



**Migration Guide for HMAC SHA256 signature
Web Service Connection**

TECHNICAL SERVICE VIRTUAL POS

Phone: 902 365 650 option 2

tpvvirtual@bancsabadell.com

Special attention for the migration SHA1 to SHA2

Monday to Friday from 10h to 19h

TABLE OF CONTENTS

1. Introduction.....	3
1.1 Purpose.....	3
2. Summary of the differences of the new model (HMAC SHA-256) as compared to the previous model (SHA-1)	4
3. General flow overview	5
3.1 Remittance of request to the Virtual POS.....	5
3.1.1 WS Input.....	5
3.1.2 Transactions Input.....	6
4. Host to Host request for payment message.....	7
4.1 Construction of the request data string	8
4.2 Identification of the signature algorithm version to be used	8
4.3 Identification of the key to be used for the signature	8
4.4 Signature of request details	9
4.5 Use of support libraries	9
4.5.1 PHP Library	9
4.5.2 JAVA Library.....	10
5. Host to Host request response	12
5.1 Response message signature	13
5.2 Use of support libraries	13
5.2.1 PHP Library	14
5.2.2 JAVA Library.....	14
6. Test environment.....	16
7. Error Codes	17
8. ANNEXES.....	20
8.1 Requests for payment (with remittance of card data).....	20
8.2 Confirmation/Refund Requests	22
8.3 Host to Host Response	23
8.4 Web Service request for payment - WDSL (<i>Web Services Description Language</i>)	25

1. Introduction

1.1 Purpose

This document includes the technical aspects necessary so that a merchant, which is currently operating in the SIS, may undertake the migration with the Virtual POS Service the new signature system based on the HMAC SHA256.

The current algorithm (SHA-1) in which the security of the connection of the shop with the virtual POS (and vice versa) is based is an obsolete algorithm according to security standards and could be subjected to attacks. So as to ensure greater security in connection with the Virtual POS SIS the preceding methods should be updated to the new signature system based on the HMAC SHA256.

This documentation applies to merchants which access the SIS via Host to Host, that is, merchants that access via the Web Service or through the "Operations" input.

So as to carry out the calculation of this new type of signature, the merchant may carry out per se using the standard functions or using the furnished libraries (PHP and JAVA).

2. **Summary of the differences of the new model (HMAC SHA-256) as compared to the previous model (SHA-1)**

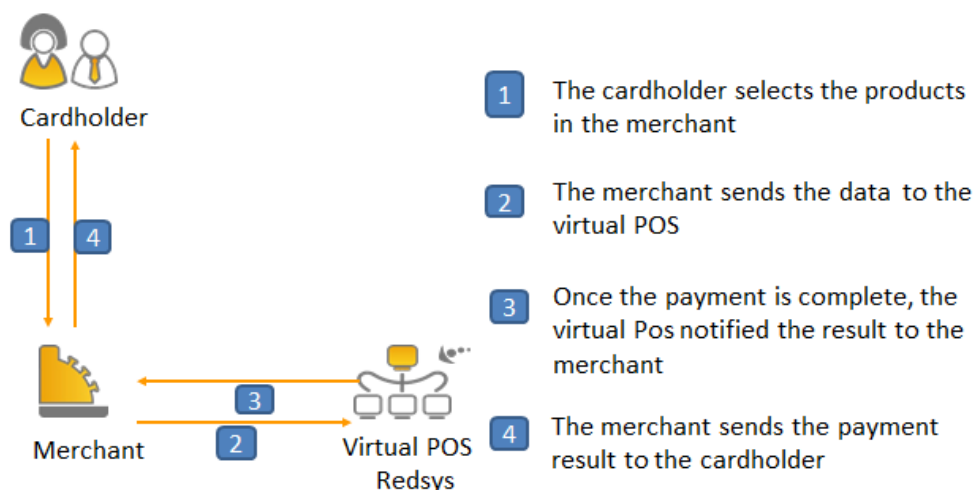
The differences of the new connection model based on the HMAC SHA-256 as compared to the previous model in use (based on SHA-1) which can be summarised in the following 3 points:

1. Format of the parameter sent.
 - **BEFORE:** A single parameter whose root element was <DATAINPUT> was sent. This root element contains all fields of the request for payment.
 - **NOW:** A single parameter whose root element was <REQUEST> is sent. The <DATAINPUT> element is part of the <REQUEST> root element, together with two new fields which will be specified in detail herein.
2. Calculation of the signature sent by the merchant.
 - **BEFORE:** The signature was calculated as a SHA-1 on the concatenation of the fields sent and was sent as a field of the <DATAINPUT> element. Any indicator parameter whatsoever of the version of the signature used is not sent.
 - **NOW:** The signature is calculated using a new key diversified per transaction, and a new algorithm (HMAC SHA256). The signature is sent in a new field which is part of the <REQUEST> root element. In addition a new field which specifies the version of the signature used which is part of the <REQUEST> root element is included.
3. Calculation of the response signature.
 - **BEFORE:** The signature was calculated as a SHA-1 on the concatenation of the fields received.
 - **NOW:** The signature is calculated using a new key diversified per transaction, and a new algorithm (HMAC SHA256) on the concatenation of the fields received.

NOTE: It is not necessary to modify any parameter whatsoever within the administration module of the merchant. The Virtual POS automatically accepts the connections based on the HMAC SHA-256, once the online store commences to send same to the Virtual POS. Similarly, the Virtual POS will use the HMAC SHA-256 format to confirm said these transactions to the shop server (notifications and navigation return of the customer).

3. General flow overview

The following diagram shows the general flow of a transaction performed via Host to Host.



