VIRTUAL POS

Developer Installation Manual

Version 1.11

VERSION CONTROL

VERSION	DATE	AFFECTS	BRIEF DESCRIPTION OF THE CHANGE
1.0	**23/05/2016**	**ALL**	**Initial document**
1.1	**09/11/2016**	**ALL**	**Concept review and data update**
1.2	**05/12/2016**	**ALL**	**Concept review and data update. The manual is divided into two independent versions: developer and functional**
1.3	**26/01/2017**	**ALL**	**Concept review and data update of SDK libraries according to latest versions received from Redsys**
1.4	**06/02/2017**	**ALL**	**iOS and Android manuals are included for Virtual POS integration**
1.5	**03/04/2017**	**ALL**	**Formatting settings**
1.6	**05/10/2017**	**ALL**	**Information expanded on 1-click payment redirection**
1.7	**22/02/2017**	**ALL**	**Uganda schilling and Chilean pesos has no decimals**
1.8	**28/03/2017**	**ALL**	**Reorganisation and new fields added**
1.9	**26/06/2018**	**Transaction Type**	**Discontinued operatives: Recurring transactions 5 & 6 and Deferred Pre-authorizations and Confirmations (O,P,Q,R y S) and Traditional Payment**
1.10	**13/12/2018**	**Card Brand**	**Adding new field "DS-Card_Brand"**
1.11	**05/02/2019**	**ALL**	**Adding new option "Payment retries"**

CONTENTS

INTRODUCTION

This guide provides the technical information necessary for the merchant, or its IT services, to successfully install the Virtual BBVA POS system in its ecommerce operations. The data required to operate in both the test and production environment is provided.

1. INTEGRATION TYPES

With BBVA's Virtual POS system, different types of integration can be carried out. This always depends on the way in which the merchant is going to operate.

We differentiate between:

- **Integration by redirection**: The "RealizarPago" (Make Payment) input will be used. In this environment, the merchant makes the POS call and the customer's session goes from the merchant's environment to the bank's environment, where customers will enter their data to make the payment. The session may then return to the merchant.

- **Webservice Integration**: The "Operaciones" (Operations) input will be used. Through this system, and fulfilling a number of requirements, the data for payment can be collected in the merchant's own environment.

- **iOS, Android and library integration (JAVA, PHP, .Net)**. Libraries are provided to configure the payment gateway in different environments.

- **Integration with external solutions**: Integration of the payment gateway in merchants using third-party tools, such as Magento, Prestashop, etc.

2. TEST ENVIRONMENT

In order to be able to carry out the installation tests, during the process of registering your BBVA Virtual POS, the entity will provide you with access parameters to a TEST POS environment where, in an isolated environment, identical to the production environment, you will be able to carry out test operations. These sales will be fictitious and therefore not valid for accounting purposes.

Access to this environment requires access to ports 25443 and 26443.

The characteristics of the test environment are detailed below:

Payment URLs:

'RealizarPago" (Make Payment) input: https://sis-t.redsys.es:25443/sis/realizarPago

'Webservice' input: https://sis-t.redsys.es:25443/sis/services/SerClsWSEntrada

Merchant number (Ds_Merchant_MerchantCode): **XXXXXXXX**

Secret code (Ds_Merchant_MerchantSignature)

SHA-1: qwertyasdf0123456789
SHA-256: sq7HjrUOBfKmC576ILgskD5srU870gJ7

Other parameters

Terminal number (Ds_Merchant_Terminal = 001)
Operation currency code (Ds_MerchantCurrency = 978)

Test card

We provide you with a card that is enabled to operate exclusively in the test environment:

Card: 4548 8120 4940 0004
Expiry: 12/20
Security code: (CVV2) 285

Attention: This card is configured simulating that the cardholder needs to been authenticated with his/her bank. To do this, the Identification Code 123456 must be entered.

In addition, the URL for access to the administration module is as follows:

https://sis-t.redsys.es:25443/canales

3. INSTALLATION

The following are the different types of configuration to be used by the developer depending on the connection method with which the merchant will operate the BBVA Virtual POS system

3.1 REDIRECTION INPUT - "RealizarPago" (Make Payment)

The integration of the Virtual POS is carried out through a connection by means of a Redirection of the purchasing customer's browser.

This form of connection allows you to transfer the session from the end customer to the Virtual POS, so that the payment method selection and data entry are carried out in the secure environment of the Virtual POS server and outside the merchant's responsibility.

In addition to the ease of implementation for the merchant and peace of mind with regard to the liability of payment data, this connection mode enables the use of authentication mechanisms such as 3DSecure, where the card's bank directly asks the cardholder to provide secret data that makes the purchase more secure.

The merchant must send the data of the payment request encoded in UTF-8 to the Virtual POS through the cardholder's browser. To do this, you must prepare a form with the following fields:

- Ds_SignatureVersion: Constant indicating the signature version being used.
- Ds_MerchantParameters: Chain in JSON format with all parameters of the request coded in Base 64 and without carriage returns (Appendix 5.1 of this document includes the list of parameters that can be sent in a payment request).
- Ds_Signature: Signature of the data sent. This is the result of the HMAC SHA256 of the JSON string encoded in Base 64 sent in the previous parameter.

This form should be sent to the following URLs depending on whether you want to request testing or actual operations:

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

RECEIVING THE RESULT (ONLINE NOTIFICATION)

Once the transaction has been managed, the Virtual POS can inform the merchant of the result of the transaction by means of connecting directly to its server. This notification is optional and is configured for each terminal.

The online notification consists of an HTTP POST with the information on the result encoded in UTF-8. The following fields will be included in the POST:

- **Ds_SignatureVersion**: Constant indicating the signature version being used.
- **Ds_MerchantParameters**: Chain in JSON format with all parameters of the response coded in Base 64 and without carriage returns (Appendix 5.2 of this document includes the list of parameters that can be included in the online notification).
- **Ds_Signature**: Signature of the data sent. Result of the HMAC SHA256 of the JSON string encoded in Base 64 sent in the previous parameter. The merchant is responsible for validating the HMAC sent by the Virtual POS to ensure the validity of the response. This validation is necessary to ensure that the data has not been tampered with and that the origin is actually the Virtual POS.

SENDING REQUEST FORM

The merchant must put together a form with the payment request's parameters that must be sent to the Virtual POS through the customer's browser. There are two possibilities:

Without Sending Card Data:

```
<form name="from" action="https://sis-t.redsys.es:25443/sis/realizarPago"method="POST">
    <input type="hidden" name="Ds_SignatureVersion" value="HMAC_SHA256_V1"/>

    <input type="hidden" name="Ds_MerchantParameters" value="
    eyJEU19NRVJDSEFOVF9BTU9VTlQiOiI5OTkiLCJEU19NRVJDSEFOVF9PUkRFUiI6IjEyMzQ1Njc4OTAiLCJEU1
    9NRVJDSEFOVF9NRVJDSEFOVENPREUiOiI5OTkwMDg4ODEiLCJEU19NRVJDSEFOVF9DVVJSRU5DWSI6Ijk
    3OCIsIkRTX01FUkNIQU5UX1RSQU5TQUNUSU9OVFlQRSI6IjAiLCJEU19NRVJDSEFOVF9URVJNSU5BTCI6IjE
    iLCJEU19NRVJDSEFOVF9NRVJDSEFOVFVSTCI6Imh0dHA6XC9cL3d3dy5wcnVlYmEuY29tXC91cmxOb3RpZ
    mljYWNpb24ucGhwIiwiRFNfTUVSQ0hBTlRfVVJMT0siOiJodHRwOlwvXC93d3cucHJ1ZWJhLmNvbVwvdXJs
    T0sucGhwIiwiRFNfTUVSQ0hBTlRfVVJMS08iOiJodHRwOlwvXC93d3cucHJ1ZWJhLmNvbVwvdXJ
    sS08ucGhwIn0="/>

    <input type="hidden" name="Ds_Signature"
    value="PqV2+SF6asdasMjXasKJRTh3UIYya1hmU/igHkzhC+R="/>

</form>
```

Sending Card Data:

```
<form name="from" action="https://sis-t.redsys.es:25443/sis/realizarPago" method="POST">
    <input type="hidden" name="Ds_SignatureVersion" value="HMAC_SHA256_V1"/>
    <input type="hidden" name="Ds_MerchantParameters" value="
    eyJEU19NRVJDSEFOVF9BTU9VTlQiOiIxNDUiLCJEU19NRVJDSEFOVF9PUkRFUiI6IjE0NDYwNjg1ODEiLCJE
    U19NRVJDSEFOVF9NRVJDSEFOVENPREUiOiI5OTkwMDg4ODEiLCJEU19NRVJDSEFOVF9DVVJSRU5DWSI6
    Ijk3OCIsIkRTX01FUkNIQU5UX1RSQU5TQUNUSU9OVFlQRSI6IjAiLCJEU19NRVJDSEFOVF9URVJNSU5BTCI6
    IjEiLCJEU19NRVJDSEFOVF9NRVJDSEFOVFVSTCI6Imh0dHA6XC9cL3d3dy5wcnVlYmEuY29tXC91cmxOb3R
    pZmljYWNpb24ucGhwIiwiRFNfTUVSQ0hBTlRfVVJMT0siOiJodHRwOlwvXC93d3cucHJ1ZWJhLmNvbVwvd
    XJsT0sucGhwIiwiRFNfTUVSQ0hBTlRfVVJMS08iOiJodHRwOlwvXC93d3cucHJ1ZWJhLmNvbVwvdXJsS08uc
    GhwIiwiRFNfTUVSQ0hBTlRfUEFOIjoiNDU0ODgxMjA0OTQwMDAwNCIsIkRTX01FUkNIQU5UX0VYUElSW
    URBVEUiOiIxNTEyIiwiRFNfTUVSQ0hBTlRfQ1ZWMiI6IjEyMyJ9"/>
    <input type="hidden" name="Ds_Signature"
    value="PqV2+SF6asdasMjXasKJRTh3UIYya1hmU/igHkzhC+R="/>
</form>
```

Attention: If the 2nd option is chosen, the merchant must comply with the PCI-DSS security standard.

PUTTING TOGETHER DATA STREAM - Ds_MerchantParameters

A string with all of the request's data must be put together in JSON format. JSON is an open, text-based data exchange format. Just like XML it is designed to be readable and independent of the technology platform. Data encoding in JSON is very lightweight so it is ideal for data exchange in Web applications.

The name of each parameter must be indicated in capital letters or with "CamelCase" (For example: DS_MERCHANT_AMOUNT or Ds_Merchant_Amount).

Example without sending card data:

{"DS_MERCHANT_AMOUNT":"145","DS_MERCHANT_ORDER":"1446117555","DS_MERCHANT_MERCHANTCODE":"999008881","DS_MERCHANT_CURRENCY":"978","DS_MERCHANT_TRANSACTIONTYPE":"0","DS_MERCHANT_TERMINAL":"1","DS_MERCHANT_MERCHANTURL":"http://www.prueba.com/urlNotificacion.php","DS_MERCHANT_URLOK":"http://www.prueba.com/urlOK.php","DS_MERCHANT_URLKO":"http://www.prueba.com/urlKO.php"}

Example with sending card data:

{"DS_MERCHANT_AMOUNT":"145","DS_MERCHANT_ORDER":"1446068581","DS_MERCHANT_MERCHANTCODE":"999008881","DS_MERCHANT_CURRENCY":"978","DS_MERCHANT_TRANSACTIONTYPE":"0","DS_MERCHANT_TERMINAL":"1","DS_MERCHANT_MERCHANTURL":"http://www.prueba.com/urlNotificacion.php","DS_MERCHANT_URLOK":"http://www.prueba.com/urlOK.php","DS_MERCHANT_URLKO":"http://www.pruba.com/urlKO.php","DS_MERCHANT_PAN":"4548812049400004","DS_MERCHANT_EXPIRYDATE":"1512","DS_MERCHANT_CVV2":"123"}

Once the JSON string has been put together with all the fields, it is necessary to encode it in BASE64 without carriage returns to ensure that it remains constant and is not altered when going through the customer's/buyer's browser.

The JSON objects that have just been shown coded in BASE64 are shown below:

Example JSON encrypted without sending card data:

eyJEU19NRVJDSEFOVF9BTU9VTlQiOiI5OTkiLCJEU19NRVJDSEFOVF9PUkRFUiI6IjEyMzQ1Njc4OTAiLCJEU19NRVJDSEFOVF9NRVJDSEFOVENPREUiOiI5OTkwMDg4ODEiLCJEU19NRVJDSEFOVF9DVVJSRU5DWSI6Ijk3OCIsIkRTX01FUkNIQU5UX1RSQU5TQUNUSU9OVFlQRSI6IjAiLCJEU19NRVJDSEFOVF9URVJNSU5BTCI6IjEiLCJEU19NRVJDSEFOVF9NRVJDSEFOVFVSTCI6Imh0dHA6XC9cL3d3dy5wcnVlYmEuY29tXC91cmxOb3RpZmljYWNpb24ucGhw

IiwiRFNfTUVSQ0hBTlRfVVJMT0siOiJodHRwOlwvXC93d3cucHJ1ZWJhLmNvbVwvdXJsT0sucGhwIiwiRFNfTUVSQ0hBTlRfVVJMS08iOiJodHRwOlwvXC93d3cucHJ1ZWJhLmNvbVwvdXJ sS08ucGhwIn0

Example JSON encrypted with sending card data:

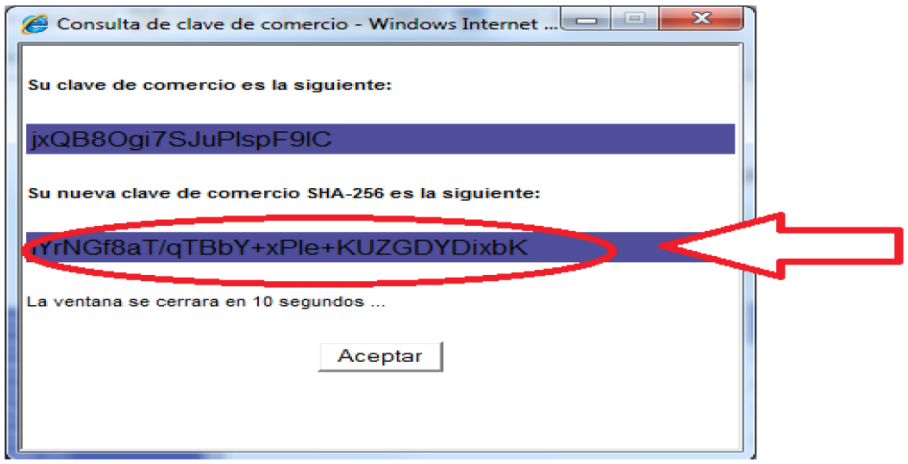
eyJEU19NRVJDSEFOVF9BTU9VTlQiOiIxNDUiLCJEU19NRVJDSEFOVF9PUkRFUiI6IjE0NDYwNjg1ODEiLCJEU19NRVJDSEFOVF9NRVJDSEFOVENPREUiOiI5OTkwMDg4ODEiLCJEU19NRVJDSEFOVF9DVVJSRU5DWSI6Ijk3OCIsIkRTX01FUkNIQU5UX1RSQU5TQUNUSU9OVFlQRSI6IjAiLCJEU19NRVJDSEFOVF9URVJNSU5BTCI6IjEiLCJEU19NRVJDSEFOVF9NRVJDSEFOVFVSTCI6Imh0dHA6XC9cL3d3dy5wcnVlYmEuY29tXC91cmxOb3RpZmljYWNpb24ucGhwIiwiRFNfTUVSQ0hBTlRfVVJMT0siOiJodHRwOlwvXC93d3cucHJ1ZWJhLmNvbVwvdXJsT0

sucGhwIiwiRFNfTUVSQ0hBTlRfVVJMS08iOiJodHRwOlwvXC93d3cucHJ1ZWJhLmNvbVwvdXJsS08ucGhwIiwiRFNfTUVSQ0hBTlRfUEFOIjoiNDU0ODgxMjA0OTQwMDAwNCIsIkRTX01FUkNIQU5UX0VYUElSWURBVEUiOiIxNTEyIiwiRFNfTUVSQ0hBTlRfQ1ZWMiI6IjEyMyJ9

The string resulting from the encoding in BASE64 will be the value of the parameter Ds_MerchantParameters (the list of parameters that can be sent in a payment request are included in Appendix 5.1 of this document).

SIGNING THE REQUEST'S DATA

To calculate the signature, a specific code must be used for each terminal. You can obtain the code by accessing the Administration Module, "Consulta datos de Comercio" (Query merchant's data) option, in the "Ver clave" (View code) section, as shown in the following image:



IMPORTANT NOTE: This code must be stored on the merchant's server in the securest way possible to avoid it being used fraudulently. The merchant is responsible for properly safeguarding this code and keeping it a secret.

Once the data chain to be signed and the terminal's specific code has been put together, the signature must be calculated following these steps:

1. An operation-specific key is generated. To obtain the derived code to be used in an operation, a 3DES encryption must be performed between the merchant's code, which must be previously decoded in BASE 64, and the value of the operation's order number (Ds_Merchant_Order).

2. The HMACSHA256 is calculated from the value of parameter Ds_MerchantParameters and the code obtained in the previous step.

3. The result obtained is encoded in BASE 64, and the result of the encoding will be the value of parameter Ds_Signature, as can be seen in the example of the form shown at the beginning of the section.

