connector on the payment gateway. The simplest way of making this call is an HTML form, using the POST method.

4.1.1. Fields for the payment request

All the data for the payment request must be provided as outlined in chapter Description of the protocol.

InterfaceVersion must be set to **HP_2.3.**

The data dictionary and the message description chapter describe all the settings for the payment request, their format as well as whether they are mandatory or optional.

4.1.2. Example

A form example is shown below:

```
<form method="post" action="https://url.to.sips.server/paymentInit">

   <input type="hidden" name="Data"
value="amount=55|currencyCode=978|merchantId=011223744550001|normalReturnUrl=http://www.normalreturnurl.com|transactionReference=534654|keyVersion=1">

   <input type="hidden" name="InterfaceVersion" value="HP_2.3">

   <input type="hidden" name="Seal"
value="21a57f2fe765e1ae4a8bf15d73fc1bf2a533f547f2343d12a499d9c0592044d4">

   <input type="submit" value="Proceed to payment">

 </form>
```

4.1.3. Error management

All the fields received through the connector for the Sips Payment Direct gateway are checked individually. The list of error messages which may be displayed during this verification stage as well as the solutions to implement are described in the table below.

Message	Cause	Solution
Unknown version interface: <version>	The value <version> in the field POST InterfaceVersion is unknown	Check the interface version in this user guide
Invalid keyword: <name param >=<value param>	The request contains a <name param> setting, which is not expected in the payment request	Check the settings for the payment request in the data dictionary
Invalid field size: <name param >=<value param>	The value of the setting <name param> is not the correct length	Check the length of the settings for the payment request in the data dictionary
Invalid field value: <name param >=<value param>	The value of the setting <name param> is not in the correct format	Check the format of the settings for the payment request in the data dictionary
Mandatory field missing: <name param >	The mandatory setting <name of the param >is missing from the payment request	Check the mandatory settings for the payment request in the data dictionary.
Unknown security version: <version>	The value <version> in the keyVersion setting is unknown	Check the versions of keys which are available on the Merchant interface
Invalid signature	Verification of the signature for the payment request has failed. This may be due to incorrect calculation of the signature or may indicate that some fields were falsified after the signature was calculated.	Check the signature calculation regulations in the data dictionary.
Transaction already processed: <transaction reference>	A payment request with the same transactionReference has already been received and processed by the Sips servers	Check that the transactionReference is unique for a given transaction
<Other messages>	In case of technical errors, there can be various other error messages.	Contact technical support.

4.2. Response message

Two types of responses are possible. Although the protocols, formats and content of the two responses are exactly the same, the two responses must be managed differently as they meet two different needs.

4.2.1. Manual response

The main goal of the manual response is to redirect the Internet user to the Merchant's website with the payment result, so that the Merchant can make the right decision as regards their customer. For example, in case of an error, the Merchant may suggest trying the payment again and relaunching the process. In case of a successful payment, the Merchant may display a thank you message and begin to deliver the merchandise, if need be.

The last stage in the Sips Payment payment process involves displaying a redirection link to the customer. When the Internet user clicks on this link, the Sips server redirects them to the URL contained in the field normalReturnUrl, which is provided at the beginning of the payment process. Redirection is an HTTP POST request which contains the response settings as described in section. It is the Merchant's responsibility to recover these settings and check the signature, thus ensuring the integrity of the response data. It is also the Merchant's responsibility to display relevant messages to their customer which relate to the response details.

It is important to note that receipt of the response is not guaranteed, as it is sent by the Internet user's browser. Basically, the final user has the option not to click on the link, or the Internet user's Internet connection may simply encounter a problem and may block the transmission of this response. Consequently, the Merchant's business processes must not be based solely on this response.

4.2.2. Automatic response

An automatic response is only sent if the field automaticResponseUrl was sent in the payment request. If this is the case, the Sips server sends an http POST response to the URL received. The response fields are identical to those sent to the manual response. The only difference between the two procedures is that the automatic response is sent directly by the Sips server, without passing through the Internet user's browser. Consequently, it is much more reliable as it will always be sent. Another consequence is that the procedure in charge of the receipt of this response must not try to respond to the calling application. Basically, the Sips server does not wait for any response following the transmission of the automatic response.

As with the manual response, the fields for the automatic response are described in section Respons message. It is the Merchant's responsibility to recover the response settings, register them in encrypted form, check the signature to ensure the integrity of the response fields and therefore update the Merchant back office as a result.

4.2.3. Payment request response codes

Hereby all response codes related to the payment request.