3.1 Remittance of request to the Virtual POS

3.1.1 WS Input

As shown in step 2 of the previous scheme, the merchant must send to the Virtual POS the data of the payment request via the Web Service with the UTF-8 encoding. To this end, the Web Service has published several methods on which the Virtual POS operates. The **"trataPetición"** method enables the undertaking of transactions through the Web Service, for which a XML must be constructed which includes the request for payment data. The exact description of the XML request is submitted by means of the WSDL file in Annex 5 (Web Service request for payment - WSDL) of the Annexes section of the present document.

This request for payment must be sent to the following URLs, depending on whether or not a testing or actual transactions request is wished to be made:

URL Connection	Environment
https://sis-t.redsys.es:25443/sis/services/SerClsWSEntrada	Testing
https://sis.redsys.es/sis/services/SerClsWSEntrada	Real

After submitting the request to the Virtual POS the latter will interpret and carry out the necessary validations so as to, subsequently, process the transaction, as shown in Step 3 of the previous scheme. Depending on the result of the transaction, a XML request document is constructed with the result thereof having a UTF-8 encoding.

3.1.2 Transactions Input

As shown in step 2 of the above scheme, the merchant must send to the Virtual POS the data of the request for payment via the "Transactions" entry with the UTF-8 encoding. To this end, a form of a single parameter called "input" and whose value is the constructed XML which includes the request for payment data must be drawn up.

This request for payment must be sent to the following URLs, depending on whether or not a testing or actual transactions request is wished to be made:

URL Connection	Environment
https://sis-t.redsys.es:25443/sis/operaciones	Testing
https://sis.redsys.es/sis/operaciones	Real

After submitting the request to the Virtual POS the latter will interpret and carry out the necessary validations so as to, subsequently, process the transaction, as shown in Step 3 of the previous scheme. Depending on the result of the transaction, a XML request document is constructed with the result thereof having a UTF-8 encoding.

NOTE: In all aspects relating to the response of the Host to Host access is set out in Section 5.

4. Host to Host request for payment message

In order that the merchant may make the request through the Banco Sabadell Web Service it is necessary to exchange a series of data, both in the request messages as well as the response messages.

The message structure shall always be the same, establishing as the source thereof the **<REQUEST>** element. In its interior three elements must always be found which make reference to:

- Request for payment details. Element identified by the **<DATAINPUT>** label.
- Version of the signature algorithm. Element identified by the **<DS_SIGNATUREVERSION>** label.
- Signature of the request for payment details. Element identified by the **<DS_SIGNATURE>** label.

An example of a request for payment message is shown below:

```
<REQUEST>
  <DATOSENTRADA>
    <DS_MERCHANT_AMOUNT>145</DS_MERCHANT_AMOUNT>
    <DS_MERCHANT_ORDER>151029142229</DS_MERCHANT_ORDER>
    <DS_MERCHANT_MERCHANTCODE>327234688</DS_MERCHANT_MERCHANTCODE>
    <DS_MERCHANT_CURRENCY>978</DS_MERCHANT_CURRENCY>
    <DS_MERCHANT_PAN>4548812049400004</DS_MERCHANT_PAN>
    <DS_MERCHANT_EXPIRYDATE>1512</DS_MERCHANT_EXPIRYDATE>
    <DS_MERCHANT_CVV2>285</DS_MERCHANT_CVV2>
    <DS_MERCHANT_TRANSACTIONTYPE>A</DS_MERCHANT_TRANSACTIONTYPE>
    <DS_MERCHANT_TERMINAL>2</DS_MERCHANT_TERMINAL>
  </DATOSENTRADA>
  <DS_SIGNATUREVERSION>HMAC_SHA256_V1</DS_SIGNATUREVERSION>
  <DS_SIGNATURE>2YW19YQ8rb/0LLav79Y5L24Yw045KxN5hme27605WxY=</DS_SIGNATURE>
</REQUEST>
```

In order to facilitate the integration of a merchant, specified in detail below are the steps to be followed so as to construct the request for payment message.

4.1 Construction of the request data string

A string with all the request data in XML format obtaining as a result the **<DATAINPUT>** element must be constructed. It must be noted that it is no longer part of this element the **<DS_MERCHANT_MERCHANTSIGNATURE>** parameter.

It must be taken into account that there are several types of requests and according to the type, varies the structure of the message and the parameters which are sent and received.

Three types of requests can be distinguished:

- Requests for payment (with remittance of card data). In Annex 1 (Requests for Payment) of the Annexes section of the present document, the parameters required for this type of request is set out including an example.
- Confirmation/Refund Requests. In Annex 3 (Confirmation/Refund Requests) of the Annexes section of the present document, the parameters required for this type of request is set out including an example.

4.2 Identification of the signature algorithm version to be used

The particular version of the algorithm being used for the signature must be identified in the request. Currently the **HMAC_SHA256_V1** value is used to identify the version of all requests, reason why this will be the value of the **<DS_SIGNATUREVERSION>** element, as can be seen in the example of the message shown at the beginning of Section 4.

4.3 Identification of the key to be used for the signature

In order to calculate the signature it is necessary to use a specific key for each terminal. The key that must be used is the one received via SMS from Banco Sabadell.

IMPORTANT NOTE: This key must be stored on the server of the merchant in the safest way possible so as to avoid fraudulent use thereof. The merchant is responsible for the appropriate custody and preservation of secrecy of said key.

4.4 Signature of request details

Once the element is constructed with the request for payment details (<DATAINPUT>) and the specific key of the terminal should be calculated using the following steps:

1. A specific key per transaction is generated. In order to obtain the derived key to be used in a transaction a 3DES encryption between the merchant key, which must be previously decoded by BASE64, and the value of the order number of the transaction (DS_MERCHANT_ORDER) must be performed.
2. The HMAC SHA256 of the <DATAINPUT> element must be calculated.
3. The obtained result is encoded in the BASE 64 and the result of the encoding shall be the value of the **<DS_SIGNATURE>** element, as can be seen in the example of the form shown at the beginning of Section 4.