RECEIVING ONLINE NOTIFICATION

Once the transaction has been managed, the Virtual POS can inform the merchant's server of the result of the transaction by means of a direct connection to the

merchant's server. This notification is optional and must be configured for each terminal in the Administration Module.
The online notification consists of an HTTP POST with the information on the result encoded in UTF-8. The following fields will be included in the POST:

- **Ds_SignatureVersion**: Constant indicating the signature version being used.

- **Ds_MerchantParameters:** Chain in JSON format with all parameters of the response coded in Base 64 and without carriage returns (Appendix 2 of the Appendix section of this document includes the list of parameters that can be included in the online notification).

- **Ds_Signature**: Signature of the data sent. Result of the HMAC SHA256 of the JSON string encoded in Base 64 sent in the previous parameter. The merchant is responsible for validating the HMAC sent by the Virtual POS to ensure the validity of the response. This validation is necessary to ensure that the data has not been tampered with and that the origin is actually the Virtual POS.

NOTE: The Virtual POS sends the online notification to the URL reported by the merchant in parameter Ds_Merchant_MerchantURL.

3.2 WEB SERVICE INPUT

This form of connection allows merchants to have the Virtual POS integrated into their own Web application. This connection does not allow interaction with the cardholder and therefore, it is not possible to use the 3DSecure protocol (secure purchase) in the authorization process. If you use this connection to process authorizations, the card data must be sent in the payment form, so compliance with the PCI-DSS standard is mandatory.

This input allows you to carry out operations through the WebService, for which you must build an XML that includes the payment request data.

The exact description of this XML request is presented in the WSDL file in the Appendix to this document.

This payment request should be sent to the following URLs depending on whether you want to perform a testing request or real operations:

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

REQUEST FOR PAYMENT WEB SERVICE MESSAGE

In order for the merchant to be able to make the request through BBVA's WebService, a series of data must be exchanged, both in the request messages and in the response messages.

The structure of the message will always be the same, with the **<REQUEST>** element being established as the root of the message There are always three elements that refer to it:

- Details of the payment request. Element identified by the label **<DATOSENTRADA>**.
- Signature algorithm version. Element identified by the label **<DS_SIGNATUREVERSION>**.
- Signature of the payment request data. Element identified by the label **<DS_SIGNATURE>**.

The following is an example of a payment request message:

```
<REQUEST>
    <DATOSENTRADA>
        <DS_MERCHANT_AMOUNT>145</DS_MERCHANT_AMOUNT>
        <DS_MERCHANT_ORDER>1444904795</DS_MERCHANT_ORDER>
        <DS_MERCHANT_MERCHANTCODE>999008881</DS_MERCHANT_MERCHANTCODE>
        <DS_MERCHANT_CURRENCY>978</DS_MERCHANT_CURRENCY>
        <DS_MERCHANT_PAN>XXXXXXXXXXXXXXXX</DS_MERCHANT_PAN>
        <DS_MERCHANT_CVV2>XXX</DS_MERCHANT_CVV2>
        <DS_MERCHANT_TRANSACTIONTYPE>0</DS_MERCHANT_TRANSACTIONTYPE>
        <DS_MERCHANT_TERMINAL>871</DS_MERCHANT_TERMINAL>
        <DS_MERCHANT_EXPIRYDATE>XXXX</DS_MERCHANT_EXPIRYDATE>
    </DATOSENTRADA>
    <DS_SIGNATUREVERSION>HMAC_SHA256_V1</DS_SIGNATUREVERSION>
    <DS_SIGNATURE>
        VV3acxBgABrS5VYcLyJD1KqIsa2pPdvajPBG510lFfg=
    </DS_SIGNATURE>
</REQUEST>
```

A string with all of the request's data must be put together in XML format, resulting in the element **<DATOSENTRADA>**.

It should be noted that there are several types of requests and depending on the type, the message structure and the parameters that are sent and received vary.

We can differentiate between three types of requests:

- Payment requests (sending card data). In Appendix 5.4 of this

document, the parameters necessary for this type of request are presented, including an example.

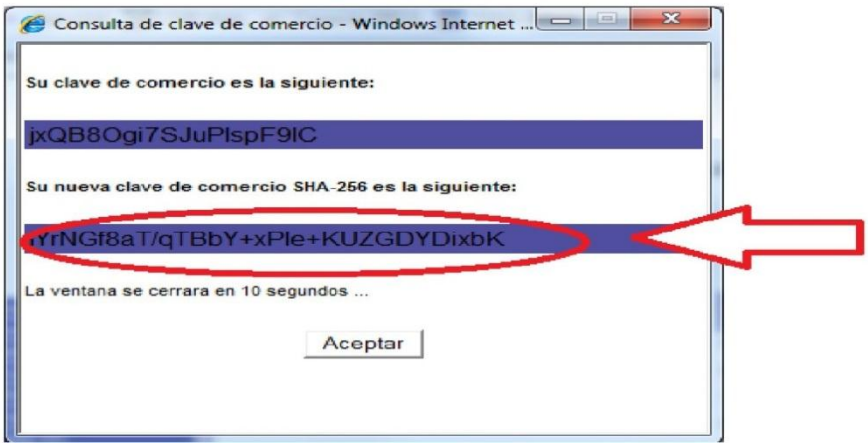
- Recurring payment requests (sending card data). In Appendix 5.5 (Recurring payment requests) of this document, the parameters necessary for this type of request are presented, including an example.
- Confirmation/Return Requests. In Appendix 5.6 of this document, the parameters necessary for this type of request are presented, including an example.

For merchants that use special operations such as 1-Click Payment, they must include the specific fields of this type of operation in the <DATOSENTRADA> element. In Appendix 5.6 of this document, the parameters necessary for this type of request are presented, including an example.

For transactions processed through Secured Terminals by Webservice, the parameter **DirectPayment=True** must be used, due there is no chance to authenticate the cardholder. These transactions will be Non-secured.

SIGNING THE REQUEST'S DATA

To calculate the signature, a specific code must be used for each terminal. You can obtain the code by accessing the Administration Module, "Consulta datos de Comercio" (Query merchant's data) option, in the "Ver clave" (View code) section, as shown in the following image:



IMPORTANT NOTE: This code must be stored on the merchant's server in the securest way possible to avoid it being used fraudulently. The merchant is responsible for properly safeguarding this code and keeping it a secret.

Once the element with the payment request data (<DATOSENTRADA>) and the terminal's specific code has been put together, the signature must be calculated following these steps:

1. An operation-specific key is generated. To obtain the derived code to be used in an operation, a 3DES encryption must be performed between the merchant's code, which must be previously decoded in BASE 64, and the value of the operation's order number (DS_MERCHANT_ORDER).

2. The HMAC SHA256 is calculated from the <DATAINPUT> element.

3. The result obtained is encoded in BASE 64, and the result of the encoding will be the value of the element <DS_SIGNATURE>, as can be seen in the example of the form shown at the beginning of the section.

WEB SERVICE REQUEST RESPONSE MESSAGE

The data that is part of the response message of a request to the Virtual TPV WebService is described below. This message is generated in XML format:

Example of payment response (merchant configured without sending card data):

```
<RETORNOXML>
  <CODIGO>0</CODIGO>
  <OPERACION>
      <Ds_Amount>145</Ds_Amount>
      <Ds_Currency>978</Ds_Currency>
      <Ds_Order>1444912789</Ds_Order>
      <Ds_Signature>
          bAuiQOymGvYzqHi7dEeuWrRYFeUjtFH6NyOoWSl0vHU=
      </Ds_Signature>
      <Ds_MerchantCode>999008881</Ds_MerchantCode>
      <Ds_Terminal>871</Ds_Terminal>
      <Ds_Response>0000</Ds_Response>
      <Ds_AuthorisationCode>050372</Ds_AuthorisationCode>
      <Ds_TransactionType>0</Ds_TransactionType>
      <Ds_SecurePayment>0</Ds_SecurePayment>
      <Ds_Language>1</Ds_Language>
      <Ds_Card_Type>D</Ds_Card_Type>
      <Ds_MerchantData></Ds_MerchantData>
      <Ds_Card_Country>724</Ds_Card_Country>
  </OPERACION>
</RETORNOXML>
```

Example of payment response (merchant configured with sending card data):

```
<RETORNOXML>
```

```
        <CODIGO>0</CODIGO>
        <OPERACION>
                <Ds_Amount>145</Ds_Amount>
                <Ds_Currency>978</Ds_Currency>
                <Ds_Order>1449821545</Ds_Order>
                <Ds_Signature>
                        6quLImPCOSTFpwhC7+ai1L+SPdKbcGx2sgC2A/1hwQo=
                </Ds_Signature>
                <Ds_MerchantCode>999008881</Ds_MerchantCode>
                <Ds_Terminal>871</Ds_Terminal>
                <Ds_Response>0000</Ds_Response>
                <Ds_AuthorisationCode>109761</Ds_AuthorisationCode>
                <Ds_TransactionType>0</Ds_TransactionType>
                <Ds_SecurePayment>0</Ds_SecurePayment>
                <Ds_Language>1</Ds_Language>
                <Ds_CardNumber>4548812049400004</Ds_CardNumber>
                <Ds_MerchantData></Ds_MerchantData>
                <Ds_Card_Country>724</Ds_Card_Country>
        </OPERACION>
</RETORNOXML>
```


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

- **(<CODIGO>)** Code: Indicates whether the operation has been successful or not, (does not indicate whether it has been authorized, only if it has been processed). An 0 indicates that the operation has been successful. If it is anything other than 0, it will have a code.

- Operation data **(<OPERACION>)** (Operation): Gathers all the necessary information about the operation that has been performed. This element determines whether the operation has been approved or not.

NOTE: The list of parameters that are part of the response is described in Appendix 5.8 of this document.

SIGNING THE RESPONSE MESSAGE

Once the response message and terminal-specific code have been obtained, provided that the operation is authorized, the signature of the response must be verified by following these steps:

1. An operation-specific key is generated. To obtain the derived code to be used in an operation, a 3DES encryption must be performed between the merchant's code, which must be previously decoded in BASE 64, and the value of the operation's order number (DS_ORDER).

2. The HMAC SHA256 is calculated from the string formed by combining the value of the following fields:

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

   If we take the response shown at the beginning of this section as an example, the resulting string would be:

   String = 145144491278999900888197800000O

   If the merchant has send card in response configured, the HMAC SHA256 must be calculated from the string formed by combining the value of the following fields:

   Cadena = Ds_Amount + Ds_Order + Ds_MerchantCode + Ds_Currency + Ds_Response + Ds_CardNumber + Ds_TransactionType + Ds_SecurePayment

   If we take the response shown at the beginning of this section as an example, the resulting string would be:

   String = 145144982154599900888197800004548812049400004O0

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

3.3 USE OF LIBRARIES - SDK

There are libraries available in PHP, JAVA and .NET to facilitate the development and generation of payment form fields.

Using the libraries supplied by BBVA is optional, although they simplify the developments to be carried out by the merchant.

SENDING OPERATIONS

PHP Library

Steps to follow

1. Import the main library file as shown below:

```
include_once 'redsysHMAC256_API_PHP_4.0.2/apiRedsys.php';
```

The merchant must decide whether it wants to do the import with the "include" or "required" function, as per developments.

2. Define an object of the main class in the library, as shown below:

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

3. Calculate the parameter **Ds_MerchantParameters**. To calculate this parameter, you must initially add all the parameters of the payment request you want to send, as shown below:

```
$miObj->setParameter("DS_MERCHANT_AMOUNT",$amount);
$miObj->setParameter("DS_MERCHANT_ORDER",$id);
$miObj->setParameter("DS_MERCHANT_MERCHANTCODE",$fuc);
$miObj->setParameter("DS_MERCHANT_CURRENCY",$moneda);
$miObj->setParameter("DS_MERCHANT_TRANSACTIONTYPE",$trans);
$miObj->setParameter("DS_MERCHANT_TERMINAL",$terminal);
$miObj->setParameter("DS_MERCHANT_MERCHANTURL",$url);
$miObj->setParameter("DS_MERCHANT_URLOK",$urlOK);
$miObj->setParameter("DS_MERCHANT_URLKO",$urlKO);
```

Finally, to calculate the Ds_MerchantParameters parameter, you must call up the library function "createMerchantParameters()", as shown below:

```
$params = $miObj->createMerchantParameters();
```

4. Calculate the parameter Ds_Signature. To calculate this parameter, you must call up the library function "createMerchantSignature()" with the code obtained from the administration module, as shown below:

```
$claveModuloAdmin = 'Mk9m98IfEblmPfrpsawt7BmxObt98Jev';
$signature = $miObj->createMerchantSignature($claveModuloAdmin);
```

5. Once the values of parameters Ds_MerchantParameters and Ds_Signature have been obtained, the payment form must be filled in with these values, as shown below:

```
<form action="https://sis.redsys.es/sis/realizarPago"
    method="POST" target="_blank">

    <input type="text" name="Ds_SignatureVersion"
        value="HMAC_SHA256_V1"/>
    <input type="text" name="Ds_MerchantParameters"
        value="<?php echo $params; ?>"/>
    <input type="text" name="Ds_Signature"
        value="<?php echo $signature; ?>"/>
    <input type="submit" value="Realizar Pago"/>

</form>
```

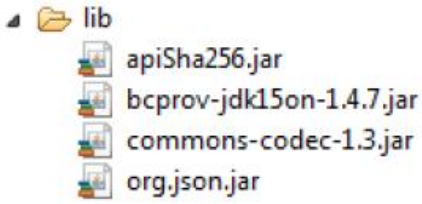
JAVA Library

Steps to follow

1. Import the library as shown below:

```
<%@page import="sis.redsys.api.ApiMacSha256"%>
```

The merchant must include under the project's construction all the libraries (JARs) provided:



2. Define an object of the main class in the library, as shown below:

```
ApiMacSha256 apiMacSha256 = new ApiMacSha256();
```

3. Calculate the parameter Ds_MerchantParameters. To calculate this parameter, you must initially add all the parameters of the payment request you want to send, as shown below:

```
apiMacSha256.setParameter("DS_MERCHANT_AMOUNT", amount);
apiMacSha256.setParameter("DS_MERCHANT_ORDER", id);
apiMacSha256.setParameter("DS_MERCHANT_MERCHANTCODE", fuc);
apiMacSha256.setParameter("DS_MERCHANT_CURRENCY", moneda);
apiMacSha256.setParameter("DS_MERCHANT_TRANSACTIONTYPE", trans);
apiMacSha256.setParameter("DS_MERCHANT_TERMINAL", terminal);
apiMacSha256.setParameter("DS_MERCHANT_MERCHANTURL", url);
apiMacSha256.setParameter("DS_MERCHANT_URLOK", urlOK);
apiMacSha256.setParameter("DS_MERCHANT_URLKO", urlKO);
```

Finally, you must call up the library function "createMerchantParameters()", as shown below:

```
String params = apiMacSha256.createMerchantParameters();
```

4. Calculate the parameter Ds_Signature. To calculate this parameter, you must call up the library function "createMerchantSignature()" with the code obtained

```
String claveModuloAdmin = "Mk9m98IfEblmPfrpsawt7BmxObt98Jev";
String signature = apiMacSha256.createMerchantSignature(claveModuloAdmin);
```

from the administration module, as shown below:

5. Once the values of parameters Ds_MerchantParameters and Ds_Signature have been obtained, the payment form must be filled in with the values obtained, as shown below:

```
<form action="https://sis.redsys.es/sis/realizarPago"
    method="POST" target="_blank">

    <input type="text" name="Ds_SignatureVersion"
        value="HMAC_SHA256_V1" />
    <input type="text" name="Ds_MerchantParameters"
        value="<%= params %>" />
    <input type="text" name="Ds_Signature"
    value="<%= signature %>" />
    <input type="submit" value="Realizar Pago" />

</form>
```

.NET Library

The steps that a merchant must follow to use the .NET library provided by BBVA are shown below:

1. Import the RedsysAPI and Newronsoft.Json library into your project.

2. Calculate the parameter Ds_MerchantParameters. To calculate this parameter, you must initially add all the parameters of the payment request you want to send, as shown below:

```
// New instance of RedysAPI
RedsysAPI r = new RedsysAPI();

// Fill Ds_MerchantParameters parameters
r.SetParameter("DS_MERCHANT_AMOUNT", amount);
r.SetParameter("DS_MERCHANT_ORDER", id);
r.SetParameter("DS_MERCHANT_MERCHANTCODE", fuc);
r.SetParameter("DS_MERCHANT_CURRENCY", currency);
r.SetParameter("DS_MERCHANT_TRANSACTIONTYPE", trans);
r.SetParameter("DS_MERCHANT_TERMINAL", terminal);
r.SetParameter("DS_MERCHANT_MERCHANTURL", url);
r.SetParameter("DS_MERCHANT_URLOK", urlOK);
r.SetParameter("DS_MERCHANT_URLKO", urlKO);
```

Finally, you must call up the library function "createMerchantParameters()", as shown below:

```
string parms = r.createMerchantParameters();
Ds_MerchantParameters.Value = parms;
```

3. Calculate the parameter Ds_Signature. To calculate this parameter, you must call up the library function "createMerchantSignature()" with the code obtained from the administration module, as shown below:

```
string sig = r.createMerchantSignature(kc);
Ds_Signature.Value = sig;
```

4. Once the values of parameters Ds_MerchantParameters and Ds_Signature have been obtained, the payment form must be filled in with the values obtained, as shown below:

```
<form action="https://sis-d.redsys.es:25443/sis/realizarPago"
method="post">
    <input runat="server" type="text" id="Ds_SignatureVersion"
        name="Ds_SignatureVersion" value="" />
    <input runat="server" size="100" type="text" id="Ds_MerchantParameters"
        name="Ds_MerchantParameters" value="" />
    <input runat="server" type="text" size="50" id="Ds_Signature"
        name="Ds_Signature" value="" />
    <input runat="server" type="submit" value="Realizar Pago" />
</form>
```

ONLINE NOTIFICATION

Online notification is an optional feature that allows the web store to receive the result of a transaction online and in real time, once the customer has completed the process in the Virtual POS.