Value	Description
00	Authorisation accepted
02	Authorisation request to be performed via telephone with the issuer, as the card authorisation threshold has been exceeded, if the forcing is authorised for the merchant
03	Invalid distance selling contract
05	Authorisation refused
12	Invalid transaction, verify the parameters transferred in the request.
14	invalid bank details or card security code
17	Buyer cancellation
24	Operation impossible. The operation the merchant wishes to perform is not compatible with the status of the transaction.
25	Transaction not found in the Sips database
30	Format error
34	Suspicion of fraud
40	Function not supported: the operation that the merchant would like to perform is not part of the list of operations for which the merchant is authorised
51	Amount too high
54	Card is past expiry date
60	Transaction pending
63	Security rules not observed, transaction stopped
75	Number of attempts at entering the card number exceeded
90	Service temporarily unavailable
94	Duplicated transaction: for a given day, the TransactionReference has already been used
97	Timeframe exceeded, transaction refused
99	Temporary problem at the Sips Office Server level

4.2.4. Acquirer response codes

Hereby all response codes related to the issuing request.

Value	Description
00	Transaction approved or processed successfully
02	Contact card issuer
03	Invalid acceptor
04	Retain the card
05	Do not honour
07	Retain the card, special circumstances
08	Approve after obtaining identification
12	Invalid transaction
13	Invalid amount
14	Invalid cardholder number
15	Card issuer unknown
30	Format error
31	Identifier of acquirer entity unknown
33	Card is past expiry date
34	Suspicion of fraud
41	Card lost
43	Card stolen
51	Insufficient funds or credit limit exceeded
54	Card is past expiry date
56	Card missing from file
57	Transaction not permitted for this cardholder
58	Transaction prohibited at terminal
59	Suspicion of fraud
60	The acceptor of the card must contact the Acquirer
61	Exceeds the withdrawal amount limit
63	Security rules not observed
68	Response not received or received too late
90	Momentary system crash
91	Card issuer inaccessible
96	System functioning incorrectly
97	Expiry of the global monitoring delay
98	Server unavailable network routing further request
99	Incident field initiator

4.2.5. Problems with receiving a response

You will find below a list of the most common problems encountered preventing the reception of automatic and manual responses. Please make sure to have them checked before initiating a call with the helpdesk .

- Check that the URLs answers are provided in the application for payment and are valid. You can simply copy / paste the URLs into your browser to check their validity.
- The URLs provided must be accessible from the outside and therefore the internet. A (password login or IP filter /) access control or a firewall can block access to your server.
- Access to URLs answers should appear in the notification log of your web server.
- If you use a non-standard port , it must be in the range 80 to 9999 to be compatible with Sips .
- You cannot add contextual parameters to URLs answers. The orderId field is deemed to receive these additional parameters , or alternatively a sessionId that allow the merchant to find customer information at the end of the payment process .

4.2.6. Error Handling - No signature in the response

There are cases where the error Sips server is not capacity to sign the response message. For example, in a "MerchantID unknown" error or if the secret key is unknown to the Sips repository. For these particular reasons, the payment server will send the response without signing in Seal field.

5. How to sign a message

5.1. Reason for signing a message

The payment request contains the transaction settings and is sent through the Internet user's browser. It is theoretically possible for a hacker to intercept the request and change the settings before the data reaches the payment server.
Therefore, it is necessary to add security to ensure the integrity of the transaction settings sent. The Sips solution meets this need by exchanging signatures.
A successful signature check involves two things:

- The **integrity** of the request and response messages, no alteration during exchange
- The **authentication** of the issuer and receiver, as they share the same secret key.

5.2. Method used for signing a message

The signature operation is completed by calculating the encrypted value based on the transaction's settings (the Data field) to which the secret key (unknown by the Internet user) is added. All character chains are converted to UTF8 before being encrypted.

The encrypting algorithm (SHA256) produces an irreversible result. Generally, when such a message is received, the message receiver must recalculate the encrypted value in order to compare it with the value received. Any difference indicates that the data exchanged was falsified.

The result must be sent in hexadecimal form in the POST field, named Seal.

5.3. Code examples

5.3.1. Php 5

```
<?php
echo hash('sha256', $data.$secretKey);
?>
```

Data and secretKey must use a UTF-8 character set. Refer to the **utf8_encode** function to convert from ISO-8859-1 to UTF-8.