NOTE: The use of the support libraries furnished by Banco Sabadell for the generation of this field as is set out in Section 4.5.

4.5 Use of support libraries

The previous sections describe the manner of accessing the SIS using the Host to Host connection and the signature system based on the HMAC SHA256. Explained in this section is the method of how to use the libraries available in PHP and JAVA so as to facilitate the undertaking and generation of the signature. The use of the libraries furnished by Banco Sabadell is optional, but simplifies the activities to be undertaken by the merchant.

4.5.1 PHP Library

Specified below are the steps which must be followed by a merchant in order to use the PHP library furnished by Banco Sabadell:

1. Import the main library file, as is shown below:

```
include './apiRedsysWs.php';
```

The merchant must decide as to whether or not the import is to be used with the "include" or "required" function, as per the activities carried out.

2. To define the object of the main library file, as is shown below:

```
$miObj = new RedsysAPIWs;
```

3. Calculation of the **<DS_SIGNATURE>** element. So as to carry out the calculation of this parameter, the library function must be called "createMerchantSignatureHostToHost()" with the key obtained from the administration module and the element with the request for payment data (**<DATAINPUT>**), as is shown below:

```
$transactionData='<DATOSENTRADA><DS_MERCHANT_AMOUNT>'. $amount. '</DS_MERCHANT_AMOUNT><DS_MERCHANT_ORDER>'  
$key = 'sq7HjrUOBfKmC576ILgskD5srU870gJ7';  
$signature = $miObj->createMerchantSignatureHostToHost($key, $transactionData);
```

After obtaining the value of the **<DS_SIGNATURE>** element, the request for payment message can be completed and the call to the Host to Host made.

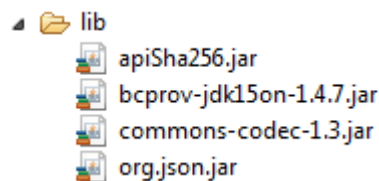
4.5.2 JAVA Library

Specified below are the steps which must be followed by a merchant in order to use the JAVA library furnished by Banco Sabadell:

1. Import the library, as is shown below:

```
<%@page import="sis.redsys.api.ApiWsMacSha256"%>
```

The merchant should include in the construction channel of the project all libraries (JARs) which are provided:



2. Calculation of the **<DS_SIGNATURE>** element. So as to carry out the calculation of this parameter, the library function must be called "createMerchantSignatureHostToHost()" with the key obtained from the administration module and the element with the request for payment data (**<DATAINPUT>**), as is shown below:

```
String transactionData="<DATOSENTRADA><DS_MERCHANT_AMOUNT>200</DS_MERCHANT_AMOUNT>..."  
String key = "sq7HjrUOBfKmC576ILgskD5srU870gJ7";  
String signature = ApiMacSha256.createMerchantSignatureHostToHost(key, transactionData);
```

After obtaining the value of the **<DS_SIGNATURE>** element, the request for payment message can be completed and the call to the Host to Host made.

5. Host to Host request response

In the present section the data which are part of the response message of a request via Host to Host is specified herein. This message is generated in the XML format and an example thereof is shown below:

```
<RETORNOXML>
  <CODIGO>0</CODIGO>
  <OPERACION>
    <Ds_Amount>145</Ds_Amount>
    <Ds_Currency>978</Ds_Currency>
    <Ds_Order>151029142229</Ds_Order>
    <Ds_Signature>
      MRvyhuDEpg4BmzfTdgHKrI5qQ9U5UD2Qe8eDadIZtyE=
    </Ds_Signature>
    <Ds_MerchantCode>327234688</Ds_MerchantCode>
    <Ds_Terminal>2</Ds_Terminal>
    <Ds_Response>0000</Ds_Response>
    <Ds_AuthorisationCode>185714</Ds_AuthorisationCode>
    <Ds_TransactionType>A</Ds_TransactionType>
    <Ds_SecurePayment>0</Ds_SecurePayment>
    <Ds_Language>1</Ds_Language>
    <Ds_MerchantData></Ds_MerchantData>
    <Ds_Card_Country>724</Ds_Card_Country>
  </OPERACION>
</RETORNOXML>
```

As can be seen in the example above, the response consists of two main elements:

- Code (**<CODE>**): Specifies whether or not the transaction was correct (does not specify if it has been authorised only if it has been processed). A 0 specifies that the transaction was correct. If it is not 0, it will have a code. (See error codes in Section 6 of this Guide)
- Details of the transaction (**<TRANSACTION>**): Compiles all necessary information regarding the transaction which has been performed. Via this element, it is determined whether or not the transaction has been authorised.

NOTE: The list of parameters which make up the response is described in Annex 4 (Host to Host Response) of the Annexes Section of the present document.

5.1 Response message signature

Once the response message and the specific key of the terminal has been obtained, whenever the transaction is authorised, the response signature must be verified using the following steps:

1. A specific key per transaction is generated. In order to obtain the derived key to be used in a transaction a 3DES encryption between the merchant key and the value of the order number of the transaction (DS_MERCHANT_ORDER) must be performed.
2. The HMAC SHA256 of the string made up by the concatenation of the value of the following fields is calculated:

String = Ds_Amount + Ds_Order + Ds_MerchantCode + Ds_Currency + Ds_Response + Ds_TransactionType + Ds_SecurePayment

Taking the example of the response which is set out at the beginning of this section, the resulting string would be:

String = 1451510291422293272346889780000A0

3. The obtained result is encoded in the BASE 64 and the result of the encoding shall be the same as the value of the **<Ds_Signature>** parameter obtained in the response.