The merchant must capture and validate all parameters together with the signature of the online notification prior to executing anything on its server.

The Virtual POS has different types of notification and are as follows:

1. **Synchronous**. Implies that the result of the purchase is sent to the merchant first and then to the customer. Even if the confirmation is incorrect, the operation does not change.
2. **Asynchronous**. Implies that the result of the authorization is communicated both to the merchant and the customer.
3. **SíncronaSOAP (SynchronousSOAP)**. The notification sent to the merchant is a SOAP request to a service that the merchant must have published. With this type of notification, the SIS does not reply to the cardholder until it receives the merchant's confirmation that is has received the notification. In the event that the SOAP response sent by the merchant has a KO value or an error occurs in the notification process, a negative response will be given to the cardholder and the transaction will not be

authorized. This type of notification will only apply to the following operations: Authorization, Preauthorization, Recurring Transaction and Authentication. For all other operations, the notification shall be sent synchronously. This type of synchronization is explained in detail in the following subsection.

4. **SíncronaSOAP (SynchronousSOAP)** con WSDL. Just like SíncronaSOAP, but in this case the SOAP server developed by the customer conforms to the specifications of a WSDL described in Appendix 6.3 (SOAP Notification) of the Appendices section of this document. This last type of notification is recommended, which guarantees a perfect understanding between the server and client.

SYNCHRONOUS AND ASYNCHRONOUS NOTIFICATION

The use of the help libraries provided by BBVA is explained in the following subsections and will depend on the type of notification configured.

The previous sections have described how to access the SIS using the Redirect Connection and the HMAC SHA256-based signature system. This section explains how the available libraries PHP, JAVA and .NET are used to facilitate the developments for receiving online notification parameters and validation of the signature. Using the libraries supplied by BBVA is optional, although they simplify the developments to be carried out by the merchant.

PHP Library

Steps to follow:

1. Import the main library file as shown below:

```
include_once 'redsysHMAC256_API_PHP_4.0.2/apiRedsys.php';
```

The merchant must decide whether it wants to do the import with the "include" or "required" function, as per developments.

2. Define an object of the main class in the library, as shown below:

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

3. Capture the online notification parameters:

```
$version = $_POST["Ds_SignatureVersion"];
$params = $_POST["Ds_MerchantParameters"];
$signatureRecibida = $_POST["Ds_Signature"];
```

4. Decode the Ds_MerchantParameters parameter. To decode this parameter, you must call up the "decodeMerchantParameters()" library function, as shown below:

```
$decodec = $miObj->decodeMerchantParameters($params);
```

Once you have called up the "decodeMerchantParameters()" function, you can obtain the value of any parameter that is likely to be included in the online notification. To obtain the value of a parameter, call up the function "getParameter()" of the library with the parameter name, as shown below to get the response code:

```
$codigoRespuesta = $miObj->getParameter("Ds_Response");
```

IMPORTANT NOTE: To ensure the security and origin of notifications, the merchant must validate the signature received and all parameters sent in the notification.

5. Validate the Ds_Signature parameter. To validate this parameter, the signature must be calculated and compared with the Ds_Signature parameter captured. To do this, call up the "createMerchantSignatureNotifO" library function with the code obtained from the administration module and the Ds_MerchantParameters parameter captured, as shown below:

```
$claveModuloAdmin = 'Mk9m98IfEblmPfrpsawt7BmxObt98Jev';
$signatureCalculada = $miObj->createMerchantSignatureNotif($claveModuloAdmin,
                                                           $params);
```

Once this is done, you can validate whether the value of the signature sent matches the value of the calculated signature, as shown below:

```
if ($signatureCalculada === $signatureRecibida){
    echo "FIRMA OK. Realizar tareas en el servidor";
} else {
    echo "FIRMA KO. Error, firma inválida";
}
```

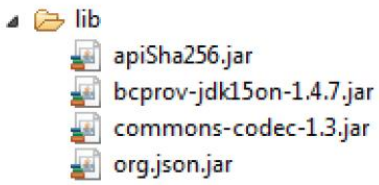
JAVA Library

The steps that a merchant must follow to use the JAVA library provided by BBVA are shown below:

1. Import the library as shown below:

```
<%@page import="sis.redsys.api.ApiMacSha256"%>
```

The merchant must include under the project's construction all the libraries (JARs) provided:



2. Define an object of the main class in the library, as shown below:

```
ApiMacSha256 apiMacSha256 = new ApiMacSha256();
```

3. Capture the online notification parameters:

```
String version = request.getParameter("Ds_SignatureVersion");
String params = request.getParameter("Ds_MerchantParameters");
String signatureRecibida = request.getParameter("Ds_Signature");
```

4. Decoding the **Ds_MerchantParameters** parameter. To decode this parameter, you must call up the "decodeMerchantParameters()" library function, as shown below:

```
String decodec = apiMacSha256.decodeMerchantParameters(params);
```

Once you have called up the "decodeMerchantParameters()" function, you can obtain the value of any parameter that is likely to be included in the online notification. To obtain the value of a parameter, call up the function "getParameter()" of the library with the parameter name, as shown below to get the response code:

```
String codigoRespuesta = apiMacSha256.getParameter("Ds_Response");
```

IMPORTANT NOTE: To ensure the security and origin of notifications, the merchant must validate the signature received and all parameters sent in the notification.

5. Validate the **Ds_Signature** parameter. To validate this parameter, the signature must be calculated and compared with the Ds_Signature parameter captured. To do this, call up the "createMerchantSignatureNotif()" library function with the code obtained from the administration module and the **Ds_MerchantParameters** parameter captured, as shown below:

```
String claveModuloAdmin = "Mk9m98IfEblmPfrpsawt7BmxObt98Jev";
String signatureCalculada = apiMacSha256.createMerchantSignatureNotif(claveModuloAdmin,
                                                                      params);
```

Once this is done, you can validate whether the value of the signature sent matches the value of the calculated signature, as shown below:

```
if (signatureCalculada.equals(signatureRecibida)) {
    System.out.println("FIRMA OK. Realizar tareas en el servidor");
} else {
    System.out.println("FIRMA KO. Error, firma inválida");
}
```

.NET Library

The steps that a merchant must follow to use the .NET library provided by BBVA are shown below:

1. Import the RedsysAPI and Newronsoft.Json library into your project.

2. Capture the online notification parameters:

```
// New instance of RedsysAPI
RedsysAPI r = new RedsysAPI();

// Obtain Ds_SignatureVersion using post
if (Request.Form["Ds_SignatureVersion"] != null)
{
    version = Request.Form["Ds_SignatureVersion"];
}

// Obtain Ds_MerchantParameters using post
if (Request.Form["Ds_MerchantParameters"] != null)
{
    data = Request.Form["Ds_MerchantParameters"];
}

// Obtan Ds_Signature using post
if (Request.Form["Ds_Signature"] != null)
{
    signatureReceived = Request.Form["Ds_Signature"];
}
```

3. Decoding [illegible] decode this parameter, you must call up the "decodeMerchantParameters()" library function that generates the JSON string of the response, as shown below:

```
string deco = r.decodeMerchantParameters(data);
```

IMPORTANT NOTE: To ensure the security and origin of notifications, the merchant must validate the signature received and all parameters sent in the notification.

4. Validate the Ds_Signature parameter. To validate this parameter, the signature must be calculated and compared with the Ds_Signature parameter captured. To do this, call up the "createMerchantSignatureNotif()" library function with the code obtained from the administration module and the Ds_MerchantParameters parameter captured, as shown below:

```
var kc = "Mk9m98IfEblmPfrpsawt7BmxObt98Jev";
string notif = r.createMerchantSignatureNotif(kc, data);
```

Once this is done, you can validate whether the value of the signature sent matches the value of the calculated signature, as shown below:

```
string text = "";
if (notif.Equals(signatureReceived) && notif != "")
{
    text = "SIGNATURE OK";
}
else
{
    text = "SIGNATURE KO";
}
```

SOAP SYNCHRONIZATION

This synchronization method allows the merchant to receive a notification of the transaction in a SOAP service. If the merchant has no privileges to activate this permission with its user, it must request activation through its organization. This synchronization is an actual notification, so it makes no sense to fill in the online notification field, as it will not be considered.

If the SincronizaciónSOAP option is enabled for a merchant, it will mean that the SIS will send notifications for Authorization, Preauthorization, Delayed Authorization, Recurring Transaction and Authentication as SOAP requests to a service that the merchant will have published. For other operations, notifications will be made synchronously and according to the option chosen in the merchant's configuration for online notifications.

The main peculiarity of this notification is that the SIS waits for a response to the notification before submitting the result of the operation to the cardholder making the purchase. In the case

where the merchant returns a response with a KO value or an error occurs during the notification process, the SIS will cancel the operation and provide the cardholder with a receipt with the KO result, i.e. the SIS makes the result of the operation conditional on the response it obtains from the merchant in the notification.

The URL of the rpcrouter to which the SIS will connect and where the SOAP service will be published must be sent by the merchant in the 'Ds_Merchant_MerchantURL' parameter of the SIS input form. The characteristics of the SOAP service to be published by merchants are described in Appendix 3 (SOAP Notification) of this document's Appendices section.

This section explains how the available PHP, JAVA and .NET libraries are used to facilitate the developments for receiving online notification (SOAP) parameters and validating the signature.

Using the libraries supplied by BBVA is optional, although they simplify the developments to be carried out by the merchant.

PHP Library

The steps that a merchant must follow to use the PHP library provided by BBVA are shown below:

1. Import the main library file as shown below:

```
include_once 'redsysHMAC256_API_PHP_4.0.2/apiRedsys.php';
```

The merchant must decide whether it wants to do the import with the "include" or "required" function, as per developments.

2. Define an object of the main class in the library, as shown below:

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

3. Validate the signature sent in the notification. To validate this parameter, the signature must be calculated and compared with the signature that is sent in the notification. To perform the signature calculation, call up the "createMerchantSignatureNotifSOAPRequest()" library function with the code obtained from the administration module and the value of the message received in the notification.

```
function procesaNotificacionSIS($XML) {

$claveModuloAdmin = 'Mk9m98IfEblmPfrpsawt7BmxObt98Jev';
$signatureCalculada = $miObj->createMerchantSignatureNotifSOAPRequest($claveModuloAdmin,$XML);
```

Once this is done, the merchant must capture the value of the signature received (<Signature> parameter) and validate whether its value matches the value of the calculated signature, as shown below.

```
if ($signatureCalculada === $signatureRecibida){
    echo "FIRMA OK. Realizar tareas en el servidor";
} else {
    echo "FIRMA KO. Error, firma inválida";
}
```

IMPORTANT NOTE: To ensure the security and origin of notifications, the merchant must validate the signature received and all parameters sent in the notification.

4. Once the signature is validated, the merchant must send the notification response. This response is signed and in order to carry out the calculation of the signature, the order number of the message received in the notification must first be entered. To obtain the purchase order number, call up the "getOrderNotifSOAP()" library function with the value of the message received in the notification. Once the order number has been obtained, all that remains is to calculate the signature to be sent in the response. To perform the signature calculation, call up the "createMerchantSignatureNotifSOAPResponse()" library function with the code obtained from the administration module and the value of the response message and the captured order number, as shown below:

```
$numPedido = $miObj->getOrderNotifSOAP($XML);

$response='<Response Ds_Version="0.0">
                <Ds_Response_Merchant>OK</Ds_Response_Merchant>
            </Response>';

$claveModuloAdmin = 'Mk9m98IfEblmPfrpsawt7BmxObt98Jev';

$firmaRespuesta = $miObj->createMerchantSignatureNotifSOAPResponse($claveModuloAdmin,
                                                                   $response,
                                                                   $numPedido);
```

Finally, the final message must be formed using the response message and the

signature obtained, as described in Appendix 6.3 (SOAP Notification) in the Appendices section of this manual.

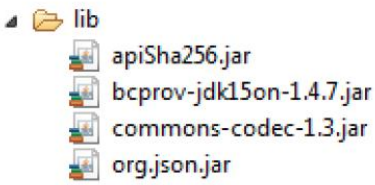
JAVA Library

The steps that a merchant must follow to use the JAVA library provided by BBVA are shown below:

1. Import the library as shown below:

```
<%@page import="sis.redsys.api.ApiMacSha256"%>
```

The merchant must include under the project's construction all the libraries (JARs) provided:



2. Define an object of the main class in the library, as shown below:

```
ApiMacSha256 apiMacSha256 = new ApiMacSha256();
```

3. Validate the signature sent in the notification. To validate this parameter, the signature must be calculated and compared with the signature that is sent in the notification. To perform the signature calculation, call up the "createMerchantSignatureNotifSOAPRequest()" library function with the code obtained from the administration module and the value of the message received in the notification.

```
String claveModuloAdmin = "Mk9m98IfEblmPfrpsawt7BmxObt98Jev";
String signatureCalculada = apiMacSha256.createMerchantSignatureNotifSOAPRequest(claveModuloAdmin, XML);
```

Once this has been done, the merchant must capture the value of the (<Signature> parameter) received and validate whether its value matches the value of the calculated signature, as shown below:

```
if (signatureCalculada.equals(signatureRecibida)) {
    System.out.println("FIRMA OK. Realizar tareas en el servidor");
} else {
    System.out.println("FIRMA KO. Error, firma inválida");
}
```

IMPORTANT NOTE: To ensure the security and origin of notifications, the merchant must validate the signature received and all parameters sent in the notification.

4. Once the signature is validated, the merchant must send the notification response. This response is signed and in order to carry out the calculation of the signature, the order number of the message received in the notification must first be entered. To obtain the purchase order number, call up the "getOrderNotifSOAP()" library function with the value of the message received in the notification.

   Once the order number has been obtained, all that remains is to calculate the signature to be sent in the response. To perform the signature calculation, call up the "createMerchantSignatureNotifSOAPResponse()" library function with the code obtained from the administration module and the value of the response message and the captured order number, as shown below:

```
String numPedido = apiMacSha256.getOrderNotifSOAP(XML);

String respons = "<Response Ds_Version='0.0'><Ds_Response_Merchant>OK</Ds_Response_Merchant></Response>";

String claveModuloAdmin = "Mk9m98IfEblmPfrpsawt7BmxObt98Jev";

String signatureCalculada = apiMacSha256.createMerchantSignatureNotifSOAPResponse(claveModuloAdmin,
                                                                                   respons,
                                                                                   numPedido);
```

   Finally, the final message must be formed using the response message and the signature obtained, as described in Appendix 9.3 (SOAP Notification) in this document.

.NET Library

The steps that a merchant must follow to use the .NET library provided by BBVA are shown below:

1. Import the library as shown below:

```
RedsysAPISoap r = new RedsysAPISoap();
```

2. Validate the signature sent in the notification. To validate this parameter, the signature must be calculated and compared with the signature that is sent in the notification. To perform the signature calculation, call up the

"createMerchantSignatureNotifSOAPRequest()" library function with the code obtained from the administration module and the value of the message received in the notification.

```
var kc = "sq7HjrUOBfKmC576ILgskD5srU870gJ7";
string signatureCalculate = r.createMerchantSignatureNotifSOAPRequest(kc, XML);
```

Once this has been done, the merchant must capture the value of the (**<Signature>** parameter) received and validate whether its value matches the value of the calculated signature, as shown below:

```
signatureReceived = r.GetParameter(XML, "<Signature>", "</Signature>");

    if (signatureCalculate == signatureReceived)
    {
        res = "FIRMA OK";
    }
    else
    {
        res = "FIRMA KO";
    }
```

IMPORTANT NOTE: To ensure the security and origin of notifications, the merchant must validate the received signature and all parameters sent in the notification.

3. Once the signature is validated, the merchant must send the notification response. This response is signed and in order to carry out the calculation of the signature, the order number of the message received in the notification must first be entered. To obtain the purchase order number, call up the "getOrderNotifSOAP()" library function with the value of the message received in the notification.

   Once the order number has been obtained, all that remains is to calculate the signature to be sent in the response. To perform the signature calculation, call up the "createMerchantSignatureNotifSOAPResponse()" library function with the code obtained from the administration module and the value of the response message and the captured order number, as shown below:

```
string numOrder = r.GetOrderNotifSOAP(XML);
string respons = "<Response Ds_Version='0.0'><Ds_Response_Merchant>OK</Ds_Response_Merchant></Response>";
string signatureResponse = r.createSignatureNotifSOAPResponse(kc, respons, numOrder);
```

Finally, the final message must be formed using the response message and the signature obtained, as described in Appendix 6.3 (SOAP Notification) of this manual.

RETURN OF NAVIGATION CONTROL

Once the customer has completed the process in the Virtual POS, the navigation is redirected to the web store. This return to the store's website is made to the URL communicated as a parameter in the initial call to the Virtual POS or, failing that, it is obtained from the terminal's configuration in the Virtual POS administration module. Different return URLs may be available depending on the outcome of the transaction (URL OK and URL KO).

The merchant must capture and validate, if its merchant's configuration so requires (Parameter in the URLs = YES), the parameters of the return of navigation control prior to any execution on its server.