5.3.2. Java

```
import java.security.MessageDigest;



public class ExampleSHA256 {

        /**
         * table to convert a nibble to a hex char.
         */
        static final char[] hexChar = {
          '0' , '1' , '2' , '3' ,
          '4' , '5' , '6' , '7' ,
          '8' , '9' , 'a' , 'b' ,
          'c' , 'd' , 'e' , 'f'};
        /**
         * Fast convert a byte array to a hex string
```

```
 * with possible leading zero.
 * @param b array of bytes to convert to string
 * @return hex representation, two chars per byte.
 */
public static String encodeHexString ( byte[] b )
  {
  StringBuffer sb = new StringBuffer( b.length * 2 );
  for ( int i=0; i<b.length; i++ )
    {
    // look up high nibble char
    sb.append( hexChar [( b[i] & 0xf0 ) >>> 4] );

    // look up low nibble char
    sb.append( hexChar [b[i] & 0x0f] );
    }
  return sb.toString();
  }

/**
 * Computes the seal
 * @param Data the parameters to cipher
 * @param secretKey the secret key to append to the parameters
 * @return hex representation of the seal, two chars per byte.
 */
public static String computeSeal(String Data, String secretKey) throws Exception
{
  MessageDigest md = MessageDigest.getInstance("SHA-256");

  md.update((Data+secretKey).getBytes("UTF-8"));
```

```
        return encodeHexString(md.digest());
    }

    /**
     * @param args
     */
    public static void main(String[] args) {response
        try {
            System.out.println (computeSeal("parameters", "key"));
        } catch (Exception e) {
            e.printStackTrace();
        }
    }

}
```

5.3.3..net

(Completed using a simple form called "Form 1" containing two text fields for entering: txtSips, txtSecretKey and another for displaying: lblHEX)

```
using System;
using System.Collections.Generic;
using System.ComponentModel;
using System.Data;
using System.Drawing;
using System.Text;
using System.Windows.Forms;
using System.Security.Cryptography;

namespace ExampleDotNET
{
  public partial class Form1 : Form
```

```
  {
    public Form1()
    {
      InitializeComponent();
    }

    private void cmdGO_Click(object sender, EventArgs e)
    {
      String sChaine = txtSips.Text + txtSecretKey.Text;
      UTF8Encoding utf8 = new UTF8Encoding();
      Byte[] encodedBytes = utf8.GetBytes(sChaine);

      byte[] shaResult;
      SHA256 shaM = new SHA256Managed();
      shaResult = shaM.ComputeHash(encodedBytes);

      lblHEX.Text = ByteArrayToHEX(shaResult);
    }

    private string ByteArrayToHEX(byte[] ba)
    {
      StringBuilder hex = new StringBuilder(ba.Length * 2);
      foreach (byte b in ba)
        hex.AppendFormat("{0:x2}", b);
      return hex.ToString();
    }

  }
}
```

6 Message description

6.1 Payment request

6.1.1 Generic fields

Field	Required
captureDay	No
captureMode	No
currencyCode	Yes
merchantId	Yes
normalReturnUrl	Yes
amount	Yes
transactionReference	Yes
keyVersion	Yes
automaticResponseUrl	No
customerId	No
customerIpAddress	No
customerLanguage	No
expirationDate	No
hashSalt1	No
hashSalt2	No
hashAlgorithm1	No
hashAlgorithm2	No
invoiceReference	No
merchantSessionId	No
merchantTransactionDateTime	No
merchantWalletID	No
orderChannel	No
orderId	No
paymentMeanBrandList	No
paymentPattern	No
returnContext	No
statementReference	No
templateName	No
transactionActors	No
transactionOrigin	No

6.1.2 Optional fields related to fraude detection

Field	Required
fraudData.allowedCardArea	No

fraudData.allowedCardCountryList	No
fraudData.allowedIpArea	No
fraudData.allowedIpCountryList	No
fraudData.bypass3DS	No
fraudData.bypassCtrlList	No
fraudData.bypassInfoList	No
fraudData.deniedCardArea	No
fraudData.deniedCardCountryList	No
fraudData.deniedIpArea	No
fraudData.deniedIpCountryList	No

6.1.3 Optional fields related to payment in N instalment

Field	Required
instalmentData.number	No
instalmentData.datesList	No
instalmentData.transactionReferencesList	No
instalmentData.amountsList	No

6.1.4 Optional fields related to the payment page

Field	Required
paypageData.bypassReceiptPage	No

6.1.5 Optional fields related to authentication

Field	Required
authenticationData.check3DS	No
authenticationData.checkCSC	No

6.1.6 Optional fields related to the payment mean Paypal

Field	Required
paymentMeanData.paypal.landingPage	No
paymentMeanData.paypal.addrOverride	No
paymentMeanData.paypal.invoiceId	No
paymentMeanData.paypal.dupFlag	No
paymentMeanData.paypal.dupDesc	No

paymentMeanData.paypal.dupCustom	No
paymentMeanData.paypal.dupType	No
paymentMeanData.paypal.mobile	No

6.1.7 Optional fields related to the payment mean SDD

Field	Required
paymentMeanData.sdd.mandateAuthentMethod	No
paymentMeanData.sdd.mandateUsage	No

6.1.8 Optional fields for billing data

6.1.8.1 Billing address

Field	Required
billingAddress.addressAdditional1	No
billingAddress.addressAdditional2	No
billingAddress.addressAdditional3	No
billingAddress.city	No
billingAddress.company	No
billingAddress.country	No
billingAddress.postBox	No
billingAddress.state	No
billingAddress.street	No
billingAddress.streetNumber	No
billingAddress.zipCode	No

6.1.8.2 Billing address

Field	Required
billingContact.email	No
billingContact.firstname	No
billingContact.gender	No
billingContact.lastname	No
billingContact.mobile	No
billingContact.phone	No
billingContact.title	No

6.1.9 Optional fields for customer data

6.1.9.1 Customeraddress

Field	Required
customerAddress.addressAdditional1	No
customerAddress.addressAdditional2	No
customerAddress.addressAdditional3	No
customerAddress.city	No
customerAddress.company	No
customerAddress.country	No
customerAddress.postBox	No
customerAddress.state	No
customerAddress.street	No
customerAddress.streetNumber	No
customerAddress.zipCode	No

6.1.9.2 Customer contact

Field	Required
customerContact.email	No
customerContact.firstname	No
customerContact.gender	No
customerContact.lastname	No
customerContact.mobile	No
customerContact.phone	No
customerContact.title	No

6.1.9.3 Customer data

Field	Required
customerData.birthCity	No
customerData.birthCountry	No
customerData.birthDate	No
customerData.birthZipCode	No
customerData.nationalityCountry	No
customerData.newPwd	No
customerData.pwd	No

6.1.10 Optional fields for delivery

6.1.10.1 Delivery address

Field	Required
deliveryAddress.addressAdditional1	No
deliveryAddress.addressAdditional2	No
deliveryAddress.addressAdditional3	No
deliveryAddress.city	No
deliveryAddress.company	No
deliveryAddress.country	No
deliveryAddress.postBox	No
deliveryAddress.state	No
deliveryAddress.street	No
deliveryAddress.streetNumber	No
deliveryAddress.zipCode	No

6.1.10.2 Delivery contact

Field	Presence (M/O)
deliveryContact.email	O
deliveryContact.firstname	O
deliveryContact.gender	O
deliveryContact.lastname	O
deliveryContact.mobile	O
deliveryContact.phone	O
deliveryContact.Title	O

6.2 Payment response (automatic and manual)

The content of the manual response is the same for automatic response. The contents follow the result of the payment (with success or not).

Fieldname	Required	Included in version	Comment

acquirerResponseCode	YES	HP_2.0	
Amount	YES	HP_1.0	Value is passed in the payment request.
autorisationId	YES	HP_1.0	Value is passed in the payment request.
captureDay	YES	HP_1.0	Value is passed in the payment request.
captureLimiteDate	NO	HP_2.3	
captureMode	YES	HP_1.0	Value is passed in the payment request.
cardCSCResultCode	YES	HP_2.0	
complementaryCode*	NO	HP_1.0	
complementaryInfo*	NO	HP_2.0	
currencyCode	YES	HP_1.0	Value is passed in the payment request.
customerEmail	YES	HP_2.0	Value is passed in the payment request. Only available for version HP_2.0
customerId	YES	HP_2.0	Value is passed in the payment request.
customerIpAddress	YES	HP_2.0	Value is passed in the payment request.
customerMobilePhone	YES	HP_2.1	Value is passed in the payment request. Only available for version HP_2.1
dccStatus*	NO	HP_2.3	
dccResponseCode*	NO	HP_2.3	
dccAmount*	NO	HP_2.3	
dccExchangeRate*	NO	HP_2.3	
dccProvider*	NO	HP_2.3	
dccCurrencyCode*	NO	HP_2.3	
guaranteeIndicator	NO	HP_2.0	
hashPan1	NO	HP_2.3	
hashPan2	NO	HP_2.3	
holderAuthentMethod*	NO	HP_2.4	

holderAuthentRelegation*	NO	HP_2.0	
holderAuthentStatus*	NO	HP_2.0	
interfaceVersion*	NO	HP_1.0	
issuerEnrollementIndicator*	NO	HP_2.0	
keyVersion	YES	HP_1.0	Value is passed in the payment request.
maskedPan*	NO	HP_1.0	
merchantId	YES	HP_1.0	Value is passed in the payment request.
merchantSessionId	YES	HP_2.0	Value is passed in the payment request.
merchantTransactionDateTime	YES	HP_2.0	Value is passed in the payment request.
merchantWalletID	YES	HP_2.0	Value is passed in the payment request.
orderChannel	YES	HP_2.0	Value is passed in the payment request.
orderId	YES	HP_1.0	Value is passed in the payment request.
panEntryMode**	NO	HP_2.4	
panExpiryDate*	NO	HP_2.0	
paymentMeanBrand*	NO	HP_1.0	
paymentMeanData.sdd*	NO	HP_2.3	
paymentMeanType*	YES	HP_1.0	
paymentPattern	YES	HP_2.0	Value is passed in the payment request.
responseCode	YES	HP_1.0	
returnContext	NO	HP_1.0	Value is passed in the payment request.
scoreColor*	NO	HP_2.0	
scoreInfo*	NO	HP_2.0	
scoreProfile*	NO	HP_2.0	
scoreThreshold*	NO	HP_2.0	
scoreValue*	NO	HP_2.0	
statementReference*	NO	HP_2.4	Value is passed in the payment request.

tokenPan*	NO	HP_2.0	
transactionActor	YES	HP_2.2	Value is passed in the payment request.
transactionDateTime	YES	HP_1.0	
transactionOrigin	YES	HP_2.0	Value is passed in the payment request.
transactionReference	YES	HP_1.0	Value is passed in the payment request.
walletType	NO	HP_2.4	

* These fields are provided when available, depending on the status of the transaction and the payment method chosen.

7. Testing

7.1. How to test

The test and integration stages can be completed using the pooled demonstration environment.

The technical details required to use this environment are described below:

Connector demo URL atos.com/paymentInit	https://payment-webinit.simu.sips-
Merchant ID	002001000000001
Version of the key	1
Secret key	002001000000001_KEY1

In the simulation environment, the authorization process is simulated. This means that it is not necessary to use the real payment methods in order to complete these tests.

7.2. Testing card transactions

When you select Visa, Mastercard or Maestro, you will be redirected towards the card information page, where you can enter your card details.

The following simulation regulations apply to all cards:

- The PAN must be between 16 and 19 digits long.
- The first six digits of the PAN determine the type of card, according to the table below:

Card type	Beginning of the card number
VISA	410000
Mastercard	510000
Maestro	500000

- You can simulate all response codes (cf. data dictionary) by changing the last two digits.
- The security code is three or four digits long. This value is irrelevant to the result of the transaction.

Example: if you use card number 4100000000000005, the card will be identified as a VISA and the payment will be refused (response code 05).

7.3. Testing iDEAL transactions

When you select iDEAL, you are redirected to the iDEAL simulation server, which simulates an iDEAL transaction according to the transaction amount. Then you return to the payment server which displays the ticket with the transaction result.

iDEAL simulation regulations:

Transaction amount	iDEAL response
EUR 2.00	Transaction cancelled
EUR 3.00	Transaction expired
EUR 4.00	Transaction not completed
EUR 5.00	Transaction failure
Other cases	Transaction OK

8. Going Live

The next step is to connect to the production environment for the real start-up.

In order to do this, the Merchant must change the payment server URL and use the Merchant IDs received during the registration stage.

8.1. Merchant IDs

The URL for the production payment server is: https://payment-webinit.sips-atos.com/paymentInit

To access the production environment, you will need the following three pieces of information:

- The Merchant ID (**merchantID)** which identifies the eCommerce site on the Sips payment server
- The version (**keyVersion)** of the secret key
- The secret key (**secretKey)** used to sign requests and verify responses

The Merchant ID (**merchantID)** is provided at the end of the Merchant registration stage.

You can download the version of the key (**keyVersion)** and the secret key (**secretKey)** from the extranet https://download.sips-atos.com using the username and password provided by technical support at the end of the Merchant registration stage.

8.2. Production validation

Once the Merchant starts using their own IDs on the production server, any transactions performed are real from end-to-end, up until the funds are credited to the Merchant's account and debited from the Buyer's account.

Before the shop is really opened to the public, the Merchant may submit a request to validate the end-to-end payment, up until the funds are credited to the Merchant's account and debited from the Buyer's account.