NOTE: The use of the support libraries furnished by Banco Sabadell for the generation of this parameter is set out in Section 5.2.

5.2 Use of support libraries

This section explains how the libraries available in PHP and JAVA in order to facilitate the developments and generation of the response signature are used. The use of the libraries furnished by Banco Sabadell is optional, but simplifies the activities to be undertaken by the merchant.

5.2.1 PHP Library

Specified below are the steps which must be followed by a merchant in order to use the PHP library furnished by Banco Sabadell:

1. Import the main library file, as is shown below:

```
include './apiRedsysWs.php';
```

The merchant must decide as to whether or not the import is to be used with the "include" or "required" function, as per the activities carried out.

2. To define the object of the main library file, as is shown below:

```
$miObj = new RedsysAPI;
```

3. Calculation of the **<Ds_Signature>** parameter. So as to carry out the calculation of this parameter, the library function must be called "createSignatureResponseHostToHost()" with the key obtained from the administration module, the string that is wished to be signed (concatenation of fields described in Paragraph 2 of Section 5.1 of the present document) and the order number.

```
$concatenatedString = "145151029142229327234688978000A0";  
$order = "151029142229";  
$key = 'sq7HjrUOBfKmC576ILgskD5srU870gJ7';  
$signature = $miObj->createMerchantSignatureResponseHostToHost($key,  
                                                                $concatenatedString,  
                                                                $order)
```

The result obtained should be the same as the value of the **<Ds_Signature>** parameter obtained in the response.

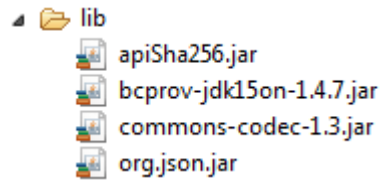
5.2.2 JAVA Library

Specified below are the steps which must be followed by a merchant in order to use the JAVA library furnished by Banco Sabadell:

1. Import the library, as is shown below:

```
<%@page import="sis.redsys.api.ApiWsMacSha256"%>
```

The merchant should include in the construction channel of the project all libraries (JARs) which are provided:



2. Calculation of the **<Ds_Signature>** parameter. So as to carry out the calculation of this parameter, the library function must be called "createSignatureResponseHostToHost()" with the key obtained from the administration module, the string that is wished to be signed (concatenation of fields described in Paragraph 2 of Section 5.1 of the present document) and the order number.

```
String concatenatedString = "1451510291422293272346889780000A0";  
String order = "1451510291422293272346889780000A0";  
String key = "sq7HjrU0BfKmc576ILgskD5srU870gJ7";  
String signature = ApiMacSha256.createMerchantSignatureResponseHostToHost(key,  
concatenatedString,  
order);
```

The result obtained should be the same as the value of the **<Ds_Signature>** parameter obtained in the response.

6. Test environment

Test environment allows to make the needed tests to verify the right behavior of the system before running payments on Real Environment. This Test environment is the same as the Real Environment but payments have no accounting validity.

The encryption key provided for test environment is common to all terminals which operation test environment with B. Sabadell. You can also use your merchant's test environment to do the test payments that you consider. In case that you do not have the configuration data, please get in touch with our IT Service on the 902 365 650 phone or send an email to: tpvvirtual@bancsabadell.com

Those are the parameters for test environment which are described as follow:

1. URL to send payment orders:

Web Service Gateway:

<https://sis-t.redsys.es:25443/sis/services/SerClsWSEntrada>

2. Merchant ID (Ds_Merchant_MerchantCode): 327234688

3. Encryption key: sq7HjrUOBfKmC576ILgskD5srU870gJ7

4. Terminal (Ds_Merchant_Terminal):

- Terminal 002 – Currency EUROS (Ds_MerchantCurrency = "978") and Non 3D-Secure (allows payments without authentication).

5. Accepted test card:

- CardNumber: 4548 8120 4940 0004
- Expiration date 12/17
- CVV2 (Card Verification Value): 123.

For 3D secure payments, just in case that the authentication of the card holder is needed, the personal identification number (PIN) is: 123456.

6. Administration Module URL:

<https://sis-t.redsys.es:25443/canales/bsabadell>

7. Administration Module Access:

- Terminal 2 (Non 3DSecure):
 - User: 327234688-002
 - Password: 123456a

7. Error Codes

This section sets out a glossary of errors which may occur in the integration process.