The use of the help libraries provided by BBVA for capturing and validating the parameters of the navigation control return is explained below.

Using the libraries supplied by BBVA is optional, although they simplify the developments to be carried out by the merchant.

PHP Library

The steps that a merchant must follow to use the PHP library provided by BBVA are shown below:

1. Import the main library file as shown below:

```
include_once 'redsysHMAC256_API_PHP_4.0.2/apiRedsys.php';
```

   The merchant must decide whether it wants to do the import with the "include" or "required" function, as per developments.

2. Define an object of the main class in the library, as shown below:

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

3. Capture the online notification parameters:

```
$version = $_GET["Ds_SignatureVersion"];
$params = $_GET["Ds_MerchantParameters"];
$signatureRecibida = $_GET["Ds_Signature"];
```

4. Decode the **Ds_MerchantParameters** parameter. To decode this parameter, you must call up the "decodeMerchantParameters()" library function, as shown below:

```
$decodec = $miObj->decodeMerchantParameters($params);
```

Once you have called up the "decodeMerchantParameters()" function, you can obtain the value of any parameter that may be included in the online notification (Appendix 2 of the Appendices section of this document). To obtain the value of a parameter, call up the "getParameter()" function of the

library with the parameter name, as shown below to get the response code:

```
$codigoRespuesta = $miObj->getParameter("Ds_Response");
```

IMPORTANT NOTE: It is important to validate all the parameters sent in the communication. To update the status online, do NOT use this communication but the online notification described in the other sections, since the navigation return depends on what the customer does in its browser.

5. Validate the **Ds_Signature** parameter. To validate this parameter, the signature must be calculated and compared with the **Ds_Signature** parameter captured. To do this, call up the "createMerchantSignatureNotif()" library function with the code obtained from the administration module and the **Ds_MerchantParameters** parameter captured, as shown below:

```
$claveModuloAdmin = 'Mk9m98IfEblmPfrpsawt7BmxObt98Jev';
$signatureCalculada = $miObj->createMerchantSignatureNotif($claveModuloAdmin,
                                                           $params);
```

matches the value of the calculated signature, as shown below:

```
if ($signatureCalculada === $signatureRecibida){
    echo "FIRMA OK. Realizar tareas en el servidor";
} else {
    echo "FIRMA KO. Error, firma inválida";
}
```

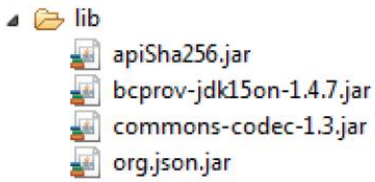
JAVA Library

The steps that a merchant must follow to use the JAVA library provided by BBVA are shown below:

1. Import the library as shown below:

```
<%@page import="sis.redsys.api.ApiMacSha256"%>
```

   The merchant must include under the project's construction all the libraries (JARs) provided:



2. Define an object of the main class in the library, as shown below:

```
ApiMacSha256 apiMacSha256 = new ApiMacSha256();
```

3. Capture the parameters of the navigation control return:

```
String version = request.getParameter("Ds_SignatureVersion");
String params = request.getParameter("Ds_MerchantParameters");
String signatureRecibida = request.getParameter("Ds_Signature");
```

4. Decoding the **Ds_MerchantParameters** parameters. To decode this parameter, you must call up the "decodeMerchantParameters()" library function, as shown below:

```
String decodec = apiMacSha256.decodeMerchantParameters(params);
```

   Once you have called up the "decodeMerchantParameters()" function, you can obtain the value of any parameter that may be included in the navigation control return (Appendix 2 of the Appendices section of this document). To

obtain the value of a parameter, call up the function "getParameter()" of the library with the parameter name, as shown below to get the response code:

```
String codigoRespuesta = apiMacSha256.getParameter("Ds_Response");
```

IMPORTANT NOTE: It is important to validate all the parameters sent in the communication. To update the status online, do NOT use this communication but the online notification described in the other sections, since the navigation return depends on what the customer does in its browser.

5. Validate the **Ds_Signature** parameter. To validate this parameter, the signature must be calculated and compared with the **Ds_Signature** parameter captured. To do this, call up the "createMerchantSignatureNotif()" library function with the code obtained from the administration module and the **Ds_MerchantParameters** parameter captured, as shown below:

```
String claveModuloAdmin = "Mk9m98IfEblmPfrpsawt7BmxObt98Jev";
String signatureCalculada = apiMacSha256.createMerchantSignatureNotif(claveModuloAdmin,
                                                                      params);
```

Once this is done, you can validate whether the value of the signature sent matches the value of the calculated signature, as shown below:

```
if (signatureCalculada.equals(signatureRecibida)) {
    System.out.println("FIRMA OK. Realizar tareas en el servidor");
} else {
    System.out.println("FIRMA KO. Error, firma inválida");
}
```

.NET Library

The steps that a merchant must follow to use the .NET library provided by BBVA are shown below:

1. Import the library as shown below:

```
using RedsysAPIPrj;
```

2. Define an object of the main class in the library, as shown below:

```
RedsysAPI r = new RedsysAPI();
```

3. Capture the parameters of the navigation control return:

```
string version = Request.QueryString["Ds_SignatureVersion"];

string parms = Request.QueryString["Ds_MerchantParameters"];

string signatureRecibida = Request.QueryString["Ds_Signature"];
```

IMPORTANT NOTE: It is important to validate all the parameters sent in the communication. To update the status online, do NOT use this communication but the online notification described in the other sections, since the navigation return depends on what the customer does in its browser.

4. Validate the Ds_Signature parameter. To validate this parameter, the signature must be calculated and compared with the Ds_Signature parameter captured. To do this, call up the "createMerchantSignatureNotif()" library function with the code obtained from the administration module and the Ds_MerchantParameters parameter captured, as shown below:

```
var kc = "sq7HjrUOBfKmC576ILgskD5srU870gJ7";

string signatureCalculada =  r.createMerchantSignatureNotif(kc, parms);
```

Once this is done, you can validate whether the value of the signature sent matches the value of the calculated signature, as shown below:

```
if (signatureRecibida == signatureCalculada)
{
    result.InnerHtml = "FIRMA OK. Realizar tareas en el servidor";
}
else
{
    result.InnerHtml = "FIRMA KO. Error, firma invalida";
}
```

3.4 IN-APP INSTALLATION

3.4.1 Manual for integrating the Android TPVvirtual framework

Introduction

This document defines the functionality included in the framework for Android, which allows payment operations to be made. The framework will be prepared for the available environments: development, integration and production.

This documentation refers to non-secure payment (methods without WebView) and secure payment, with 3DSecure authentication if required, in all methods with WebView included.

Library integration.

Eclipse (deprecated)

To include the library in the project, it is necessary to carry out the following steps:

1. Create a new library project in the workspace.
2. Unzip the file TPVVinApp.aar.
3. Copy the classes.jar file contained in the previous package into the libs folder of your library project.
4. Replace the res directory with that of the .aar file
5. Copy the gson.jar library from the libs directory.
6. Add this library project to your project.

Android Studio

1. Create a new module from the. AAR file within our project.
    a. To do this, go to File-> New Module to Import .JAR or .AAR Package.
2. In the build.gradle of our project we will add the following Project compile line (':TPVinAppLibrary) in the dependencies section.
    a. We must also indicate that you need to compile the gson library for proper operation: compile 'com.google.code.gson:gson:2.+'

Permissions required in the Manifest

For the library to operate properly, at least the following permissions must be declared inside the file "AndroidManifest.xml" of the application that makes use of it:

Requests to web services

- android.permission.INTERNET

Available operations

The framework provided implements a number of methods to perform the following operations:

1. Purchase
2. Purchase with a reference request
3. Purchase with reference
4. Purchase in WebView
5. Purchase in WebView with reference request
6. Purchase in WebView with reference

The first operation allows you to purchase a product by entering the necessary data. These are card number, expiry date and cvv2 code.

The second method allows you to make a purchase, just like the previous operation, but requesting a reference for future payments where you will request it instead of requesting the card data.

The third option allows you to make a purchase with the previously requested reference. These payments are called "One Click" since, as it is unnecessary to enter the card data, they are the fastest.

In the next three processes, a WebView is shown, with the information of the embedded purchase, where the card details are requested, also, if you have a previous reference, you can make the payment directly.

Configuration

As a first step, it is necessary to establish a series of parameters common to all the methods that will be used in the class "IAManualEntryActivity". These parameters are:

1. Application license:

   IAConfigurationLibrary.setAppLicense("xxxxxx");

   To apply for this license you will need to contact our mobility department (movilidad@redsys.es) indicating the name of the application package that will use this library as well as the merchant(s) and terminals that you want to register for this application.

2. Execution environment: There are 3 environments available:

- development (IAConstants.ENVIRONMENT_DEVELOPMENT) ,
- integration (IAConstants.ENVIRONMENT_INTEGRATION),
- production (IAConstants.ENVIRONMENT_REAL).

The environment is set using the constants defined in class "IAConstants". For stability reasons, it is recommended to only use the integration and production environment.

3. Merchant's FUC.
4. Terminal.
5. Order number.
6. The value of the transaction.
7. Currency code used. Value "978" for euros if not indicated.
8. Type of transaction:

a. IAConstants.NORMAL_PAY : Normal payment --> default value in payments with WebView

b. IAConstants.PREAUTHORIZATION_PAY : Pre-authorization

c. IAConstants.TRADITIONAL_PAY: Traditional Payment --> Default value for direct payments

To do this, you will need to move all these data to this class using an intent.

For instance:

```
intent.putExtra(IAConstants.EXTRA_ENVIRONMENT,
IAConstants.ENVIRONMENT_xxxxxx);

intent.putExtra(IAConstants.EXTRA_MERCHANT_CODE, "xxxxxxxxx");

intent.putExtra(IAConstants.EXTRA_MERCHANT_TERMINAL,"x");

intent.putExtra(IAConstants.EXTRA_ORDER_CODE,"xxxx");

intent.putExtra(IAConstants.EXTRA_AMOUNT,xx.xx); //double

intent.putExtra((IAConstants.EXTRA_MERCHANT_CURRENCY, currency);

intent.putExtra((IAConstants.EXTRA_OPERATION_TYPE, operationType);
//Send Traditional by default
```

Purchase

In order to make a direct purchase, without the cardholder's authentication (Unsecure), it will be necessary to create an intent to the class: IAManualEntryActivity and include the commented fields in the previous example.

Purchase with a reference request

If, on the other hand, you need to request a reference for a future sale, you must create an intent to the class: IAManualEntryActivity and add the parameters (in addition to those included in the configuration section):

```
intent.putExtra(IAConstants.EXTRA_MERCHANT_IDENTIFIER,
IAConstants.VALUE_MERCHANT_IDENTIFIER_REQUIRED);
```

Where "VALUE_MERCHANT_IDENTIFIER_REQUIRED" constant is defined in the class "IAConstants"

```
// VALORES OPCIONALES
intent.putExtra(IAConstants.EXTRA_DS_MERCHANT_GROUP,
"merchantGroup"); intent.putExtra(IAConstants.EXTRA_DS_MERCHANT_URL,
merchantURL);
¡ntent.putExtra(IAConstants.EXTRA_DS_MERCHANT_DIRECT_PAYMENT,
"true"); // true or false
```

Purchase with reference

If you already have a reference, all you need to do is add, in addition to the basic parameters, the reference in the field "Merchant_identifier" and create an intent to the class: IAManualEntryActivity.

```
intent.putExtra(IAConstants.EXTRA_MERCHANT_IDENTIFIER,reference);
// VALORES OPCIONALES
intent.putExtra(IAConstants.EXTRA_DS_MERCHANT_GROUP,"merchantGrou
p"); intent.putExtra(IAConstants.EXTRA_DS_MERCHANT_URL,merchantURL);
intent.putExtra(IAConstants.EXTRA_DS_MERCHANT_DIRECT_PAYMENT,
"true"); // true or false
```

Where reference will contain the reference collected in the response to the call to request a reference (or a reference that we already have available).

Purchase in WebView

In this option, you must create an intent to class IAWebViewEntryActivity, by adding the base fields. If you would like to make a purchase with a reference, we would also have to add the following value:

```
intent.putExtra(IAConstants.EXTRA_MERCHANT_IDENTIFIER,
IAConstants.VALUE_MERCHANT_IDENTIFIER_REQUIRED);
```

If we have a reference, we will be able to send it directly, otherwise, we will send the value IAConstants.VALUE_MERCHANT_IDENTIFIER_REQUIRED to make a request.

To receive a response at the end of the payment from a WebView, you must define certain url OK values (If the payment was correct) or urlKO if it was incorrect.

```
intent.putExtra(IAConstants.EXTRA_DS_MERCHANT_URL_OK,
"merchantURLOK");
intent.putExtra(IAConstants.EXTRA_DS_MERCHANT_URL_KO,
"merchantURLKO");
```

//by default, the following values can be used if no specific url is available:

URL OK: https://sis-d.redsys.es/PruebasSDF_V2Web/imode/ok.jsp
URL KO: https://sis-d.redsys.es/PruebasSDF_V2Web/imode/ko.jsp
// VALORES OPCIONALES

```
intent.putExtra(IAConstants.EXTRA_DS_MERCHANT_DATA,merchantData);
intent.putExtra(IAConstants.EXTRA_DS_MERCHANT_GROUP,"merchantGrou
p"); intent.putExtra(IAConstants.EXTRA_DS_MERCHANT_URL,merchantURL);
intent.putExtra(IAConstants.EXTRA_DS_MERCHANT_DIRECT_PAYMENT,
"true"); // true or false
```

Response codes

All the intents created in the previous calls must wait for a result. To do this, in the onActivityResult method, you can receive the following responses from the library:

- RESULT_OK: the payment has been made successfully.
- RESULT_CANCELED: The user has canceled the payment.
- ACTIVITY_RESULT_NO_VALID_PAYMENT: Payment error. The data entered is invalid.
- ACTIVITY_RESULT_ERR: An error occurred in the payment.

The object response received will be of the type IAPetitionResponse which contains the following fields that can be queried with the respective getters and extracted in the field IAConstants.EXTRA_PETITION_RESPONSE:

```
String response;
String responseCode
String responseAmount
String responseCurrency
String responseOrder
String responseSignature
String responseMerchantCode
String responseTerminal
```

String responseResponse
String responseAuthorisationCode
String responseTransactionType
String responseSecurePayment
String responseLanguage
String responseCardNumber
String responseCardType
String responseMerchantData
String responseCardCountry
String responseDate
String responseHour
String responseIdentifier
Boolean isResponseSignatureCorrect
String responseNSU;
String rts;

In the event that the payment is not made correctly, you will receive the code ACTIVITY_RESULT_ERR, the response must be extracted from the fieldIAConstants.EXTRA_PETITION_RESPONSE_ERROR and we'll get an object IAErrReturnObject with a code and an error message.

Below you can find a list of possible error codes returned by the library:

CODE	VALUE	MEANING
ENCRYPTION_DESENCRYPTION_SIGNATURE_ERROR	ERR_INAPP_0001	Signature error (usually occurs when a parameter entered is incorrect)
GENERIC_ERROR	ERR_INAPP_0002	Generic error
COMMUNICATION_WITH_WEBSERVICE_FAILED	ERR_INAPP_0010	Error communicating with the server
INVALID_VALUES_FOR_A_PETITION	ERR_INAPP_0020	The fields entered by the user are incorrect
SERVER_ERROR_RESPONSE	ERR_INAPP_0030	Server response error
NO_FIELDS_IN_RESPONSE	ERR_INAPP_0040	The operation (through WebView) was erroneous and there are no results to show in the response.
11	11	The message's signature is incorrect
29	29	Service cryptography failure
31	31	Incorrect application
60	60	Incorrect JSON format
61	61	Error when obtaining the merchant's signature code
62	62	Terminal signature type unsupported
78	78	Description with SIS error code. Consult SIS documentation for more information.

Latest changes in the new version

- The library has been updated to the new SIS signature service for its merchants.

- Options added for making a regular, traditional or pre-authorization payment. To this end, the need to include the field "IAConstants.EXTRA_OPERATION_TYPE" in all operations has been identified. If not specified, by default, use Traditional Payment for direct payments and Normal Payment for WebView payments.

  Similarly, to ensure that WebView operations are answered, you need to send the url ok and ko fields.

- The amount data type has been changed to use a double.

- The list of error codes has been extended.

- Due to problems of incompatibility with the gson library included as jar, the way of using it has been changed, allowing it to be compiled in the project's build.gradle.

3.4.2 Manual for integrating the TPVVirtual framework in iOS

Introduction

This document defines the functionality contained in the POS framework. This library allows you to perform e-commerce operations in third-party apps.

The framework will be prepared for the integration and production environments.

There are two types of payments in e-commerce: direct and WebView payments. Direct payment features a screen, which is specific to the framework, which collects the user's card data, preventing the app from viewing them.

This method is not subject to 3DSecure authentication. Conversely, payment in WebView does allow different authentication methods.

Library integration

To include the library in the project, you must carry out the following steps:

1. Import the library and bundle into the project.

Available operations

The framework provided implements a number of methods to perform the following operations:

1. Direct payment
2. Direct payment with a reference request
3. Direct Payment with a reference (on screen)
4. Direct Payment with a reference (direct method)
5. Direct payment in WebView
6. Direct payment in WebView with a reference request
7. Direct payment in WebView with a reference

Configuration

Before beginning the integration, a number of parameters common to all the methods that will be used in the classes "IAManualEntryActivity" and "IAWeblEntryViewController" must be established.

These parameters are:

- Application license: This is an alphanumeric code provided by BBVA that is used to validate the applications that make use of the library.

```
[IAConfigurationLibrary setAppLicense:@"123456789a"];
```

- Execution environment: There are 4 environments available (development: @"des", integration: @"int", ccal: @"ccal" and production: @"real"). It is recommended to use only the integration and production environments.

  The environment is established using the constants defined in class "CommonUtils".

```
[CommonUtils setEntorno:@"int"];
```

  Other parameters: It will be necessary to know the following elements when correctly integrating the library.

- Fuc: Merchant's number on which to operate.
- Terminal: Terminal associated to the merchant's number on which it will operate.
- Order number: Code to identify the order.
- Amount: Value of the payment to be made.
- "Identifier": field indicating whether the requests are for reference or to include it in the payment.
- Merchant Group: (if applicable) Merchants group code
- Currency/Moneda: Currency in which you are going to operate.
- Merchant URL: merchant's Url where the operation is reported.
- URL OK/KO: Urls ok or ko to which the WebView will be directed depending on the result of the purchase.
- Optional Data: Optional data that can be sent by the merchant.
- Transaction Type: Type of transaction to be carried out. The following are allowed:
    - Traditional payments - @"A".
    - Normal payments - @"0".
    - Preauthorizations - @"1".

Direct payment

This method will display a native library screen that will prompt the user for the card data. The app will have to create the next viewController and present it modally:

```
IAManualEntryViewController *mevc =
[[IAManualEntryViewController alloc]
initWithMerchantCode:textFuc.text
merchantTerminal:textTerminal.text orderNumber:[CommonUtils
getDMYHMSActualDate] amount:[textAmount.text floatValue]
identifier:@"" currency:[coinsCode
objectAtIndex:selectedCurrency] merchanGroup:CodigoGrupo.text
DirectPayment:directPaymentValue MerchantUrl:URLrespuesta.text
andTransactionType:tipoOperacion];
[mevc setDelegate:self];
[self presentViewController:mevc animated:YES completion:nil];
```

Direct payment with a reference request

This method is similar to the previous one because it makes a payment, only with the extra functionality of returning a reference associated to the card used. This reference can be used for future payments.

To use this functionality, the merchant must have it activated in its configuration.

To implement it, it will be necessary to create the following viewController, present it modally and add the value in the identifier field @ **"REQUIRED".**

```
IAManualEntryViewController *mevc =
[[IAManualEntryViewController alloc]
initWithMerchantCode:textFuc.text
merchantTerminal:textTerminal.text orderNumber:[CommonUtils
getDMYHMSActualDate] amount:[textAmount.text floatValue]
identifier:@"REQUIRED" currency:[coinsCode
objectAtIndex:selectedCurrency] merchanGroup:CodigoGrupo.text
DirectPayment: directPaymentValue MerchantUrl: URLrespuesta.text
andTransactionType:tipoOperacion];
[mevc setDelegate:self];
[self presentViewController:mevc animated:YES completion:nil];
```

Note:

- "Identifier": field in charge of informing the library about the payment by reference operation. In this case it will be necessary to add the REQUIRED value.
- "DirectPayment": true/false.

Direct Payment with a reference (on screen)

This method is similar to the previous one because it makes a payment, only with the extra functionality of making a reference associated to the card used.

To use this functionality, the merchant must have it activated in its configuration.

To implement it, it will be necessary to create the following viewController, present it modally and add the value of the stored reference.

```
IAManualEntryViewController *mevc =
[[IAManualEntryViewController alloc]
initWithMerchantCode:textFuc.text
merchantTerminal:textTerminal.text orderNumber:[CommonUtils
getDMYHMSActualDate] amount:[textAmount.text floatValue]
identifier:self.reference currency:[coinsCode
objectAtIndex:selectedCurrency] merchanGroup:CodigoGrupo.text
DirectPayment: directPaymentValue MerchantUrl: URLrespuesta.text
andTransactionType:tipoOperacion];
[mevc setDelegate:self];
[self presentViewController:mevc animated:YES completion:nil];
```

Note:

- "Identifier": field in charge of informing the library about the payment by reference operation. In this case the value of the stored reference must be added.
- "DirectPayment": true/false.

Direct Payment with a reference (direct method)

This method allows you to make direct payments with a reference without the need to display an intermediate screen. It is necessary to put together a **"IATPVVirtualPetitionJSON"** type object and then call up the peticionPago method from IAPetitionManager.

For instance:

```
IATPVVirtualPetitionJSON *pet = [[IATPVVirtualPetitionJSON
alloc] initWithMerchantCode:textFuc.text
terminal:textTerminal.text order:[CommonUtils
getDMYHMSActualDate] amount:[textAmount.text floatValue]
currency:[coinsCode objectAtIndex:selectedCurrency] pan:@""
cvv2:@"" expireDate:@"" identifier:self.reference
group:CodigoGrupo.text directPayment:directPaymentValue
merchantUrl:URLrespuesta.text andTransactionType:tipoOperacion];

[IAPetitionManager peticionPago:pet withBlock:^(NSDictionary
*result, NSError *error) {}];
```

Payment in WebView (Deprecated)

This type of operation is not recommended as the merchant's code must be entered in the application. The recommended option is found in point 2.8. The initialization of this method is similar to the previous ones.

```
NSString* formattedNumber = [NSString stringWithFormat:@"%.02f",
0.79];
IAWebEntryViewController *mevc=[[IAWebEntryViewController alloc]
initWithMerchantCode:textFuc.text
merchantTerminal:textTerminal.text
merchantKey:@"qwertyasdf0123456789" orderNumber:[CommonUtils
getDMYHMSActualDate] amount:[textAmount.text floatValue]
currency:@"978" identifier:VALUE_MERCHANT_IDENTIFIER_REQUIRED];
[self presentViewController:mevc animated:YES completion:nil];
```

Payment in WebView (recommended)

To make a secure payment, i.e. through a WebView that may or may not request 3DSecure authentication, it is necessary to create the following viewController:

```
IAWebEntryViewController *mevc=[[IAWebEntryViewController alloc]
initWithMerchantCode:textFuc.text
merchantTerminal:textTerminal.text orderNumber:[CommonUtils
getDMYHMSActualDate] amount:[textAmount.text floatValue]
currency: @"978" identifier:@"" merchanURL:URLrespuesta.text
urlOK:URLOK.text urlKO:URLKO.text
directPayment:directPaymentValue
optionalData:DatosOpcionales.text merchantGroup:CodigoGrupo.text
reference:ref transactionType:@"0"];

[self presentViewController:mevc animated:YES completion:nil];
```

Note: The following parameters appear in this method: Ok URL, Ko URL, merchant group and optional data, explained at the start.

Payment in WebView requesting reference

It is the same case as the previous point except that @"REQUIRED" is added in the reference field:

```
IAWebEntryViewController *mevc=[[IAWebEntryViewController alloc]
initWithMerchantCode:textFuc.text
merchantTerminal:textTerminal.text orderNumber:[CommonUtils
getDMYHMSActualDate] amount:[textAmount.text floatValue]
currency: @"978" identifier:@"REQUIRED"
merchanURL:URLrespuesta.text urlOK:URLOK.text urlKO:URLKO.text
directPayment:directPaymentValue
optionalData:DatosOpcionales.text merchantGroup:CodigoGrupo.text
reference:ref transactionType:@"0"];
```

```
[self presentViewController:mevc animated:YES completion:nil];
```

Note: The following parameters appear in this method: Ok URL, Ko URL, merchant group and optional data, explained at the start.

Payment in WebView with a reference

It is the same case as the previous point except that the reference is added in the reference field:

```
IAWebEntryViewController *mevc=[[IAWebEntryViewController alloc]
initWithMerchantCode:textFuc.text
merchantTerminal:textTerminal.text orderNumber:[CommonUtils
getDMYHMSActualDate] amount:[textAmount.text floatValue]
currency: @"978" identifier:@"REQUIRED"
merchanURL:URLrespuesta.text urlOK:URLOK.text urlKO:URLKO.text
directPayment:directPaymentValue
optionalData:DatosOpcionales.text merchantGroup:CodigoGrupo.text
reference:ref transactionType:@"0"];

[self presentViewController:mevc animated:YES completion:nil];
```

Note: The following parameters appear in this method: Ok URL, Ko URL, merchant group and optional data, explained at the start.

Responses

The result of each operation is returned in the following callbacks. A new callback has been added over previous versions to increase response information. (Maintains compatibility with the previous method).

Callbacks from IAManualEntryActivity ->

Adding the delegate `IAManualViewControllerDelegate` in the file .h.

- `(void)didFinishOperationOK:(IATPVVirtualPetitionResponse DTO *)petitionResponse`
- `(void)didFinishOperationKO:(NSError *)petitionError`

New method:

- `(void)didFinishOperationOKDictionary:(NSDictionary *)petitionResponse`

Callbacks from IAWebEntryViewController ->

Adding the delegate `IAWebEntryViewControllerDelegate` in the file .h.

- **(void)didFinishWebOperationOK:(IATPVVirtualWebViewResponseDTO *) petitionResponse**
- **(void)didFinishOperationKO:(NSError*)petitionError**
- **(void)didFinishWebOperationKO:(NSError*)petitionError**
- **(void)didFinishWebOperationOKDictionary:(NSDictionary*)petition Response**

It will be necessary to take into account, above all, the parameter (petitionResponse. responseIdentifier) as this is the reference for cases where it has been requested.

The use of the following methods is recommended:

- **didFinishOperationOKDictionary**
- **didFinishWebOperationOKDictionary**

Since it allows you to be prepared for any response returned by the server and not to have to update the library.

To process NSDictionary responses, the following implementation is recommended:

With petitionResponse being a NSDictionary:

```
[petitionResponse valueForKey:@"Ds_MerchantCode"]
```

Compatibility with the previous methods is maintained, with the responses being the same:

For the `IAManualEntryViewController`, the response has the following parameters:

```
@interface IATPVVirtualPetitionResponseDTO : NSObject

@property (nonatomic, strong)NSString *code;
@property (nonatomic, strong)NSString *responseCodigo;
@property (nonatomic, strong)NSString *responseAmount;
@property (nonatomic, strong)NSString *responseCurrency;
@property (nonatomic, strong)NSString *responseOrder;
@property (nonatomic, strong)NSString *responseSignature;
@property (nonatomic, strong)NSString *responseMerchantCode;
@property (nonatomic, strong)NSString *responseTerminal;
@property (nonatomic, strong)NSString *responseResponse;
@property (nonatomic, strong)NSString
*responseAuthorisationCode;
@property (nonatomic, strong)NSString *responseTransactionType;
@property (nonatomic, strong)NSString *responseSecurePayment;
@property (nonatomic, strong)NSString *responseLanguage;
@property (nonatomic, strong)NSString *responseCardNumber;
@property (nonatomic, strong)NSString *responseCardType;
@property (nonatomic, strong)NSString *responseMerchantData;
@property (nonatomic, strong)NSString *responseCardCountry;
@property (nonatomic, strong)NSString *responseIdentifier;
@property (nonatomic, strong)NSString *responseDate;
@property (nonatomic, strong)NSString *responseHour;
@property BOOL responseSignatureCorrect;
```

For the `IAWebEntryViewController`, , the response has the following parameters:

```
@interface IATPVVirtualWebViewResponseDTO : NSObject
@property (nonatomic, strong)NSString *responseDate;
@property (nonatomic, strong)NSString *responseAmount;
@property (nonatomic, strong)NSString *responseHour;
@property (nonatomic, strong)NSString *responseSecurePayment;
@property (nonatomic, strong)NSString *responseCurrency;
@property (nonatomic, strong)NSString *responseOrder;
@property (nonatomic, strong)NSString *responseMerchantCode;
@property (nonatomic, strong)NSString *responseTerminal;
@property (nonatomic, strong)NSString *responseResponse;
@property (nonatomic, strong)NSString *responseTransactionType;
@property (nonatomic, strong)NSString *responseMerchantData;
@property (nonatomic, strong)NSString *responseAuthorisationCode;
@property (nonatomic, strong)NSString *responseExpiryDate;
@property (nonatomic, strong)NSString *responseIdentifier;
@property (nonatomic, strong)NSString *responseLanguage;
@property (nonatomic, strong)NSString *responseCardCountry;
```

Error codes

The error codes of the mobile library are as follows:

Code	Descripción
11	The message's signature is incorrect
29	Service cryptography failure
31	Incorrect application
60	Incorrect JSON format
61	Error when obtaining the merchant's signature code
62	Terminal signature type unsupported
78	BBVA virtual POS own error

All errors with code 78 contain the corresponding error of the BBVA virtual POS in their description. The documentation is appended in the following section.

3.5 USE OF EXTERNAL SOLUTIONS

The use of third-party tools to configure your ecommerce site, such as those offered by companies such as Prestashop, Magento, Shopify, etc., are supported by the BBVA Virtual POS. That is, they can be perfectly integrated and configured to use the payment gateway service from the tool itself.

To be able to make this connection correctly, you must ask your provider to provide you with the necessary modules for this purpose, and these must be adapted and updated for optimum use of the tool.

The BBVA Virtual POS service does not create, distribute or maintain any of these modules, since it is the providers themselves who adapt to the BBVA Virtual POS.

4. RESPONSE CODE TABLE (Ds_Response)

Response codes sent by the card's issuing bank

RESPONSE CODES TO INDICATE THAT THE TRANSACTION HAS BEEN APPROVED		
CODE	**BRIEF DESCRIPTION**	**COMMENT**
000	TRANSACTION APPROVED	Transaction authorized by the card's issuing bank
001	TRANSACTION APPROVED AFTER OWNER IDENTIFICATION	Exclusive code for Verified by Visa or MasterCard SecureCode transactions. The transaction has been authorized and the issuing bank also informs us that it has properly authenticated the cardholder's identity.
002 - 099	TRANSACTION APPROVED	Transaction authorized by the issuing bank.
900	TRANSACTION APPROVED	Transaction authorized for returns and confirmations.

RESPONSE CODES TO INDICATE THAT THE TRANSACTION HAS BEEN REFUSED		
CODE	**BRIEF DESCRIPTION**	**COMMENT**
101	EXPIRED CARD	Transaction denied because the expiry date of the card that has been reported in the payment is earlier than the current effective date.
102	CARD TEMPORARILY BLOCKED OR UNDER SUSPICION OF FRAUD	Card temporarily blocked by the issuing bank or under suspicion of fraud.
104	OPERATION NOT PERMITTED.	Operation not allowed for that card type.
106	NUMBER OF ATTEMPTS EXCEEDEDED	Number of attempts with an incorrect PIN exceeded.

CODE	BRIEF DESCRIPTION	COMMENT
107	CONTACTING THE ISSUER	The issuing bank does not allow automatic approval. You must phone your local Authorization Center to obtain manual approval.
109	INVALID IDENTIFICATION OF THE MERCHANT OR TERMINAL	Denied because the merchant is not properly registered in international card systems.
110	INVALID AMOUNT	The transaction amount is unusual for the type of merchant requesting payment authorization.
114	CARD DOES NOT SUPPORT THE TYPE OF OPERATION REQUESTED	Operation not allowed for that card type.
116	INSUFFICIENT BALANCE	The cardholder does not have enough credit to make the payment.
118	UNREGISTERED CARD	Non-existent card or not registered by the issuing bank.
119	UNSPECIFIED REFUSAL WITHOUT REASON	Transaction denied by issuing bank due to ecommerce control
125	CARD NOT EFFECTIVE	Non-existent card or not registered by the issuing bank.
129	CVV2/CVC2 ERROR	Unique code for transactions where the 3-digit CVV2 (Visa card) or CVC2 (MasterCard card) code on the back of the card is requested. The CVV2/CVC2 code reported by the purchaser is incorrect.
167	CONTACTING THE ISSUER: SUSPECTED FRAUD	Due to a suspicion that the transaction is fraudulent, the issuing bank does not allow automatic authorization. You must phone your local Authorization Center to obtain manual approval.
180	CARD NOT INCLUDED IN THE SERVICE	Operation not allowed for that card type.
181 - 182	CARD WITH DEBIT OR CREDIT RESTRICTIONS	Card temporarily blocked by the issuing bank.
184	AUTHENTICATION ERROR	Exclusive code for Verified by Visa or MasterCard SecureCode transactions. The transaction has been denied because the issuing bank could not properly authenticate the cardholder.
190	UNSPECIFIED REFUSAL WITHOUT REASON	Transaction denied by the issuing bank but without giving details of the reason.
191	WRONG EXPIRY DATE	Transaction denied because the expiry date of the card that has been reported in the payment does not match the card currently in force.

RESPONSE CODES TO INDICATE THAT THE TRANSACTION HAS BEEN DENIED AND THAT, IN ADDITION, THE ISSUING BANK CONSIDERS THE CARD TO BE IN A SITUATION OF POSSIBLE FRAUD AND THEREFORE REQUESTS THAT IT BE PHYSICALLY RETAINED OR VIRTUALLY ACTIVATED IN "BLACK LISTS".		
CODE	**BRIEF DESCRIPTION**	**COMMENT**
201	EXPIRED CARD	Transaction denied because the expiry date of the card that has been reported in the payment is earlier than the current effective date. In addition, the issuing bank considers the card to be in a situation of possible fraud and therefore requests that it be physically retained or virtually activated in "black lists".
202	CARD TEMPORARILY BLOCKED OR UNDER SUSPICION OF FRAUD	Card temporarily blocked by the issuing bank or under suspicion of fraud. In addition, the issuing bank considers the card to be in a situation of possible fraud and therefore requests that it be physically retained or virtually activated in "black lists".
204	OPERATION NOT PERMITTED	Operation not allowed for that card type. In addition, the issuing bank considers the card to be in a situation of possible fraud and therefore requests that it be physically retained or virtually activated in "black lists".