SIS Glossary of Errors

ERROR	DESCRIPTION	MESSAGE (ANNEX VI)
SIS0007	Error when deconstructing the input XML or errors occurring upon access via a former signature system having configured the HMAC SHA256 type key	MSG0008
SIS0008	Error Ds_Merchant_MerchantCode missing	MSG0008
SIS0009	Error format in Ds_Merchant_MerchantCode	MSG0008
SIS0010	Error Ds_Merchant_Terminal missing	MSG0008
SIS0011	Error format in Ds_Merchant_Terminal	MSG0008
SIS0014	Error format in Ds_Merchant_Order	MSG0008
SIS0015	Error Ds_Merchant_Currency missing	MSG0008
SIS0016	Error format in Ds_Merchant_Currency	MSG0008
SIS0017	Error transactions in pesetas are not accepted	MSG0008
SIS0018	Error Ds_Merchant_Amount missing	MSG0008
SIS0019	Error format in Ds_Merchant_Amount	MSG0008
SIS0020	Error Ds_Merchant_MerchantSignature missing	MSG0008
SIS0021	Error the Ds_Merchant_MerchantSignature is empty	MSG0008
SIS0022	Error format in Ds_Merchant_TransactionType	MSG0008
SIS0023	Error unknown Ds_Merchant_TransactionType	MSG0008
SIS0024	Error Ds_Merchant_ConsumerLanguage has more than 3 positions	MSG0008
SIS0025	Error format in Ds_Merchant_ConsumerLanguage	MSG0008
SIS0026	Error Nonexistent merchant/terminal sent	MSG0008
SIS0027	Error Currency sent by the merchants is different to that which is assigned to that terminal	MSG0008
SIS0028	Error Merchant/terminal is deregistered	MSG0008
SIS0030	Error in a payment using a card which has a transaction type which is neither payment nor preauthorisation	MSG0000
SIS0031	Method of payment undefined	MSG0000
SIS0033	Error in a payment using a mobile telephone which has a transaction type which is neither payment nor preauthorisation	MSG0000
SIS0034	Error accessing database	MSG0000
SIS0037	Invalid telephone number	MSG0000
SIS0038	Java error	MSG0000
SIS0040	Error the merchant/terminal has no assigned method of payment	MSG0008
SIS0041	Error calculating the HASH of the merchant data.	MSG0008
SIS0042	The sent signature is incorrect	MSG0008
SIS0043	Error when performing online notification	MSG0008
SIS0046	The bin of the card is not registered	MSG0002
SIS0051	Error repeated order number	MSG0001
SIS0054	Error there is no transaction on which to perform the refund	MSG0008
SIS0055	Error there is more than one payment with the same order number	MSG0008
SIS0056	The transaction on which it is wished to be refunded is not authorised	MSG0008
SIS0057	The amount to be refunded exceeds the allowed amount	MSG0008
SIS0058	Inconsistency of data, in the validation of a confirmation	MSG0008
SIS0059	Error there is no transaction on which to perform the confirmation	MSG0008
SIS0060	There is already a confirmation associated to the preauthorisation	MSG0008
SIS0061	The preauthorisation on which it is wished to confirm is not authorised	MSG0008
SIS0062	The amount to be confirmed exceeds the allowed amount	MSG0008
SIS0063	Error.Card number not available	MSG0008
SIS0064	Error.The card number cannot have more than 19 positions	MSG0008
SIS0065	Error.Non numerical card number	MSG0008

ERROR	DESCRIPTION	MESSAGE (ANNEX VI)
SIS0066	Error.Expiry month is not available	MSG0008
SIS0067	Error.Non numerical expiry month	MSG0008
SIS0068	Error.Invalid expiry month	MSG0008
SIS0069	Error.Expiry year is not available	MSG0008
SIS0070	Error.Non numerical expiry year	MSG0008
SIS0071	Expired card	MSG0000
SIS0072	Non voidable transaction	MSG0000
SIS0074	Error Ds_Merchant_Order missing	MSG0008
SIS0075	Error the Ds_Merchant_Order has less than 4 positions or more than 12	MSG0008
SIS0076	Error the Ds_Merchant_Order does not have the first four numerical positions	MSG0008
SIS0077	Error the Ds_Merchant_Order does not have the first four numerical positions.Is not used	MSG0000
SIS0078	Method of payment unavailable	MSG0005
SIS0079	Error when making a payment using a card	MSG0000
SIS0081	New session, stored data has been lost	MSG0007
SIS0084	The Ds_Merchant_Conciliation value is void	MSG0008
SIS0085	The Ds_Merchant_Conciliation value is not numerical	MSG0008
SIS0086	The Ds_Merchant_Conciliation value does not occupy 6 positions	MSG0008
SIS0089	The Ds_Merchant_ExpiryDate value does not occupy 4 positions	MSG0008
SIS0092	The Ds_Merchant_ExpiryDate value is void	MSG0008
SIS0093	Card not found in the ranges table	MSG0006
SIS0094	Card was not authenticated as 3D Secure	MSG0004
SIS0097	Value of the Ds_Merchant_CComercio field is invalid	MSG0008
SIS0098	Value of the Ds_Merchant_CVentana field is invalid	MSG0008
SIS0112	Error The specified type of transaction Ds_Merchant_Transaction_Type is not allowed	MSG0008
SIS0113	Exception produced in the transactions servlet	MSG0008
SIS0114	Error, has been called using a GET instead of a POST	MSG0000
SIS0115	Error there is no transaction on which to perform the payment of the instalment	MSG0008
SIS0116	The transaction on which it is wished to pay an instalment is not a valid transaction	MSG0008
SIS0117	The transaction on which it is wished to pay an instalment is not authorised	MSG0008
SIS0118	Has exceeded the total amount of the instalments	MSG0008
SIS0119	Value of the Ds_Merchant_DateFrequency field is invalid	MSG0008
SIS0120	Value of the Ds_Merchant_ChargeExpiryDate field is invalid	MSG0008
SIS0121	Value of the Ds_Merchant_SumTotal field is invalid	MSG0008
SIS0122	Value of the Ds_Merchant_DateFrequency field or not Ds_Merchant_SumTotal has an incorrect format	MSG0008
SIS0123	Has exceeded the deadline for making transactions	MSG0008
SIS0124	The minimum frequency in next recurring payment has not lapsed	MSG0008
SIS0132	The Authorisation Confirmation date may not exceed by more than 7 days to that of the Preauthorisation.	MSG0008
SIS0133	The Authentication Confirmation date may not exceed by more than 45 days to that of the Previous Authentication.	MSG0008
SIS0139	Error the initial recurring payment is duplicated	MSG0008
SIS0142	Deadline for payment exceeded	MSG0000
SIS0197	Error upon obtaining the data from the shopping basket in a gateway transaction type	MSG0000
SIS0198	Error the amount exceeds the limit permitted for the merchant	MSG0000
SIS0199	Error the number of transactions exceeds the limit permitted for the merchant	MSG0008
SIS0200	Error the accumulated amount exceeds the limit permitted for the merchant	MSG0008
SIS0214	The merchant does not accept refunds	MSG0008
SIS0216	Error Ds_Merchant_CVV2 has more than 3 positions	MSG0008
SIS0217	Error format in Ds_Merchant_CVV2	MSG0008
SIS0218	The merchant does not accept secure transactions for the input/transactions	MSG0008

ERROR	DESCRIPTION	MESSAGE (ANNEX VI)
SIS0219	Error the number of card transactions exceeds the limit permitted for the merchant	MSG0008
SIS0220	Error the accumulated card amount exceeds the limit permitted for the merchant	MSG0008
SIS0221	Error the CVV2 is obligatory	MSG0008
SIS0222	There is already a cancellation associated to the preauthorisation	MSG0008
SIS0223	The preauthorisation which is wished to be cancelled is not authorised	MSG0008
SIS0224	The merchant does not accept cancellations for not having an enlarged signature	MSG0008
SIS0225	Error there is no transaction on which to perform the cancellation	MSG0008
SIS0226	Inconsistency of data, in the validation of a cancellation	MSG0008
SIS0227	Value of the Ds_Merchant_TransactionDate field is invalid	MSG0008
SIS0229	There is no deferred payment code requested	MSG0008
SIS0252	The merchant does not accept the remittance of the card	MSG0008
SIS0253	The card fails the check-digit	MSG0006
SIS0254	Error the number of card transactions of the IP exceeds the limit permitted for the merchant	MSG0008
SIS0255	Error the accumulated amount of the IP exceeds the limit permitted for the merchant	MSG0008
SIS0256	The merchant cannot perform preauthorisations	MSG0008
SIS0257	This card does not allow preauthorisation transactions	MSG0008
SIS0258	Inconsistency of data, in the validation of a confirmation	MSG0008
SIS0261	Transaction retained due to exceeding the restrictions control in the SIS	MSG0008
SIS0270	The merchant cannot perform deferred preauthorisations	MSG0008
SIS0274	Unknown transaction type or not permitted by this entry to the SIS	MSG0008
SIS0429	Error in the version sent by the merchant in the Ds_SignatureVersion parameter	MSG0008
SIS0432	FUC error of the merchant incorrect	MSG0008
SIS0433	Terminal error of the merchant incorrect	MSG0008
SIS0434	Error no order number on the transaction sent by the merchant	MSG0008
SIS0435	Error in the calculation of the signature	MSG0008
SIS0436	Error in the construction of the parent element <REQUEST>	MSG0008
SIS0437	Error in the construction of the <DS_SIGNATUREVERSION> element	MSG0008
SIS0438	Error in the construction of the <DATAINPUT> element	MSG0008
SIS0439	Error in the construction of the <DS_SIGNATURE> element	MSG0008

8. ANNEXES

8.1 Requests for payment (with remittance of card data)

In the present annex the necessary data and its characteristics are described herein, so as to send a Host to Host request in the XML format. Likewise an example of how to use that data in the request for payment messages is included.

Data name	Leng./ Type	Description
<i>DS_MERCHANT_AMOUNT</i>	12 / N	Mandatory. The last two positions are considered decimals, except for Yens which do not have decimals.
<i>DS_MERCHANT_ORDER</i>	12 / A-N	Mandatory. Order number. The first 4 digits must be numeric. Each order is a one-off, it cannot be repeated.
<i>DS_MERCHANT_MERCHANTCODE</i>	9 / N	Mandatory. FUC Code assigned to the merchant.
<i>DS_MERCHANT_TERMINAL</i>	3 / N	Mandatory. Terminal number which is assigned by its bank. The default value "001". 3 is considered its maximum length.
<i>DS_MERCHANT_CURRENCY</i>	4 / N	Mandatory. Merchant currency. Which must be taken out under contract for the Terminal. Value 978 for Euros, 840 for Dollars, 826 for Pounds Sterling and 392 for Yen.
<i>DS_MERCHANT_PAN</i>	19 / N	Mandatory. Card.Its length depends on the type of card.
<i>DS_MERCHANT_EXPIRYDATE</i>	4 / N	Mandatory. Card expiry date. Its format is YYMM, YY being the last two digits of the year and MM the two digits of the month.
<i>DS_MERCHANT_CVV2</i>	3-4 / N	Mandatory. CVV2 code of the card.
<i>DS_MERCHANT_TRANSACTIONTYPE</i>	1 / A-N	Mandatory. Field for the merchant in order to indicate what type of transaction it is. .The possible values are: A – Traditional payment 1 – Preauthorisation O – Deferred Authorisation
Type A: ASCII characters from 65 = A to 90 = Z and that of 97 = a to 122 = z . Type N: ASCII characters from 30 = 0 to 39 = 9 .		

An example of a request for payment message is shown below:

```
<DATAINPUT>
  <DS_MERCHANT_AMOUNT>145</DS_MERCHANT_AMOUNT>
  <DS_MERCHANT_ORDER>050911523002</DS_MERCHANT_ORDER>
  <DS_MERCHANT_MERCHANTCODE>999008881</DS_MERCHANT_MERCHANTCO
  DE>
  <DS_MERCHANT_CURRENCY>978</DS_MERCHANT_CURRENCY>
  <DS_MERCHANT_PAN>XXXXXXXXXXXX</DS_MERCHANT_PAN>
  <DS_MERCHANT_CVV2>XXX</DS_MERCHANT_CVV2>
  <DS_MERCHANT_TRANSACTIONTYPE>A</DS_MERCHANT_TRANSACTIONTYPE>
  <DS_MERCHANT_TERMINAL>999</DS_MERCHANT_TERMINAL>
  <DS_MERCHANT_EXPIRYDATE>XXXX</DS_MERCHANT_EXPIRYDATE>
</DATAINPUT>
```

8.2 Confirmation/Refund Requests

In the present annex the necessary data and its characteristics are described herein, so as to send a Host to Host request in the XML format. Likewise an example of how to use that data in the request for payment messages is included.