RESPONSE CODES TO INDICATE THAT THE TRANSACTION HAS BEEN DENIED AND THAT, IN ADDITION, THE ISSUING BANK CONSIDERS THE CARD TO BE IN A SITUATION OF POSSIBLE FRAUD AND THEREFORE REQUESTS THAT IT BE PHYSICALLY RETAINED OR VIRTUALLY ACTIVATED IN "BLACK LISTS".		
CODE	**BRIEF DESCRIPTION**	**COMMENT**
207	CONTACTING THE ISSUER	The issuing bank does not allow automatic approval. You must phone your local Authorization Center to obtain manual approval. In addition, the issuing bank considers the card to be in a situation of possible fraud and therefore requests that it be physically retained or virtually activated in "black lists".
208 - 209	LOST OR STOLEN CARD	Card blocked by the issuing bank because the cardholder has told you that it has been lost or stolen. In addition, the issuing bank considers the card to be in a situation of possible fraud and therefore requests that it be physically retained or virtually activated in "black lists".
280	CVV2/CVC2 ERROR	Unique code for transactions where the 3-digit CVV2 (Visa card) or CVC2 (MasterCard card) code on the back of the card is requested. The CVV2/CVC2 code reported by the purchaser is incorrect. In addition, the issuing bank considers the card to be in a situation of possible fraud and therefore requests that it be physically retained or virtually activated in "black lists".

290	UNSPECIFIED REFUSAL WITHOUT REASON	Transaction denied by the issuing bank but without giving details of the reason. In addition, the issuing bank considers the card to be in a situation of possible fraud and therefore requests that it be physically retained or virtually activated in "black lists".

RESPONSE CODES REFERRING TO CANCELLATION OR PARTIAL REVERSAL (Ds_Merchant_TransactionType = 3)		
CODE	**BRIEF DESCRIPTION**	**COMMENT**
400	CANCELLATION ACCEPTED	Cancellation or partial reversal transaction accepted by the issuing bank.
480	ORIGINAL OPERATION NOT FOUND OR TIME-OUT EXCEEDED	The cancellation or partial reversal has not been accepted because the original transaction has not been located, or because the issuing bank has not responded within the predefined time-out.
481	CANCELLATION ACCEPTED	Cancellation or partial reversal transaction accepted by the issuing bank. However, the response from the issuing bank has been received very late, outside of the predefined time-out period.

RESPONSE CODES REFERRING TO PRE-AUTHORIZATION OR PRE-AUTHENTICATION RECONCILIATIONS(Ds_Merchant_TransactionType = 2 o 8)		
CODE	**BRIEF DESCRIPTION**	**COMMENT**
500	RECONCILIATION ACCEPTED	The reconciliation transaction has been accepted by the issuing bank.
501 - 503	ORIGINAL OPERATION NOT FOUND OR TIME-OUT EXCEEDED	The reconciliation has not been accepted because the original transaction has not been located, or because the issuing bank has not responded within the predefined time-out.

RESPONSE CODES REFERRING TO DEFERRED PRE-AUTHORIZATIONS OR RECURRING DEFERRED PRE-AUTHENTICATION RECONCILIATIONS (Ds_Merchant_TransactionType = O or R)		
CODE	**BRIEF DESCRIPTION**	**COMMENT**
9928	PRE-AUTHORIZATION CANCELLATION CARRIED OUT BY THE SYSTEM	The system has canceled the deferred pre-authorization after more than 72 hours.
9929	PRE-AUTHORIZATION CANCELLATION CARRIED OUT BY THE MERCHANT	The cancellation of the pre-authorization has been accepted

Response codes sent by BBVA's own payment platform

RESPONSE CODES TO INDICATE THAT THE TRANSACTION HAS BEEN REFUSED BY THE BBVA PAYMENT PLATFORM		
CODE	**BRIEF DESCRIPTION**	**COMMENT**

904	MERCHANT NOT REGISTERED IN FUC	There is a problem in the merchant code configuration. Contact BBVA to solve it.
909	SYSTEM ERROR	Error in the stability of the BBVA payment platform or the Visa or MasterCard exchange systems.
912	ISSUER UNAVAILABLE	The issuing bank's approval center is not currently operational.
913	DUPLICATE TRANSMISSION	A transaction with the same order number has recently been processed (Ds_Merchant_Order).
916	AMOUNT TOO SMALL	It is not possible to operate with this amount.
928	TIME-OUT EXCEEDED	The issuing bank does not respond to the authorization request within the predefined timeout.
940	PREVIOUSLY CANCELED TRANSACTION	A request is being made to cancel or partially reverse a transaction that was previously canceled.
941	AUTHORIZATION TRANSACTION ALREADY CANCELED BY AN EARLIER CANCELLATION	Confirmation is being requested of a transaction with a order number (Ds_Merchant_Order) which corresponds to a previously canceled operation.
942	ORIGINAL AUTHORIZATION TRANSACTION DENIED	Confirmation is being requested of a transaction with a order number (Ds_Merchant_Order) which corresponds to a previously denied operation.
943	DIFFERENT ORIGINAL TRANSACTION DATA	An erroneous confirmation is being requested.
RESPONSE CODES TO INDICATE THAT THE TRANSACTION HAS BEEN REFUSED BY THE BBVA PAYMENT PLATFORM		
CODE	**BRIEF DESCRIPTION**	**COMMENT**
944	ERRONEOUS SESSION	A request is being made to open a third session. Only two sessions are allowed to be open in the payment process (current and previous sessions pending closure).
945	DUPLICATE TRANSMISSION	A transaction with the same order number has recently been processed (Ds_Merchant_Order).
946	OPERATION TO BE CANCELED IN PROCESS	A request has been made to cancel or partially reverse an original transaction that is still in process and pending a response.
947	DUPLICATE TRANSMISSION IN PROCESS	You are trying to process a transaction with the same order number (Ds_Merchant_Order) as another transaction that is still to be answered.
949	INOPERATIVE TERMINAL	The merchant's number (Ds_Merchant_MerchantCode) or the terminal's number (Ds_Merchant_Terminal) are not registered or operational.
950	RETURN NOT PERMITTED	The return is not permitted by regulation.
965	REGULATORY VIOLATION	Violation of Visa or MasterCard Regulations
9064	WRONG CARD LENGTH	No. of incorrect card positions
9078	THERE IS NO PAYMENT METHOD	The payment types defined for the terminal (Ds_Merchant_Terminal) for which the transaction is processed do not allow payment with the type of card reported.
9093	CARD DOES NOT EXIST	Card does not exist.
9094	REFUSAL OF ISSUERS	Operation denied by international issuers
9104	SECURE OPER. IS NOT POSSIBLE	Merchant with mandatory authentication and cardholder without secure purchase code

9126	DUPLICATE PROCESSING	The same operation has been processed previously
9142	PAYMENT TIME EXCEEDED	Time limit for payment exceeded
9216	WRONG CVV CODE	The CVV code has more than 3 digits
9218	SECURE OPERATIONS CANNOT BE PERFORMED	The Operations input does not allow Secure operations
9253	ERRONEOUS CHECK-DIGIT	Card does not comply with the check-digit (position 16 of the card number calculated according to Luhn's algorithm).
9256	PRE-AUTHORIZATION NOT AUTHORIZED IN MERCHANT	Merchant cannot preauthorize
9257	PREAUT. NOT ENABLED FOR THE CARD	Card is not enabled to process Preauthorizations
9261/9282	OPERATION EXCEEDS BBVA ALERT	Transaction exceeds any limit set by BBVA
9281	SURPASSES BLOCKING ALERTS	Operation exceeds blocking alerts, cannot be processed
9283	SURPASSES BLOCKING ALERTS	Operation exceeds blocking alerts, cannot be processed
9334	OPERATION EXCEEDS BBVA ALERT	Transaction exceeds any limit set by BBVA
9454	PAYMENT METHOD UNAVAILABLE	AMEX card operating in merchant without active AMEX PAYMENT
9456	PAYMENT METHOD UNAVAILABLE	VbVisa card operating in merchant without active VbVisa PAYMENT
RESPONSE CODES TO INDICATE THAT THE TRANSACTION HAS BEEN REFUSED BY THE BBVA PAYMENT PLATFORM		
CODE	**BRIEF DESCRIPTION**	**COMMENT**
9912	ISSUER UNAVAILABLE	The issuing bank's approval center is not currently operational.
9913	ERROR IN CONFIRMATION	Error in the confirmation sent by the merchant to the Virtual POS (only applicable in SOAP synchronization option)
9914	"KO" CONFIRMATION	"KO" confirmation of the merchant (applicable only in SOAP synchronization option)
9915	PAYMENT CANCELED	The user has canceled the payment
9928	DEFERRED AUTHORIZATION CANCELED	Cancellation of deferred authorization carried out by SIS (batch process)
9929	DEFERRED AUTHORIZATION CANCELED	Cancellation of deferred authorization carried out by the merchant
9997	SIMULTANEOUS DUAL OPERATION	There is a previous operation being processed with the same card number simultaneously.
9998	OPERATION STATUS: REQUESTED	Temporary status while the transaction is being processed. When the operation is complete, this code will change. If the customer closes the browser before the gateway appears, this will be the error code that will appear in the module.
9999	OPERATION STATUS: AUTHENTICATING	Temporary status while the POS is authenticated by the cardholder. Once this process is completed, the POS will assign a new code to the operation. If the customer closes the browser before receiving a response from the authentication, this will be the error code that will appear in the module.

5. INTEGRATION ERROR CODES

This section provides a glossary of errors that may occur in the integration process.

ERROR	DESCRIPTION
SIS0007	Error when removing the input XML
SIS0008	Ds_Merchant_MerchantCode error fault
SIS0009	Formatting error in Ds_Merchant_MerchantCode
SIS0010	Ds_Merchant_Terminal error fault
SIS0011	Formatting error in Ds_Merchant_Terminal
SIS0014	Formatting error in Ds_Merchant_Order
SIS0015	Ds_Merchant_Currency error fault
SIS0016	Formatting error in Ds_Merchant_Currency
SIS0017	No operations in pesetas allowed error
SIS0018	Ds_Merchant_Amount error fault
SIS0019	Formatting error in Ds_Merchant_Amount
SIS0020	Ds_Merchant_MerchantSignature error fault
SIS0021	Ds_Merchant_MerchantSignature is empty error
SIS0022	Formatting error in Ds_Merchant_TransactionType
SIS0023	Ds_Merchant_TransactionType unknown error
SIS0024	Ds_Merchant_ConsumerLanguage has more than 3 positions error
SIS0025	Formatting error in Ds_Merchant_ConsumerLanguage
SIS0026	Merchant/terminal sent does not exist error
SIS0027	Currency sent by the merchant is different from the one assigned to that terminal error.
SIS0028	Merchant / terminal has been canceled error
SIS0030	In a card payment a type of operation that is neither a payment nor pre-authorization has occurred error
SIS0031	Payment method undefined
SIS0033	In a mobile payment a type of operation that is neither a payment nor pre-authorization has occurred error
SIS0034	Database access error
SIS0037	Invalid phone number
SIS0038	Java error
SIS0040	Merchant/terminal does not have any payment method assigned error
SIS0041	Error in calculating the merchant's data HASH.
SIS0042	Signature sent is not correct
SIS0043	Error while performing online notification
SIS0046	Card BIN is not registered
SIS0051	Repeated order number error
SIS0054	No operation on which to make the return error
SIS0055	More than one payment with the same order number error
SIS0056	The operation you want to make a refund against is not authorized
SIS0057	The amount to be refunded exceeds the permitted amount

SIS0058	Data inconsistency, in validating a confirmation
SIS0059	No operation on which to make the confirmation error
SIS0060	There is already a confirmation associated with the pre-authorization
SIS0061	The preauthorization you want to confirm is not authorized
SIS0062	The amount to be confirmed exceeds the allowed amount
SIS0063	Error. Card number unavailable
SIS0064	Error. Card number cannot have more than 19 positions
SIS0065	Error. The card number is not numeric
SIS0066	Error. Expiry month unavailable
SIS0067	Error. The expiry month is not numerical
SIS0068	Error. The expiry month is not valid
SIS0069	Error. Expiry year unavailable
SIS0070	Error. The expiry year is not numerical
SIS0071	Card expired
SIS0072	Non-cancelable operation
SIS0074	Ds_Merchant_Order error fault
SIS0075	Ds_Merchant_Order has less than 4 or more than 12 positions error
SIS0076	Ds_Merchant_Order does not have the first four numerical positions error
SIS0077	Ds_Merchant_Order does not have the first four numerical positions error. No
SIS0078	Payment method unavailable
SIS0079	Error during card payment
SIS0081	New session, the stored data has been lost
SIS0084	The value of Ds_Merchant_Conciliation is null
SIS0085	The value of Ds_Merchant_Conciliation is not numeric
SIS0086	The value of Ds_Merchant_Conciliation does not occupy 6 positions
SIS0089	The value of Ds_Merchant_ExpiryDate does not occupy 4 positions
SIS0092	The value of Ds_Merchant_ExpiryDate is null
SIS0093	Card not found in the range table
SIS0094	Card was not authenticated as 3D Secure
SIS0097	Invalid value of Ds_Merchant_CComercio field
SIS0098	Invalid value of Ds_Merchant_CVentana field
SIS0112	The type of transaction specified in Ds_Merchant_Transaction_Type is not allowed error.
SIS0113	Exception occurred in operations servlet
SIS0114	Error, called with a GET instead of a POST
SIS0115	No operation on which to pay the installment error
SIS0116	The operation on which you want to pay an installment is not a valid operation
SIS0117	The operation you want to pay an installment on is not authorized
SIS0118	The total amount of the installments has been exceeded
SIS0119	Invalid value of Ds_Merchant_DateFrecuency field
SIS0120	Invalid value of Ds_Merchant_ChargeExpiryDate field
SIS0121	Invalid value of Ds_Merchant_SumTotal field

SIS0122	Value of Ds_Merchant_DateFrecuency or Ds_Merchant_SumTotal field has incorrect formatting
SIS0123	The transaction deadline has been exceeded
SIS0124	Minimum frequency has not elapsed in a subsequent recurring payment
SIS0132	The Authorization Confirmation date cannot be more than 7 days later than that of Preauthorization.
SIS0133	The Authentication Confirmation date cannot be more than 45 days later than the Pre-Authentication date.
SIS0139	Initial recurring payment error is duplicated
SIS0142	Time for payment exceeded
SIS0197	Error when obtaining the shopping basket data in gateway operation
SIS0198	Amount exceeds the limit allowed for the merchant error
SIS0199	Number of operations exceeds the limit allowed for the merchant error
SIS0200	Accumulated amount exceeds the limit allowed for the merchant error
SIS0214	Merchant does not accept returns
SIS0216	Ds_Merchant_CVV2 has more than 3 positions error
SIS0217	Formatting error in Ds_Merchant_CVV2
SIS0218	The merchant does not allow secure transactions by entry/transactions
SIS0219	Number of card operations exceeds the limit allowed for the merchant error
SIS0220	Accumulated card amount exceeds the limit allowed for the merchant error
SIS0221	The CVV2 is required error
SIS0222	There is already a cancellation associated with the pre-authorization
SIS0223	The preauthorization which you wish to cancel is not authorized
SIS0224	The merchant does not allow cancellations due to not having an extended signature
SIS0225	No operation on which to make the cancellation error
SIS0226	Data inconsistency, in validating a cancellation
SIS0227	Invalid value of Ds_Merchant_TransactionDate field
SIS0229	The requested deferred payment code does not exist
SIS0252	Merchant does not allow credit cards to be sent
SIS0253	Card does not comply with the check-digit
SIS0254	Number of operations of the IP exceeds the limit allowed for the merchant
SIS0255	Accumulated amount for the IP exceeds the limit allowed for the merchant
SIS0256	Merchant cannot preauthorize
SIS0257	This card does not allow pre-authorization operations
SIS0258	Data inconsistency, in validating a confirmation
SIS0261	Operation stopped for overcoming restrictions on SIS entry control
SIS0270	Merchant cannot carry out deferred authorizations
SIS0274	Type of operation unknown or not allowed through this entry to the SIS
SIS0282	Operation stopped for overcoming restrictions on SIS entry control
SIS0429	Error in the version sent by merchant in Ds_SignatureVersion parameter