Data name	Leng./ Type	Description
<i>DS_MERCHANT_AMOUNT</i>	12 / N	Mandatory. The last two positions are considered decimals, except for Yens which do not have decimals.
<i>DS_MERCHANT_ORDER</i>	12 / A-N	Mandatory. Order number.The first 4 digits must be numeric.Each order is a one-off, it cannot be repeated.
<i>DS_MERCHANT_MERCHANTCODE</i>	9 / N	Mandatory. FUC Code assigned to the merchant.
<i>DS_MERCHANT_TERMINAL</i>	3 / N	Mandatory. Terminal number which is assigned by its bank.The default value "001". 3 is considered its maximum length.
<i>DS_MERCHANT_CURRENCY</i>	4 / N	Mandatory. Merchant currency.Which must be taken out under contract for the Terminal. Value 978 for Euros, 840 for Dollars, 826 for Pounds Sterling and 392 for Yen.
<i>DS_MERCHANT_TRANSACTIONTYPE</i>	1 / A-N	Mandatory. Field for the merchant in order to indicate what type of transaction it is.The possible values are: 2 – Confirmation 3 – Automatic Refund 6 – Next Transaction 9 – Cancellation of Preauthorisation P - Deferred authorisation confirmation Q - Deferred authorisation cancellation S – Deferred next recurring authorisation
<i>DS_MERCHANT_AUTHORIZATIONCODE</i>	6 / Num	Optional. It represents the authorisation code necessary so as to identify a next recurring transaction in the refunds of the next recurring transactions.Mandatory for recurring transactions refunds.
Type A: ASCII characters from 65 = A to 90 = Z and that of 97 = a to 122 = z . Type N: ASCII characters from 30 = 0 to 39 = 9 .		

An example of a recurring payment request message is shown below:

```
<DATAINPUT>
  <DS_MERCHANT_AMOUNT>145</DS_MERCHANT_AMOUNT>
  <DS_MERCHANT_ORDER>050911523002</DS_MERCHANT_ORDER>
  <DS_MERCHANT_MERCHANTCODE>999008881</DS_MERCHANT_MERCHANTCODE>
  <DS_MERCHANT_CURRENCY>978</DS_MERCHANT_CURRENCY>
  <DS_MERCHANT_TRANSACTIONTYPE>3</DS_MERCHANT_TRANSACTIONTYPE>
  <DS_MERCHANT_TERMINAL>999</DS_MERCHANT_TERMINAL>
</DATAINPUT>
```

8.3 Host to Host Response

Set out below is a table which contains all the parameters which make up the Host to Host response.

Data name	Leng./ Type	Description
<i>CODE</i>		Mandatory. Specifies whether or not the transaction was correct or not (does not specify if it has been authorised only if it has been processed). A 0 specifies that the transaction was correct. If it is not 0, it will have a code. (See error codes in Section 6 of this Guide)
<i>Ds_Amount</i>	12 / A-N	Mandatory. For Euros the last two positions are considered decimals, except for yens which do not have decimals.
<i>Ds_Currency</i>	4 / N	Mandatory. Merchant currency.
<i>Ds_Order</i>	12 / A-N	Mandatory. Order number.
<i>Ds_Signature</i>	40 / A-N	Mandatory. Merchant signature.
<i>Ds_MerchantCode</i>	9 / N	Mandatory. FUC Code associated to the merchant.
<i>Ds_Terminal</i>	3 / N	Mandatory. Merchant terminal number.
<i>Ds_Response</i>	4 / N	Mandatory. Value specifying the result of the transaction. Will specify as to whether or not it has been authorised. The possible values of this field are described in the following table:
<i>Ds_AuthorisationCode</i>	6 / N	Optional. Authorisation code if any available for the authorised transactions.
<i>Ds_TransactionType</i>	1 / A-N	Mandatory. Specifies what type of transaction has been performed. The possible values are: A – Traditional payment 1 – Preauthorisation 2 – Confirmation 3 – Automatic Refund 5 – Recurring Transaction 6 – Next Transaction 9 – Cancellation of Preauthorisation O – Deferred Authorisation P - Deferred authorisation confirmation Q - Deferred authorisation cancellation R – Initial deferred recurring transaction S – Deferred next recurring authorisation
<i>Ds_SecurePayment</i>		Mandatory. Specifies whether or not the payment is secure: <ul style="list-style-type: none"> 0: secure (not applicable) 1: not secure.
<i>Ds_Language</i>	1 / N	Mandatory. Language.


Type A: ASCII characters from 65 = **A** to 90 = **Z** and that of 97 = **a** to 122 = **z**.
Type N: ASCII characters from 30 = **0** to 39 = **9**.

These are the possible values of the Ds_Response or "Response Code":

CODE	MEANING
0000 - 0099	Authorised transaction for payments and preauthorisations
900	Authorised transaction for refunds and confirmations
101	Expired card
102	Card in transitional exception or under suspected fraud
106	PIN attempts exceeded
125	Ineffective Card
129	Security code (CVV2/CVC2) incorrect
180	Card unrelated to the service
184	Error in cardholder authentication
190	Denial of the issuer without specifying reason
191	Erroneous expiry date
202	Card in transitional exception or under suspected fraud with card withdrawal
904	Merchant not registered in FUC
909	System error
913	Repeated order
944	Incorrect Session
950	Refund transaction not permitted
9912/912	Issuer not available
9064	Incorrect card number of positions
9078	Transaction type not allowed for that card
9093	Nonexistent card
9094	International servers rejection
9104	Merchant with "insurance holder" and cardholder without secure purchase key
9218	The merchant does not accept secure transactions per input/transactions
9253	Card fails check-digit
9256	The merchant cannot perform preauthorisations
9257	This card does not allow preauthorisation operations
9261	Transaction retained due to exceeding the restrictions control in the SIS
9913	Error in the confirmation which the merchant sends to the Virtual POS (only applicable in the SOAP synchronisation option)
9914	"KO" confirmation of the merchant (only applicable in the SOAP synchronisation option)
9915	At the request of the user the payment has been cancelled
9928	Cancellation of deferred authorisation performed by the SIS (batch process)
9929	Cancellation of deferred authorisation performed by the merchant
9997	Is being processed in another transaction in the SIS with the same card
9998	Transaction in application process of the card data
9999	Transaction has been redirected to the issuer for authentication

These response codes, in addition to the inherent Host to Host response, shown in the "response code" field of the transactions inquiries, whenever the transaction is not authorised, as is shown in the following figure:

Page 2 of 101

Terminal	Date Time	Operation type No. Order	Result Authorisation No. o Response Code	Amount	Payment Type
1	01-10-15 01-10-2015 16:50:14	Traditional Authorisation 151001165012	Authorised 581954	1,00 EUR 2 /	T
1	01-10-15 01-10-2015 16:50:16	Traditional Authorisation 151001165015	Sin Finalizar 9997		

8.4 Web Service request for payment - WDSL (Web Services Description Language)

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definition targetNamespace="http://webservice.sis.sermepa.es"
xmlns:apachesoap="http://xml.apache.org/xml-soap"
xmlns:impl="http://webservice.sis.sermepa.es"
xmlns:intf="http://webservice.sis.sermepa.es"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/"
xmlns:wsdlsoap="http://schemas.xmlsoap.org/wsdl/soap/"
xmlns:xsd="http://www.w3.org/2001/XMLSchema">
<wsdl:types>
<schema elementFormDefault="qualified"
targetNamespace="http://webservice.sis.sermepa.es"
xmlns="http://www.w3.org/2001/XMLSchema" xmlns:apachesoap="http://xml.apache.org/xml-soap"
xmlns:impl="http://webservice.sis.sermepa.es"
xmlns:intf="http://webservice.sis.sermepa.es"
xmlns:wsdl="http://schemas.xmlsoap.org/wsdl/">
<element name="trataPeticion">
<complexType>
<sequence>
<element name="datoEntrada" nillable="true" type="xsd:string"/>
</sequence>
</complexType>
</element>
<element name="trataPeticionResponse">
<complexType>
<sequence>
<element name="trataPeticionReturn" nillable="true" type="xsd:string"/>
</sequence>
</complexType>
</element>
<element name="consultaDCC">
<complexType>
<sequence>
<element name="datoEntrada" nillable="true" type="xsd:string"/>
</sequence>
</complexType>
</element>
<element name="consultaDCCResponse">
<complexType>
<sequence>
<element name="consultaDCCReturn" nillable="true" type="xsd:string"/>
</sequence>
</complexType>
</element>
</schema>
</wsdl:types>
```

```

<wsdl:message name="consultaDCCRequest">
<wsdl:part element="intf:consultaDCC" name="parameters"/>
</wsdl:message>
<wsdl:message name="trataPeticionResponse">
<wsdl:part element="intf:trataPeticionResponse" name="parameters"/>
</wsdl:message>
<wsdl:message name="trataPeticionRequest">
<wsdl:part element="intf:trataPeticion" name="parameters"/>
</wsdl:message>
<wsdl:message name="consultaDCCResponse">
<wsdl:part element="intf:consultaDCCResponse" name="parameters"/>
</wsdl:message>
<wsdl:portType name="">
<wsdl:operation name="trataPeticion">
<wsdl:input message="intf:trataPeticionRequest" name="trataPeticionRequest"/>
<wsdl:output message="intf:trataPeticionResponse" name="trataPeticionResponse"/>
</wsdl:operation>
<wsdl:operation name="consultaDCC">
<wsdl:input message="intf:consultaDCCRequest" name="consultaDCCRequest"/>
<wsdl:output message="intf:consultaDCCResponse" name="consultaDCCResponse"/>
</wsdl:operation>
</wsdl:portType>
<wsdl:binding name="SerClsWSEntradaSoapBinding" type="intf:SerClsWSEntrada">
<wsdl:soap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
<wsdl:operation name="trataPeticion">
<wsdl:soap:operation soapAction="" />
<wsdl:input name="trataPeticionRequest">
<wsdl:soap:body use="literal"/>
</wsdl:input>
<wsdl:output name="trataPeticionResponse">
<wsdl:soap:body use="literal"/>
</wsdl:output>
</wsdl:operation>
<wsdl:operation name="consultaDCC">
<wsdl:soap:operation soapAction="" />
<wsdl:input name="consultaDCCRequest">
<wsdl:soap:body use="literal"/>
</wsdl:input>
<wsdl:output name="consultaDCCResponse">
<wsdl:soap:body use="literal"/>
</wsdl:output>
</wsdl:operation>
</wsdl:binding>
<wsdl:serviceName="SerClsWSEntradaService">
<wsdl:port binding="intf:SerClsWSEntradaSoapBinding" name="SerClsWSEntrada">
<wsdl:soap:address location="https://sis.redsys.es/sis/services/SerClsWSEntrada"/>
</wsdl:port>
</wsdl:service>
</wsdl:definitions>

```