SIS0430	Error in decoding the Ds_MerchantParameters parameter
SIS0431	Error of the JSON object being sent encoded in the Ds_MerchantParameters parameter
SIS0432	Wrong merchant FUC error
SIS0433	Wrong merchant terminal error
SIS0434	No order number in the operation sent by the merchant error
SIS0435	Signature calculation error
SIS0444	Error occurred when accessing through an old signing system that has the HMAC SHA256 code configured
SIS0448	Error, the card used for the operation is DINERS and the merchant doesn't have the "Pago DINERS" (DINERS PAYMENT) payment method
SIS0449	Error, the payment type of Ds_TransactionType (A) is not allowed for the merchant.
SIS0450	Error, the payment type of Ds_TransactionType (A) is not allowed for the merchant for Amex cards.
SIS0452	Payment method unavailable (4B Card)
SIS0453	Error, the card used for the operation is JCB and the merchant doesn't have the "Pago JCB" (JCB Payment) payment method
SIS0454	Error, the card used for the operation is AMEX and the merchant doesn't have the "Pago Amex" (Amex Payment) payment method
SIS0455	Payment method unavailable
SIS0456	Unsecured payment method (Visa) unavailable
SIS0457	Error, "MasterCard SecureCode" payment method applied with Response [VEReq, VERes] = N with MasterCard Commercial card and the merchant does not have the "MasterCard Comercial" payment method
SIS0458	Error, "MasterCard SecureCode" payment method applied with Response [VEReq, VERes] = U with MasterCard Commercial card and the merchant does not have the "MasterCard Comercial" payment method
SIS0459	Error, "JCB Secure" payment method applied with Response [VEReq, VERes] = U and the merchant doesn't have the "Pago JCB" (JCB payment) payment method
SIS0460	Error, "AMEX SafeKey" payment method applied with Response [VEReq, VERes] = N and the merchant does not have the "Pago AMEX" (AMEX Payment) payment method
SIS0461	Error, "AMEX SafeKey" payment method applied with Response [VEReq, VERes] = U and the merchant does not have the "Pago AMEX" (AMEX Payment) payment method
SIS0462	Error, "Verified By Visa","MasterCard SecureCode","JCB Secure" or "AMEX SafeKey" payment method is applied and the operation is Host To Host
SIS0463	Error, a payment method that is not among those that the SIS does not allow to be executed is selected.
SIS0464	Error, 3DSecure authentication result is "NO_3DSECURE" with MasterCard Commercial card and the merchant does not have the "MasterCard Comercial" payment method
SIS0465	Error, 3DSecure authentication result is "NO_3DSECURE" with MasterCard Commercial card and the merchant does not have the "MasterCard Comercial" payment method
APL02 ERRORS PROCESSING XML MESSAGE	
XML0000	Miscellaneous errors in the XML-String process received

XML0001	Error in generating the DOM from the XML-String received and the DTD defined
XML0002	The "Message" element does not exist in the XML-String received error
XML0003	Error, The type of "Message" in the XML-String received has an unknown or invalid value in the request
XML0004	The element "Ds_MerchantCode" does not exist in the XML-String received error
XML0005	The element "Ds_MerchantCode" is blank in the XML-String received error
XML0006	The element "Ds_MerchantCode" has an incorrect length in the XML-String received
XML0007	The element "Ds_MerchantCode" does not have numerical formatting in the XMLString received error
XML0008	The element "Ds_Terminal" does not exist in the XML-String received error
XML0009	The element "Ds_Terminal" is blank in the XML-String received error
XML0010	The element "Ds_Terminal" has an incorrect length in the received XMLString
XML0011	The element "Ds_Terminal" does not have numerical formatting in the XMLString received error
XML0012	The element "Ds_Order" does not exist in the XML-String received error
XML0013	The element "Ds_Order" is blank in the XML-String received error
XML0014	The element "Ds_Order" has an incorrect length in the XML-String received
XML0015	The element "Ds_Order" does not have its top 4 numerical positions in XML-String received
XML0016	The element "Ds_TransactionType" does not exist in the XML-String received error
XML0017	The element "Ds_TransactionType" is empty in the XML-String received error
XML0018	The element "Ds_TransactionType" has an incorrect length in the XML-String received error
XML0019	The element "Ds_TransactionType" does not have numerical formatting in the XML-String received error
XML0020	The element "Ds_TransactionType" has an unknown or invalid value in a Transaction message error
XML0021	The "Signature" element does not exist in the XML-String received error
XML0022	The element "Signature" is empty in the XML-String received error
XML0023	Incorrect signature error
XML0024	There are no operations in TZE for the requested data
XML0025	Response XML is poorly formed error
XML0026	The "Ds_fecha_inicio" element does not exist in the XML-String received error
XML0027	Error the element "Ds_fecha_fin" does not exist in the XML-String received

6. APPENDIXES

6.1 REDIRECTION INPUT/ REQUEST DATA

In the payment request to the Virtual SIS POS, a number of mandatory and other optional data will have to be sent.

The required data for transaction management is marked as such in the Comments column of the following table:

DATA ITEM	DATA NAME	Length / Type	COMMENTS
Merchant identification:	***Ds_Merchant_MerchantCode***	***9/N.***	**Required. FUC code assigned to the merchant.**
Terminal number	***Ds_Merchant_ Terminal***	***3/N.***	**Required. Terminal number which will be assigned to it by its bank. Three is considered to be its maximum length**
Transaction type	***Ds_Merchant_ Transaction Type***	***1/N***	**Mandatory for the merchant to indicate what type of transaction it is. The possible values are:** **0 - Authorization** **1 - Preauthorization** **2 - Confirmation of pre-authorization** **3 - Automatic partial or full refund** **5 - Pre-authentication** **6 - Pre-authentication confirmation** **7 - Cancellation of pre-authorization**
Amount	***Ds_Merchant_Amount***	***12/N***	**Required. The last two positions are considered decimal, except in the case of the Yen, Chilean Pesos and Ugandan Schilling that does not use decimal points.**
Currency	***Ds_Merchant_ Currency***	***4/N.***	**Required. The numerical code of the currency must be sent according to ISO-4217, for example:** **978 euros** **840 dollars** **826 pounds** **392 yen** **4 is considered to be its maximum length**

Order Number	***Ds_Merchant_ Order***	***12 / AN.***	**Required. The first 4 digits must be numerical, for the remaining digits only use the following ASCII characters** **From 30=0 to 39 = 9** **From 65=A to 90 = Z** **From 97=a to 122 = z** **The code must be different from any previous transaction**
Merchant URL for "online" notification	***Ds_Merchant_MerchantURL***	***250/AN***	**Mandatory if the merchant has "online" notification. Merchant URL that will receive a post with the transaction data.**
Product description	***Ds_Merchant_ProductDescription***	***125/AN***	**Optional. 125 is considered to be its maximum length This field will be displayed to the cardholder in the purchase confirmation screen.**
Cardholder's full name	***Ds_Merchant_ Titular***	***60/AN***	**Optional. Its maximum length is 60 characters. This field will be displayed to the cardholder in the purchase confirmation screen.**
URLOK	***Ds_Merchant_UrlOK***	***250/AN***	**Optional: if it is sent, it will be used as URLOK ignoring the one configured in the administration module if there.**
URL KO	***Ds_Merchant_UrlKO***	***250/AN***	**Optional: if it is sent, it will be used as URLKO ignoring the one configured in the administration module if there.**
Merchant identification: merchant name	***Ds_Merchant_MerchantName***	***25/AN***	**Optional: this will be the merchant's name that will appear on the customer's ticket (optional).**
Consumer language	***Ds_Merchant_ConsumerLanguage***	***3/N.***	**Optional: Value 0 will indicate that the customer language has not been determined (optional).** **Other possible values are:** **Castilian Spanish-001** **English-002** **Catalan-003** **French-004** **German-005** **Dutch-006** **Italian-007** **Swedish-008** **Portuguese-009** **Valencian-010** **Polish-011** **Galician-012**
Total amount (recurring installment)	***Ds_Merchant_Sum Total***	***12/N.***	**Mandatory for Transaction Type 5. Represents the total sum of the installment amounts. The last two positions are considered decimals.**
Merchant data	***Ds_Merchant_MerchantData***	***1024 /AN***	**Optional for the merchant to be included in the data sent by the online response to the merchant if this option has been chosen.**
Frequency	***Ds_Merchant_DateFrecuency***	***5/ N***	**Frequency in days for recurring and deferred recurring transactions (mandatory for recurring)**
Limit date	***Ds_Merchant_ ChargeExpiryDate***	***10/ AN***	**YYYYY-MM-DD format limit date for Recurring Transactions (Required for recurring and deferred recurring transactions)**
Authorization Code	***Ds_Merchant_AuthorisationCode***	***6/N***	**Optional. Represents the authorization code required to identify a successive recurring transaction in returns of subsequent recurring operations. Mandatory for returns of recurring operations.**

Date of the subsequent recurring operation	***Ds_Merchant_ TransactionDate***	***10/ AN***	**Optional. yyyy-mm-dd format. Represents the date of the successive installment, required to identify the transaction in returns. Mandatory for returns of successive installments and deferred successive installments.**
Reference	***Ds_Merchant_Identifier***	***8/N***	**Optional. Your use is specific to the payment per Referral or 1-Click Payment.**
Group code	***Ds_Merchant_Group***	***9/N***	**Optional. Your use is specific to the payment per Referral or 1-Click Payment.**
Payment without authentication	***Ds_Merchant_DirectPayment***	***4/N***	**Optional. Your use is specific to the payment per Referral or 1-Click Payment.**

PCI compliance merchants (merchant card data capture) that want to process the purchase by 3DSecure will need to submit a payment form for the Redirection input, adding three fields in the payment request:

DS_MERCHANT_PAN	***19 / N***	**Card. Length depends on card type.**
DS_MERCHANT_EXPIRYDATE	***4 / N***	**Card expiration date. Its format is YYMM, with AA being the last two digits of the year and MM the two digits of the month.**
DS_MERCHANT_CVV2	***3-4 / N***	**CVV2 card code.**

6.2 REDIRECTION INPUT/ONLINE NOTIFICATION

We recommend the use of this method, as it allows the web store to receive transaction results, online in real time. The ONLINE Notification is configurable in the administration module, and supports various possibilities depending on the merchant's need. Both HTTP notification and mail notification have exactly the same format.

HTTP notification is a parallel and independent communication to the customer's navigation process through the Virtual POS, by means of which a POST with the data of the result of the operation is sent to the merchant. Obviously, on the merchant's server side, there must be a process that collects this response and performs the tasks necessary to manage orders. To do this, you will have to provide, as a parameter, a URL where you can receive these responses in the web form that you send when you make the authorization request (see the **Ds_Merchant_MerchantURL** field in "Datos del formulario de pago" (Payment form details)). This URL will be a CGI, Servlet, etc. developed in the language that the merchant considers appropriate to integrate in its Server (C, Java, Perl, PHP, ASP, etc.), capable of interpreting the response sent by the Virtual POS. You can specify a different URL for operations with OK result and another for KOs.

NOTE: This same data will be incorporated in the OK URL (Ds_Merchant_UrlOK) or KO URL (Ds_Merchant_UrlKO) if the merchant has parameter sending enabled in the response redirection.

DATA ITEM	DATA NAME	LENGTH/TYPE	COMMENTS
Date	***Ds_Date***	***mm/dd/yyyy***	**Transaction date**
Time	***Ds_Hour***	***HH:mm***	**Transaction time**
Amount	***Ds_Amount***	***12 / N.***	**Same value as in the petition.**
Currency	***Ds_Currency***	***4 / N.***	**Same value as in the petition. 4 is considered to be its maximum length.**
Order number	***Ds_Order***	***12 / AN.***	**Same value as in the petition.**
Merchant identification: FUC code	***Ds_MerchantCode***	***9 / N.***	**Same value as in the petition.**
Terminal	***Ds_Terminal***	***3 / N.***	**Terminal number which will be assigned to it by its bank. 3 is considered to be its maximum length.**
Response code	***Ds_Response***	***4 / N.***	**See table below (Possible Ds_Response values).**
Merchant data	***Ds_MerchantData***	***1024 / AN***	**Optional information sent by the merchant in the payment form.**
Secure Payment	***Ds_SecurePayment***	***1 / N.***	0 **- If the payment is NOT secure** 1 **- If the payment is secure**
Type of transaction	***Ds_ Transaction Type***	***1 / AN***	**Type of transaction which was sent in the payment form**
Holder's country	**Ds_Card_Country**	**3/N**	**Optional: Country of issue of the card with which payment has been attempted. The following link shows the country codes and their correspondence:** http://unstats.un.org/unsd/methods/m49/m49alpha.htm
Authorization code	**Ds_AuthorisationCode**	**6/ AN**	**Optional: Alphanumeric authorization code assigned to the approval of the transaction by the authorizing institution.**
Consumer language	**Ds_ConsumerLanguage**	**3 / N**	**Optional: Value 0 will indicate that the customer language has not been determined.** **3 is considered to be its maximum length.**
Card Type	**Ds_Card_Type**	**1 / AN**	**Optional: Possible values: C - Credit D - Debit**
Card Brand	**Ds_Card_Brand**	**2 / Núm**	**Optional: A closed validation should not be done on these values, as they can vary and / or add new ones. Possible values:** **1 – VISA** **2 –MASTERCARD** **8 – AMEX** **9 – JCB** **22 – UPI** **6 – DINERS** **22 – CUP** **7 - PRIVATE**

These are the possible values of the Ds_Response or "Código de respuesta" (Response code):

CODE	MEANING
0000 to 0099	Transaction authorized for payments and pre-authorizations
900	Transaction authorized for returns and confirmations
400	Authorized Transaction for cancellations
101	Card expired
102	Card in transitional exception or suspicion of fraud
106	Exceeded PIN attempts
125	Card not effective
129	Security code (CVV2/CVC2) incorrect
180	Card not included in the service
184	Cardholder authentication error
190	Refusal of issuer without specifying reason
191	Wrong expiration date
202	Card in transitional exception or suspicion of fraud with card withdrawal
904	Merchant not registered in FUC
909	System error
913	Repeated order
944	Incorrect Session
950	Return operation not allowed
9912/912	Issuer unavailable
9064	Incorrect number of card positions
9078	Type of operation not allowed for that card
9093	Non-existent card
9094	International servers rejection
9104	Merchant with "titular seguro" (secure cardholder) and cardholder without secure purchase code
9218	The merchant does not allow secure op. by entry/transactions
9253	Card does not comply with the check-digit
9256	Merchant cannot preauthorize
9257	This card does not allow pre-authorization operations
9261	Operation stopped for overcoming restrictions on SIS entry control
9913	Error in the confirmation sent by the merchant to the Virtual POS (only applicable in SOAP synchronization option)
9914	"KO" confirmation of the merchant (applicable only in SOAP synchronization option)
9915	Payment has been canceled at the user's request
9928	Cancellation of deferred authorization carried out by SIS (batch process)
9929	Cancellation of deferred authorization carried out by the merchant
9997	Another transaction is being processed in SIS with the same card
9998	Card data request operation being processed
9999	Operation that has been redirected to the sender for authentication

These response codes are displayed in the "Código de respuesta" (Response code) field of the operation query, as long as the operation is not authorized, as shown in the following image:

Página 1 de 3

Sesión / Fecha Totales	Fecha Hora	Tipo Operación Num. Pedido	Resultado NºAutorización o Cod.Respuesta	Importe	Neto Lote/Cajón
01-10-15	01-10-2015 16:50:16	Autorización Tradicional 151001165015	Sin Finalizar 9997		
01-10-15	01-10-2015 16:50:23	Autorización Tradicional 151001165022	Autorizada 581956	1,00 EUR	2 /

Net Lot/Box

6.3 REDIRECTION INPUT / ON-SOAP NOTIFICATION

The SOAP service to be published by merchants must have the following features:

1. The service must be called 'InotificacionSIS' and offer a method called 'procesaNotificacionSIS'. This method will be defined with an XML string input parameter and another output parameter of the same type. For more information, a WSDL file is attached from which the server skeleton can be built and which will serve to define the types of data to be exchanged between client and server, in order to facilitate communication.

2. The format of the messages to be exchanged in this service must conform to the following dtd:

    a. Notification message sent from the SIS with the relevant operation data:

```
<!ELEMENT Message (Request, Signature)>

<!ELEMENT Request (Fecha, Hora, Ds_SecurePayment, Ds_Amount, Ds_Currency, Ds_Order,
Ds_MerchantCode, Ds_Terminal, Ds_Response, Ds_MerchantData?, Ds_Card_Type?, DS_Card_Type?,
Ds_TransactionType, Ds_ConsumerLanguage, Ds_ErrorCode?, Ds_CardCountry?, Ds_AuthorisationCode?)>

<!ATTLIST Request Ds_Version CDATA #REQUIRED>
<!ELEMENT Fecha (#PCDATA)>
<!ELEMENT Hora (#PCDATA)>
<!ELEMENT Ds_SecurePayment (#PCDATA)>
<!ELEMENT Ds_Amount (#PCDATA)>
```

```
<!ELEMENT Ds_Currency (#PCDATA)>
<!ELEMENT Ds_Order (#PCDATA)>
<!ELEMENT Ds_MerchantCode (#PCDATA)>
<!ELEMENT Ds_Terminal (#PCDATA)>
<!ELEMENT Ds_Response (#PCDATA)>
<!ELEMENT Ds_MerchantData (#PCDATA)>
<!ELEMENT Ds_Card_Type (#PCDATA)>
<!ELEMENT Ds_TransactionType (#PCDATA)>
<!ELEMENT Ds_ConsumerLanguage (#PCDATA)>
<!ELEMENT Ds_ErrorCode (#PCDATA)>
<!ELEMENT Ds_CardCountry (#PCDATA)>
<!ELEMENT Ds_AuthorisationCode (#PCDATA)>
<!ELEMENT Signature (#PCDATA)>
<!ELEMENT DS_Card_Type (#PCDATA)>
```

To generate the Signature field value in the merchant's response message, we will apply a HMAC SHA-256 from the string <Request...>...</Request>.

For instance:
It may be the following message:

```
<Message>
  <Request Ds_Version="0.0">
  <Fecha>01/04/2003</Fecha>
  <Hora>16:57</Hora>
  <Ds_SecurePayment>1</Ds_SecurePayment>
  <Ds_Amount>345</Ds_Amount>
  <Ds_Currency>978</Ds_Currency>
  <Ds_Order>165446</Ds_Order>
  <Ds_MerchantCode>999008881</Ds_MerchantCode>
  <Ds_Terminal>001</Ds_Terminal>
  <Ds_Card_Country>724</Ds_Card_Country>
  <Ds_Response>0000</Ds_Response>
  <Ds_MerchantData>Alfombrilla para raton</Ds_MerchantData>
  <Ds_Card_Type>C</Ds_Card_Type>
  <Ds_TransactionType>1</Ds_TransactionType>
  <Ds_ConsumerLanguage>1</Ds_ConsumerLanguage>
    </Request>
    </Message>
```

Merchant's response message to the notification:

For instance:

```
<!ELEMENT Message (Response, Signature)>
<!ELEMENT Response (Ds_Response_Merchant)>
<!ATTLIST Response Ds_Version CDATA #REQUIRED>
<!ELEMENT Ds_Response_Merchant (#PCDATA)>
<!ELEMENT Signature (#PCDATA)>
```

The possible values that can be taken from the Ds_Response_Merchant label are:

- OK' when the notification has been successfully received.
- •'KO' when an error has occurred.

To generate the Signature field value in the merchant's response message, we will apply a HMAC SHA-256 from the string <Response>...</Response>.

Examples of messages exchanged in a notification with SOAP Synchronization:

Notification message sent from SIS:

```
<Message>
    <Request Ds_Version="0.0">
            <Fecha>01/04/2003</Fecha>
            <Hora>16:57</Hora>
            <Ds_SecurePayment>1</Ds_SecurePayment>
            <Ds_Amount>345</Ds_Amount>
            <Ds_Currency>978</Ds_Currency>
            <Ds_Order>165446</Ds_Order>
            <Ds_Card_Type>C</Ds_ Card_Type >
            <Ds_MerchantCode>999008881</Ds_MerchantCode>
            <Ds_Terminal>001</Ds_Terminal>
            <Ds_Card_Country>724</Ds_Card_Country>
            <Ds_Response>0000</Ds_Response>
            <Ds_MerchantData>Alfombrilla para raton</Ds_MerchantData>
            <Ds_TransactionType>1</Ds_TransactionType>
            <Ds_ConsumerLanguage>1</Ds_ConsumerLanguage>
    </Request>
<Signature>I3gacbQMEvUYN59YiHkiml-crEMwFAeogI1jlLBDFiw=</Signature>
</Message>
```

Response message from merchant to SIS:

```
<Message>
<Response Ds_Version="0.0">
<Ds_Response_Merchant>OK</Ds_Response_Merchant>
</Response>
<Signature>d/VtqOzNlds9MTL/QO12TvGDNT+yTfawFlg55ZcjX9Q=</Signature>
</Message>
```

WSDL for the InotificacionSIS service

Merchants wishing to develop a SOAP service must comply with this WSDL. From it and, by means of automatic code generation tools, the SOAP server skeleton can be developed quickly and conveniently.

The WSDL to be met by the SOAP service developed by the customer is as follows:

```
<?xml version="1.0" encoding="UTF-8"?>

<definitions name="InotificacionSIS" targetNamespace=https://sis.SERMEPA.es/sis/InotificacionSIS.wsdl
xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:tns="https://sis.SERMEPA.es/sis/InotificacionSIS.wsdl"
xmlns:soapenc="http://schemas.xmlsoap.org/soap/encoding/"
xmlns:soap="http://schemas.xmlsoap.org/wsdl/soap/"  xmlns="http://schemas.xmlsoap.org/wsdl/">

<message name="procesaNotificacionSISRequest">
  <part name="XML" type="xs:string"/>
</message>

<message name="procesaNotificacionSISResponse">
  <part name="return" type="xs:string"/>
</message>

<portType name="InotificacionSISPortType">
  <operation name="procesaNotificacionSIS">
    <input  message="tns:procesaNotificacionSISRequest"/>
    <output  message="tns:procesaNotificacionSISResponse"/>
  </operation>
</portType>

<binding name="InotificacionSISBinding" type="tns:InotificacionSISPortType">
  <soap:binding style="rpc" transport="http://schemas.xmlsoap.org/soap/http"/>
  <operation name="procesaNotificacionSIS">
    <soap:operation soapAction="urn:InotificacionSIS#procesaNotificacionSIS"
style="rpc"/>
    <input>
      <soap:body use="encoded" encodingStyle=http://schemas.xmlsoap.org/soap/encoding/
namespace="InotificacionSIS"/>
    </input>
    <output>
      <soap:body use="encoded" encodingStyle=http://schemas.xmlsoap.org/soap/encoding/
namespace="InotificacionSIS"/>
    </output>
  </operation>
</binding>

<service name="InotificacionSISService">
  <port name="InotificacionSIS" binding="tns:InotificacionSISBinding">
    <soap:address  location="http://localhost/WebServiceSIS/InotificacionSIS.asmx"/>
  </port>
</service>
    </definitions>
```

6.4 REDIRECTION INPUT / 1-CLICK PAYMENT

Payment request

A payment is made and the parameter Ds_Merchant_Identifier is added with the value REQUIRED for a new Token to be generated and returned in the notification along with the expiry date. The Token will be associated with the merchant indicated by the parameter Ds_Merchant_MerchantCode.

Data to be sent in the petition:

In this scenario, merchant performs two different integrations with the Virtual POS and this is possibly the most common way of using it.

Ds_Merchant_MerchantCode=999008881

Ds_Merchant_Terminal=1

Ds_Merchant_Currency=978

Ds_Merchant_TransactionType=0

Ds_Merchant_Amount=100

Ds_Merchant_Order=112545

Ds_Merchant_Identifier=REQUIRED

Ds_Merchant_MerchantURL=<Notification URL>

Response in online notification

If the operation is authorized, the Online notification and the OK URL will include the new parameters with the Token value generated and the card expiry as the Token expiry.

Ds_Merchant_Identifier=a091f0f9f0aaf0506930dda4a6974f1df4a0d9c1

Ds_ExpiryDate=1412

Payment with token

A payment is made and the parameter Ds_Merchant_Identifier is added with the value of the Token to be used. In this case, the parameter Ds_Merchant_DirectPayment could be used with 'true' value and it would act in the same way.

Data to be sent in the petition:

Ds_Merchant_MerchantCode=999008881

Ds_Merchant_Terminal=1

Ds_Merchant_Currency=978

Ds_Merchant_TransactionType=0

Ds_Merchant_Amount=100

Ds_Merchant_Order=112546 <Does not have to be the same as the original operation>

Ds_Merchant_Identifier=a091f0f9f0aaf0506930dda4a6974f1df4a0d9c1
Ds_Merchant_MerchantURL=<Notification URL>

There is a specific manual with more detailed information about managing 1-click payments. You can request it from your ecommerce manager.

6.5 WEB SERVICE INOUT / PAYMENT REQUEST

The following is a description of the data and their characteristics required to send a request to the BBVA Web Service in XML format. An example of how to use this data in payment request messages is also included.

Data name	Length/Type	Descripción
DS_MERCHANT_AMOUNT	*12 / N*	Required. The last two positions are considered decimal, except in the case of the Yen, Chilean Pesos and Ugandan Schilling that does not use decimal points.
DS_MERCHANT_ORDER	*12 / AN*	Required. Order number. The first 4 digits must be numeric. Each order is unique, it cannot be repeated.
DS_MERCHANT_MERCHANTCODE	*9 / N*	Required. FUC code assigned to the merchant.
DS_MERCHANT_TERMINAL	*3 / N*	Required. Terminal number which will be assigned to it by its bank. Default value "001". 3 is considered to be its maximum length.
DS_MERCHANT_CURRENCY	*4 / N*	Required. Merchant's currency. This has to be the currency contracted for the terminal. Value 978 Euros, 840 Dollars, 826 Pounds Sterling and 392 Yen.
DS_MERCHANT_PAN	*19 / N*	Required. Card. Length depends on card type.
DS_MERCHANT_EXPIRYDATE	*4 / N*	Required. Card expiration date. Its format is YYMM, with AA being the last two digits of the year and MM the two digits of the month.
DS_MERCHANT_CVV2	*3-4 / N*	Required. CVV2 card code.

DS_MERCHANT_TRANSACTIONTYPE	***1 / AN***	**Required. Field for the merchant to indicate what type of transaction it is. The possible values are: 0 - Authorization 1 - Preauthorization**
Type A: ASCII characters from 65 = A to 90 = Z and from 97 = a to 122 = z. Type N: ASCII characters from 30 = 0 to 39 = 9.		

The following is an example of a payment request message:

```
<DATOSENTRADA>
<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_PAN>XXXXXXXXXXXXX</DS_MERCHANT_PAN>
<DS_MERCHANT_CVV2>XXX</DS_MERCHANT_CVV2>
<DS_MERCHANT_TRANSACTIONTYPE>0</DS_MERCHANT_TRANSACTIONTYPE>
<DS_MERCHANT_TERMINAL>999</DS_MERCHANT_TERMINAL>
<DS_MERCHANT_EXPIRYDATE>XXXX</DS_MERCHANT_EXPIRYDATE>
< DS_MERCHANT_IDENTIFIER>REQUIRED</ DS_MERCHANT_IDENTIFIER>

</DATOSENTRADA>
```

6.6 WEB SERVICE INOUT / Confirmation/Refund Requests

The following is a description of the data and their characteristics required to send a request to the BBVA WebService in XML format. An example of how to use this data in payment request messages is also included.

Data name	Length/Type	Descripción
DS_MERCHANT_AMOUNT	***12 / N***	**Required. The last two positions are considered decimal, except in the case of the Yen, Chilean Pesos and Ugandan Schilling that does not use decimal points.**
DS_MERCHANT_ORDER	***12 / AN***	**Required. Order number. The first 4 digits must be numeric. Each order is unique, it cannot be repeated.**
DS_MERCHANT_MERCHANTCODE	***9 / N***	**Required. FUC code assigned to the merchant.**
DS_MERCHANT_TERMINAL	***3 / N***	**Required. Terminal number which will be assigned to it by its bank. Default value "001". 3 is considered to be its maximum length.**

DS_MERCHANT_CURRENCY	***4 / N***	**Required. Merchant's currency. This has to be the currency contracted for the terminal. Value 978 Euros, 840 Dollars, 826 Pounds Sterling and 392 Yen.**
DS_MERCHANT_TRANSACTIONTYPE	***1 / AN***	**Required. Field for the merchant to indicate what type of transaction it is. The possible values are: 2 - Confirmation 3 - Automatic Refund 9 - Cancellation of preauthorization**
DS_MERCHANT_AUTHORISATIONCODE	***6 / N***	**Optional. Represents the authorization code required to identify a successive recurring transaction in returns of subsequent recurring operations. Mandatory for returns of recurring operations.**
Type A: ASCII characters from 65 = A to 90 = Z and from 97 = a to 122 = z. Type N: ASCII characters from 30 = 0 to 39 = 9.		

The following is an example of a recurring payment request message:

```
<DATOSENTRADA>
    <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>
</DATOSENTRADA>
```

6.7 WEB SERVICE INPUT / 1-CLICK PAYMENT

The following is a description of the data and their characteristics required to send a request to the BBVA Web Service in XML format. An example of how to use this data in payment request messages is also included.

Data name	Length/Type	Descripción
DS_MERCHANT_AMOUNT	*12 / N*	Required. The last two positions are considered decimal, except in the case of the Yen, Chilean Pesos and Ugandan Schilling that does not use decimal points.
DS_ME RCHANT_ORDE R	*12 / AN*	Required. Order number. The first 4 digits must be numeric. Each order is unique, it cannot be repeated.
DS_MERCHANT_MERCHANTCODE	*9 / N*	Required. FUC code assigned to the merchant.
DS_MERCHANT_TERMINAL	*3 / N*	Required. Terminal number which will be assigned to it by its bank. The default value "001 ".3 is considered its maximum length.
DS_MERCHANT_CURRENCY	*4 / N*	Required. Merchant's currency. This has to be the currency contracted for the terminal. Value 978 Euros, 840 Dollars, 826 Pounds Sterling and 392 Yen.
DS_MERCHANT_PAN	*19 / N*	Required. Card. Length depends on card type.
DS_MERCHANT_EXPIRYDATE	*4 / N*	Required. Card expiration date. Its format is YYMM, with AA being the last two digits of the year and MM the two digits of the month.
DS_MERCHANT_CVV2	*3-4 / N*	Required. CVV2 card code.
DS_MERCHANT_TRANSACTIONTYPE	*1 / AN*	Required. Field for the merchant to indicate what type of transaction it is. The possible values are: 0 - Authorization 1 - Preauthorization
DS_MERCHANT_IDENTIFIER	*8/N*	Required. Its use is specified in the examples of payment by Referral or 1-Click Payment
DS_MERCHANT_GROUP	*9/N*	Optional. Its use is specified in the examples of payment by Referral or 1-Click Payment
DS_MERCHANT_DIRECTPAYMENT	*4/N*	Optional. Its use is specified in the examples of payment by Referral or 1-Click Payment
Type A: ASCII characters from 65 = A to 90 = Z and from 97 = a to 122 = z. Type N: ASCII characters from 30 = 0 to 39 = 9.		

The following is an example of a payment request message:

```
<DATOSENTRADA>
    <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_PAN>XXXXXXXXXXXXX</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>

< DS_MERCHANT_IDENTIFIER>REQUIRED</DS_MERCHANT_IDENTIFIER>

</DATOSENTRADA>

6.8 WEB SERVICE INPUT / RESPONSE MESSAGE (H2H)

Below is a table that lists all the parameters that are part of the Web Service response.

Data name	Length/Type	Descripción
CODE		**Required**. Indicates whether the operation has been successful or not (does not indicate whether it has been authorized, only if it has been processed). An 0 indicates that the operation has been successful. If it is anything other than 0, it will have a code. (See section 8 of this Guide for error codes)
Ds_Amount	*12 / AN*	**Required**. For Euros the last two positions are considered decimal, except in the case of the Yen that does not use decimal points.
Ds_Currency	*4 / N*	**Required**. Merchant's currency.
Ds_Order	*12 /A N*	**Required**. Order number.
Ds_Signature	*40 / AN*	**Required**. Merchant's signature.
Ds_MerchantCode	*9 / N*	**Required**. FUC code associated to the merchant.
Ds_Terminal	*3 / N*	**Required**. Wrong merchant Terminal error
Ds_Response	*4 / N*	**Required**. Value indicating the result of the operation. **Indicates whether or not it has been authorized**. See table on page 52
Ds_AuthorisationCode	*6 / N*	**Optional**. Authorization code if it exists for authorized operations.
Ds_ Transaction Type	*1 / AN*	**Required.** Indicates what type of transaction was performed. The possible values are: 0 - Authorization 1 - Preauthorization 2 - Confirmation 3 - Automatic Refund 9 - Cancellation of Preauthorization
Ds_SecurePayment		**Required.** Indicates whether the payment was secure or not: • 0: secure (does not apply) • 1: unsecure.

Ds_Language	*1 / N*	Required. Language.
Type A: ASCII characters from 65 = A to 90 = Z and from 97 = a to 122 = z. Type N: ASCII characters from 30 = 0 to 39 = 9.		

6.9 WEB SERVICE INPUT / WSDL PAYMENT REQUEST

```
<?xml version="1.0" encoding="UTF-8"?>
<wsdl:definitions
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="SerClsWSEntrada">
        <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">
        <wsdlsoap:binding style="document" transport="http://schemas.xmlsoap.org/soap/http"/>
        <wsdl:operation name="trataPeticion">
            <wsdlsoap:operation soapAction=""/>
            <wsdl:input name="trataPeticionRequest">
 <wsdlsoap:body use="literal"/>
            </wsdl:input>
            <wsdl:output name="trataPeticionResponse">
 <wsdlsoap:body use="literal"/>
            </wsdl:output>
        </wsdl:operation>
        <wsdl:operation name="consultaDCC">
            <wsdlsoap:operation soapAction=""/>
            <wsdl:input name="consultaDCCRequest">
 <wsdlsoap:body use="literal"/>
            </wsdl:input>
            <wsdl:output name="consultaDCCResponse">
 <wsdlsoap:body use="literal"/>
            </wsdl:output>
        </wsdl:operation>
    </wsdl:binding>
    <wsdl:service name="SerClsWSEntradaService">
        <wsdl:port binding="intf:SerClsWSEntradaSoapBinding" name="SerClsWSEntrada">
            <wsdlsoap:address location="https://sis.redsys.es/sis/services/SerClsWSEntrada"/>
        </wsdl:port>
    </wsdl:service>
 </wsdl:definitions>
```

6.10 Payment retries

The option to retry the payment offers to the cardholder the possibility to try to make the payment, with another card or with another method of payment, when the transaction has been denied by the issuer or due to an error in the cardholder's authentication (Error = 184). The cardholder will not be offered this option of retrying the payment, in case the operation is denied by application of fraud rules, by a technical error in the authentication process or by any other type of error.

For the customer to see this option, the merchant must have a specific configuration in the POS-Virtual Administration Module (Allow Repeat Order = YES, with retries) and meet the following requirements:

- The merchant must be able to receive several notifications associated with the same order number and only take into account the first notification that the operation is authorized, or in case of not receiving any notification of authorized operation, it must take into account the last Notification that you receive about the order number in question.
- Currently, the retry option is not compatible with the payment modules provided by Redsys for Prestashop, Magento, etc. virtual stores. In payment modules external to Redsys this option may not work properly, causing an error in the updating of orders in the virtual store.

The following image shows an example of the receipt that is shown to the customer, in which the option to retry is offered: