Virtual POS

Manual Integración REST Integration Manual

Version: 3.0.3
Date: 02/07/2024
Reference: RS.TE.CEL.MAN.0037

Redsys, Servicios de Procesamiento, S.L. – c/ Francisco Sancha, 12 – 28034 Madrid (Spain)
www.redsys.es

Version control

Version	Date	Concerns	Brief description of the change
1.0	01/10/2018	ALL	First version
1.1	17/12/2018	Annex 7.7	3DSecure 2.0
1.1	25/01/2019	Annex 7.9	PSP Integration
1.2	25/01/2019	Section 7.2 Confirmation / Refund Requests	Added support for cancellation of authorization, when sending type 9 in TRANSACTIONTYPE
1.3	06.02.2019	Various points	Response code is included in operations that require challenge: *Ds_response = 9568* In the 3DSecure data, the protocol version is always indicated in the protocolVersion *field* Sequence schemes are included to clarify the flow. 7.8 PSP integration point is added
1.4	11/02/2019	Various points	All 3DSecure 1.0.2 parameters and an example are included.
1.5	18/02/2019	Various points	External MPI Move point 7.7 to point 5 Different clarifications in section 5 of 3DSecure Authentication Included timeout section
2.0	27/03/2019	Various points	Added information about EMV3DS and PSD2. Added transaction information with DCC. Added Authentication information with DCC. Modifications of the Integration section for PSP

			Modifications of the section. Do you operate by PUCE? Modifications of the EMV3DS External MPI section Added reference to the new documents TPV-Virtual GuiaErroresSIS.xlsx and TPV-Virtual Parámetros Entrada-Salida.xlsx
2.1	12/04/2019	Points 5, 9, 10, 13	Important note about EMV3DS integration Added test cards for EMV3DS Points assignment for PUCE operations The SIS errors spreadsheet is modified to include it in the spreadsheet Input-output parameters
2.2	25/07/2019	Points 5,8, 10	COF and MIT operations Error code change format Creq parameter
2.3	04/10/2019	The entire document	Adaptations are added for EVM3DS 2.2
2.4	12/11/2019	Points 5.1, 5.2, 8, 9 and 10.1.3	Advances for EMV 3DS functionalities are marked. Point 10.1.3 is reorganized. Send request flow for PSP
2.5	24/01/2020	Points 5,6,8,9,11	POINTS 5, 6 added the Ds_Card_PSD2 parameter POINTS 5, MIT and tokenization Requests with EXEMPTIONS. Clarification and examples for EMV3DS advancement functionalities. Modification of test cards for EMV3DS advances.
2.6	10/02/2020	Points 6,7 and 11	Definition of new optional parameters: “CambioBCE” and “porcentajeSobreBCE” for DCC operations with currencies of the European Union

3.0	01/10/2020	The whole document	The "advance" mark is removed from the functionalities affected by PSD2 The points in which the flow of the REST request to the Virtual POS and the request with authentication of the cardholder are restructured. Point 10 is pending of brand specifications. An Annex is created with the help libraries. Cards are added for testing with all card brands.
3.0.1	26/11/2020	Point 6 Point 6.6	IMPORTANT note is added: operation that starts flow in version 2 and in the authentication process changes the flow to version 1. A test card is added for this case. Examples of 3DS v 1.0 requests are included.
3.0.2	02/07/2024	Point 6 Point 7 Point 12	3DSecure Version 1 removed. Example text is added for the DCC BCE operation. Test Cases from 3DSecure Version 1 removed.
3.0.3	02/07/2024	Point 12	DCC test card in Pounds added

CONTENTS

1. Introduction

1.1 Purpose

This document explores the technical aspects necessary for a merchant to integrate with the Virtual POS using a REST interface.

IMPORTANT NOTE: ***On the occasion of the full entry into force of the European Payments Directive PSD2 in 2020, some new features and technical specifications that will be available throughout 2020 are included in this guide, to allow preparation of works in those cases of merchants wishing to incorporate some functionalities into their payment operations, especially in relation to the authentication and exemption management that the PSD2 considers.***

1.2 Definitions, acronyms, and abbreviations

- **SIS**. Redsys Integrated Server (Virtual POS Server).
- **SCA**. Strong Customer Authentication. Cardholder's enhanced authentication.
- **Frictionless**. Authentication without cardholder participation.
- **Challenge**. Enhanced authentication of the cardholder (through OTP, static password, biometrics, etc.).
- **PSD2**. Payment Service Providers. European regulation in digital payment services.
- **3DSecure**: Security system for online payments. Hereinafter EMV3DS
- **EMV3DS**: Acronyms to identify the new version of 3DSecure in the Virtual POS.
- **MIT**. Merchant Initiated Transaction. It refers to transactions initiated directly by the merchant without the cardholder being present, such as in the case of recurring payments.
- **COF**. Credentials On File. It refers to the operation in which the card data are stored for future use.
- **DCC.** Dynamic Currency Conversion. It allows the cardholder to make the payment in his/her own currency instead of the one defined in the terminal.

1.3 References

- Documentation for integration with the SIS
- TPV-Virtual Guía SIS.
- COF ECOM Specifications Guide.
- TPV-Virtual Parámetros Entrada-Salida.xlsx

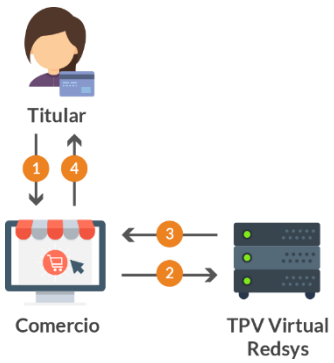
2. What does the integration with REST interface allow?

This type of connection allows merchants to have the Virtual POS integrated into their own Web application, so the entire payment process is carried out on the merchant's website and the client does not leave this web environment at any time. This connection mode also offers the possibility of authenticating the cardholder through the 3DSecure protocol, which provides greater security for purchases.

It is important to bear in mind that in this type of integration, it is the merchant that collects the client's card data to process the payment and, therefore, has an impact on its compliance with PCI-DSS.

Merchants can also use this connection mode to integrate the Virtual POS with their back office and perform associated operations such as returns or Preauthorization confirmations. In this case, it is not necessary for the merchant to process the card data, so if it only uses the REST integration for this type of operation, it will not affect PCI-DSS compliance.

The basic scheme of a payment through REST integration would be as follows:



1. The cardholder selects the products he/she wants to buy and enters the card details in a form displayed by the merchant.
2. The merchant sends the payment details to the Virtual POS.
3. Once the payment is made, the virtual POS reports the result of the operation.
4. The merchant returns the information of the payment result to the cardholder.

3. Structure of a REST request

The merchant sends a request (POST) through the REST interface to the Virtual POS. In this request, a JSON string is sent including the following parameters:

- **Ds_SignatureVersion**: Constant value indicating the signature version that is being used.
- **Ds_MerchantParameters**: String in JSON format with all the parameters of the request encoded in Base 64 and without carriage returns.
- **Ds_Signature**: Signature of the data sent. It is the result of the signature calculated following the algorithm indicated in the Ds_SignatureVersion parameter.

*NOTE: remember that to make the REST request the **Content-Type** header with the following value **"application / json"** must be sent.*

NOTE: the connection requires the use of a signature system based on HMAC SHA-256, which authenticates each other the merchant server with the Virtual POS. To develop the calculation of this type of signature, the merchant can carry out the development by itself using the standard functions of the different development environments, although to allow the developments we offer libraries (PHP, JAVA and.NET). See Annexes. These libraries are also available at the following link:

http://www.redsys.es/wps/portal/redsys/publica/areadeserviciosweb/descargaDeDocumentacionYEjecutables/

3.1 Identify the signature algorithm version to use (Ds_SignatureVersion)

The request must identify the specific version of the algorithm that is being used for the signature. Currently the value **HMAC_SHA256_V1** must be used to identify the version of all requests, so this will be the value of the **Ds_SignatureVersion parameter**.

3.2 Configure the request data string (Ds_MerchantParameters)

In the request, all the necessary parameters to identify the type of operation to be performed must be sent. All parameters will be configured in JSON format, and the name of each parameter must be indicated in uppercase or with "CamelCase" structure (For example: DS_MERCHANT_AMOUNT or Ds_Merchant_Amount).

Here is an example of the JSON object with the fields from a return request (DS_MERCHANT_TRANSACTIONTYPE = 3) before encoding it in Base 64. The examples that we will show in this document will be represented in this format to simplify their understanding:

```
{"DS_MERCHANT_MERCHANTCODE":"999008881","DS_MERCHANT_TERMINAL":"1","DS_MERCHANT_ORDER":"06080232580","DS_MERCHANT_AMOUNT":"100","DS_MERCHANT_CURRENCY":"978","DS_MERCHANT_TRANSACTIONTYPE":"3"}
```

Once the JSON chain is configured with all the fields, it is necessary to code it in BASE64 without carriage returns to ensure that it remains constant and is not altered in the submission process.

The JSON object encoded in BASE64 is shown below:

```
eyJEU19NRVJDSEFOVF9NRVJDSEFOVENPREUiOiI5OTkwMDg4ODEiLCJEU19NRVJDSEFOVF9URVJNSU5BTCI6IjEiLCJEU19NRVJDSEFOVF9PUkRFUiI6IjA2MDgwMjMyNTgwIiwiRFNfTUVSQ0hBTlRfQU1PVU5UIjoiMTAwIiwiRFNfTUVSQ0hBTlRfQ1VSUkVOQ1kiOiI5NzgiLCJEU19NRVJDSEFOVF9UUkFOU0FDVElPTlRZUEUiOiIzIn0=
```

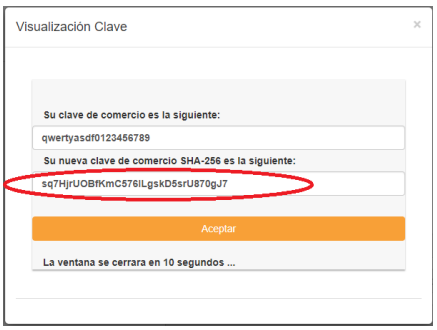
The string resulting from encoding in BASE64 will be the value of the **Ds_MerchantParameters parameter**.

NOTE 1: To use the help libraries see Annexes

NOTE 2: The complete list of all input parameters of the Virtual POS (SIS) is available in the document "TPV-Virtual Parámetros Entrada-Salida.xlsx".

3.3 Sign the request data (Ds_Signature)

To calculate the signature, it is necessary to use a specific key for each terminal. To obtain this password, you must access the Management portal, option Merchant data query, in the "View key" section you can consult this key, as shown in the following image:



IMPORTANT NOTE: This key must be saved on the merchant's server in the safest way possible to avoid its fraudulent use. ***The merchant is responsible for the proper custody and secret maintenance of this key.***

Once the data string Ds_MerchantParameters and the specific terminal key are configured, the signature must be calculated following the next steps:

1. A specific key is generated per operation. This key is obtained by encrypting the order number of the transaction (Ds_Merchant_Order) with the merchant key using 3DES encryption (CBC mode).

Example: If we start from the merchant key in base 64: sq7HjrUOBfKmC576ILgskD5srU870gJ7 and the order number is **06080232580** we have the following results:

Merchant key in hexadecimal representation:

b2aec78eb50e05f2a60b9efa20b82c903e6cad4f3bd2027b

Key of the operation in hexadecimal representation:

a5334014a4f010c8779cef789886c123

2. The HMAC SHA256 of the value of the Ds_MerchantParameters parameter in Base64 with the key obtained in the previous step is calculated.

Example: Signature value in hexadecimal format:

1a61f04e8bed872af3b4bb3b0fbec67e5725073b0035d7ab1dedf3145e898994

3. The result obtained is encoded in BASE 64, and this will be the value of the Ds_Signature parameter.

Example: Signature value in Base 64:

GmHwTovthyrztLs7D77GflclBzsANderHe3zFF6JiZQ=

NOTE: You can use the help libraries to generate the signature, see Annexes. Point 1

4. REST response structure

Once the request has been managed, the Virtual POS will respond to the merchant's server with the result information included in a JSON string.

Depending on whether the response has been processed correctly or not, two types of response will be received:

1) **Response of an operation successfully processed**

When a request has been processed correctly, once the response of the request has been received to the Virtual POS, the merchant must capture and validate the return parameters to know the result of the operation.

Any response from the Virtual POS will be a JSON string that will include the following parameters:

- **Ds_SignatureVersion**: Constant value indicating the signature version that is being used.
- **Ds_MerchantParameters**: String in JSON format with all the parameters of the response encoded in Base 64 and without carriage returns.
- **Ds_Signature**: Signature of the received data. It is the 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 have not been modified and that the actual origin is the Virtual POS. The calculation of the signature in the response is performed in the same way as in the request.**

NOTE: You can use the help libraries to validate the signature, see Annexes. Point 2

2) **Response of an operation NOT successfully processed**

When a request has not been processed correctly, the error code that will identify the reason for which the request could not be processed will be reported in JSON format.

The error that has occurred will be reported in the *errorCode* parameter, eg: **{"errorCode": "SIS0042"}**

NOTE: See list of error codes in the guide "TPV-Virtual Parámetros Entrada-Salida.xlsx", error codes section.

3) Verification of the result of the operation

The result of the operation will be reported by means of the Ds_Response or "Código de respuesta" parameter. The values of this field to indicate whether an operation has been authorized are indicated in the following table.

Ds_Response value	Description
Value from 0 to 100	Authorized operation in payments and pre-authorizations.
Value 900	Authorized operation in refunds, confirmations, and PUCE authentication
Value 400	Authorized operation in cancellations
Any other value	Check detailed list in "*TPV-Virtual Parámetros Entrada-Salida.xlsx*"

An example of an authorized payment transaction response would be the following:

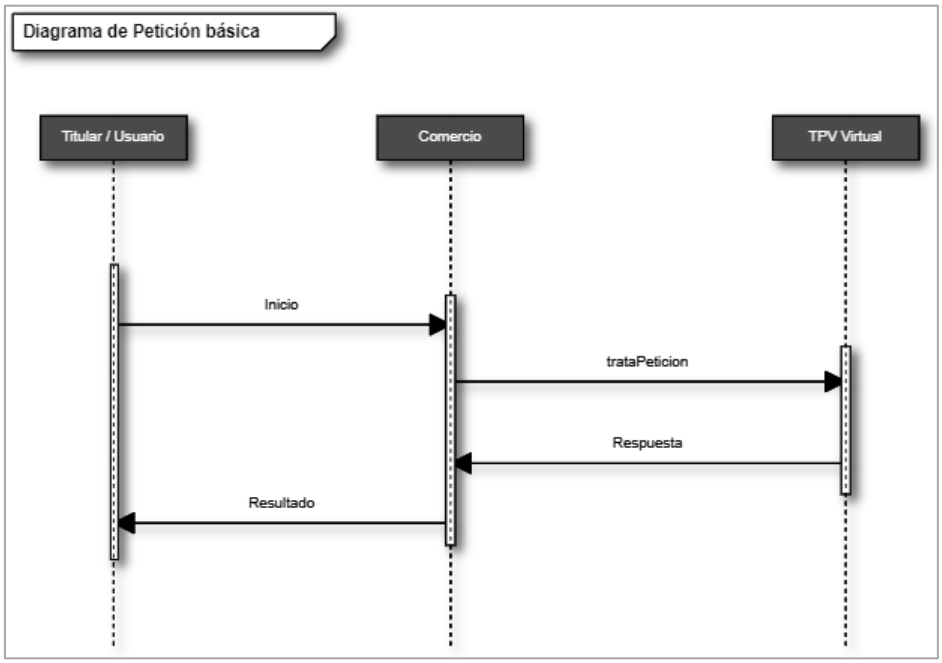
{
"Ds_Amount":"1000",
"Ds_Currency":"978",
"Ds_Order":"1552568529",
"Ds_MerchantCode":"999008881",
"Ds_Terminal":"2",
"Ds_Response":"0000",
"Ds_AuthorisationCode":"842841",
"Ds_TransactionType":"0",
"Ds_SecurePayment":"0",
"Ds_Language":"1",
"Ds_Card_Type":"C",
"Ds_Card_Country":"724",
"Ds_Card_Brand":"1"

}

NOTE 2: The complete list of all input parameters of the Virtual POS (SIS) is available in the document "TPV-Virtual Parámetros Entrada-Salida.xlsx".

5. Direct transactions (without authentication)

The following scheme presents the general flow of an operation performed through the REST input of the Virtual POS without authentication of the cardholder.



To make a request in which it is not necessary to authenticate the cardholder (ex: payment without authentication, refund, confirmation of pre-authorization, etc.), the merchant must prepare the request with the necessary parameters as indicated in point 3 and send it to the following endpoints, depending on whether it wants to make a request in the test environment or real operations:

URL Connection	Environment
https://sis-t.redsys.es:25443/sis/rest/trataPeticionREST	Tests
https://sis.redsys.es/sis/rest/trataPeticionREST	Real

NOTE: The complete list of input parameters of the Virtual POS (SIS) is available in the document "TPV-Virtual Parámetros Entrada-Salida.xlsx".

6. Transactions with 3DS v1.0 and EMV3DS authentication

6.1 Steps to perform a transaction with authentication

The REST integration interface allows payments with cardholder authentication using the 3DSecure protocol defined by the card brands. This REST integration is prepared to use the different versions of this protocol, both the 3DS v1 and the EMV 3DS 2 versions.

To perform the authentication, several additional steps must be incorporated into the payment flow, which are explained below. Depending on the version of the 3DS protocol used, this flow may vary slightly, although the main steps remain.

It is important to bear in mind that the merchant that wants to carry out authentication through REST integration must be adapted to the two versions of the 3DS protocol, since the version used in each operation will depend on the 3DS protocol version of the card involved in it.

The payment process for an operation with authentication in the REST connection follows the next steps:

- **Step 1: Initiate request**

The merchant must make a request to the Virtual POS to obtain information on the card's capacity in terms of authentication (3DS protocol version), possibility of applying exemptions, being able to carry out special operations (for example, if it allows DCC), information that will indicate how the next steps should be managed to carry out the transaction.

- **Step 2: 3DSMethod (only in EMV3DS protocol)**

If the card is associated with a 3DSMethod URL, the merchant must adapt the page in the cardholder's browser so that a direct connection is established with the issuer that allows it to capture the information of the device used by the cardholder: User-Agent, model of device, etc. For more information on this step refer to point 6.5.2.

- **Step3: Authorisation request**

The merchant, in the request for authorization of the operation, will send the result of the 3DSMethod, the additional data of the EMV3DS protocol (protocol version), as well as a possible request for SCA exemption within the framework of PSD2.

The Virtual POS will start the authentication process. In the case of an authentication in EMV3DS version, the following result can be obtained:

a. Authentication OK (Frictionless): the operation has been authenticated without requiring any action from the cardholder and the Virtual POS will continue the authorization process.

b. Challenge required: The issuing bank requires verifying the identity of the client through explicit authentication or challenge.

c. Other result: Authentication not available, authentication rejected, authentication failed, etc.

In cases A (frictionless) and C (other result), the operation ends at that point and the Virtual POS responds with the result thereof, whether authorized or denied. If the issuer's response in this step is the challenge request, the merchant must continue the flow with the following steps to complete the operation.

- **Step 4: Authentication (in case of challenge is required)**

In this step, the cardholder's browser is redirected to connect with the Issuing Bank that will verify his/her identity through an authentication, with the participation of the cardholder (challenge), this authentication will be carried out through the system requested by the Issuing Bank: OTP by SMS (One Time Password), static password, biometrics, combination of the above, etc.

- **Step 5: Authorization (validation of the challenge result)**

The merchant will send the authorization, with the result of the challenge, to the Virtual POS to finalize the authorization process.

***IMPORTANT NOTE*: The merchant must be ready to carry out any of the flows shown in the following sections, since depending on the response obtained in the "Inicia Peticion" step, the EMV3DS protocol flow must be used.**

In addition, in the case of the EMV3DS protocol, the merchant must also be ready to support both Authentication processes: Challenge (with the intervention of the cardholder) or Frictionless (without the intervention of the cardholder). The card issuer will be responsible for determining the Authentication process to be carried out.

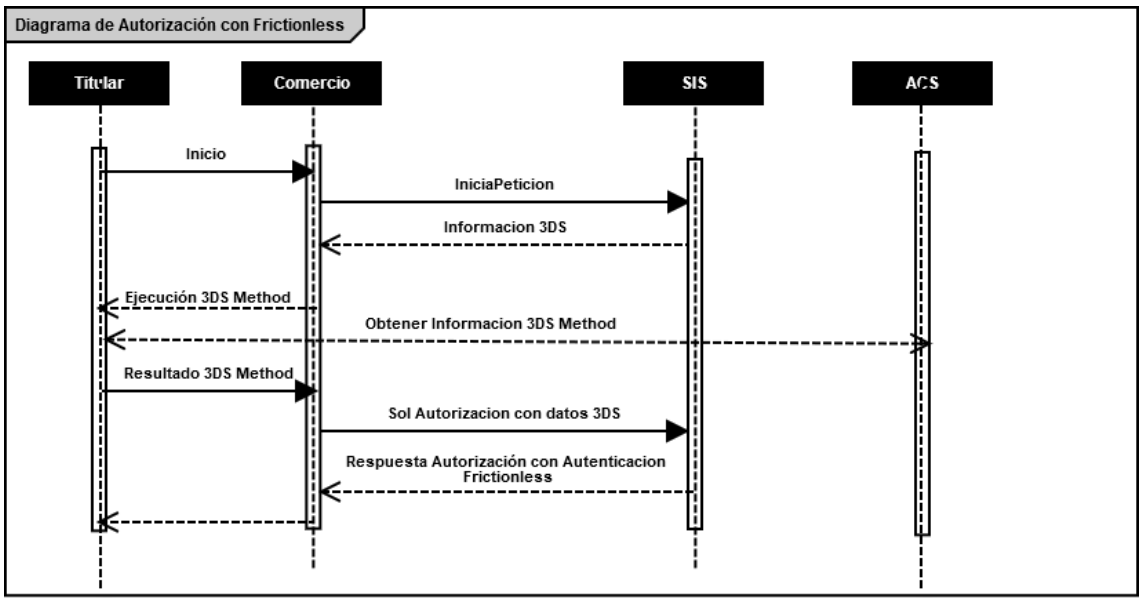
NOTE: We recommend that in step 3 (authorization request) the merchant provides all the additional information possible to help the issuer identify that the operation is being carried out by

the true cardholder. This additional information will increase the probability of a frictionless flow (authentication without cardholder intervention, thus helping to reduce the dropout rate.

The maximum time from the beginning of the request and the request for authorization is 1 hour. After this time, the request is given as lost and the flow must be carried out again from the beginning.

6.2 Authorization flow with frictionless EMV3DS authentication

The following scheme presents the general flow of an operation with frictionless authentication performed through the Virtual POS.



1. The cardholder selects the products he/she wants to buy and enters the card details in a form displayed by the merchant.
2. The merchant initiates a request by sending the data to the Virtual POS.
3. The Virtual POS checks the configuration of the card, and in the response will inform if the card supports EMV3DS authentication and the EMV3DS protocol version that is being applied.

   3.1 If the card requires it, execute the 3DSMethod: the connection is initiated from the browser to the ACS and this returns the result of the execution to the merchant.
4. The merchant sends the authorization request with a card that supports EMV3DS. In addition to the payment data, it is necessary to send the 3DSMethod result and additional data for authentication.

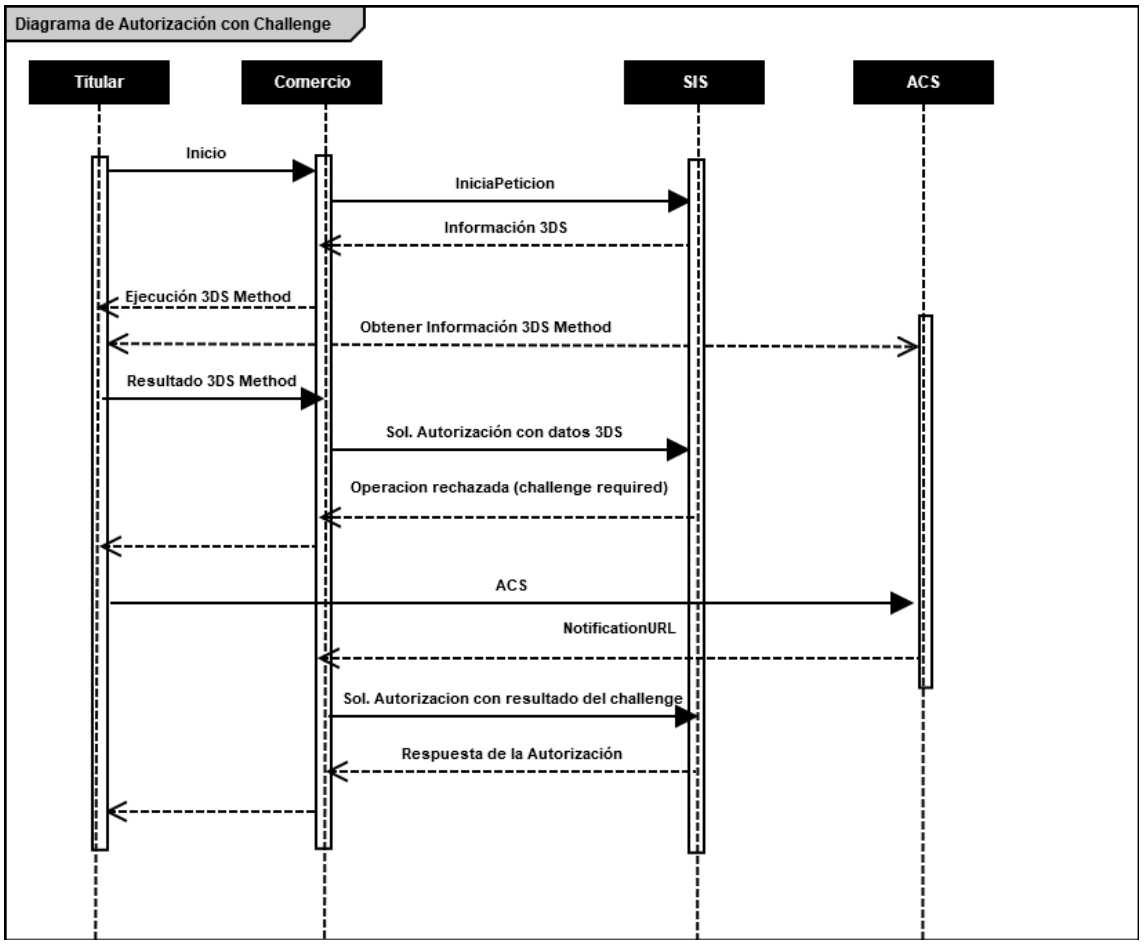
The Virtual POS initiates the authentication, and the issuer, based on the data received, authenticates the operation without the intervention of the cardholder. Then, the Virtual POS will process the authorization

5. Once the payment is made, the Virtual POS reports the result of the operation.
6. The merchant shows the result of the payment to the cardholder.

NOTE: The diagrams only consider the merchant flows and the parties involved in direct contact.

6.3 Authorization flow with challenge emv3ds authentication

The following scheme presents the general flow of an operation with challenge authentication performed through the Virtual POS.



1. The cardholder selects the products he/she wants to buy and enters the card details in a form displayed by the merchant.
2. The merchant initiates a request by sending the data to the Virtual POS.

3. The Virtual POS checks the configuration of the card, and in the response will inform if the card supports EMV3DS authentication and the EMV3DS protocol version that is being applied.

   3.1 If the card requires it, execute the 3DSMethod: connection is initiated from the browser to the ACS and this returns the result of the execution to the merchant.

4. The merchant sends the authorization request with a card that supports EMV3DS. In addition to the payment data, it is necessary to send the 3DSMethod result and additional data for authentication.

   The Virtual POS starts the authentication, and the issuer based on the data received decides that the cardholder must verify his/her identity (challenge)

5. The Virtual POS returns the information so that the cardholder can perform the challenge (authentication) with his/her issuing bank. At this time, and until the authentication response is received in step 9, the requests will be marked in the Management portal as "Unfinished" with the code = 8210.
6. The merchant redirects the cardholder via browser to connect with its issuer
7. The cardholder completes the challenge (authentication)
8. The issuer returns the challenge result (authentication) to the url indicated by the merchant
9. The merchant sends the result of the challenge (authentication) to the Virtual POS to finalize the authorization process
10. Once the payment is made, the virtual POS responds with the result of the operation.
11. The merchant responds with the information of the payment result to the cardholder

NOTE: The diagrams only consider the merchant flows and the parties involved in direct contact.

6.4 Example of requests to perform a transaction with EMV3DS authentication

6.5.1 Initiate request

This request allows obtaining information about the type of 3D Secure authentication that can be performed, in addition to the URL of the 3DSMethod, if it exists.

The request is initiated through a REST request to the Virtual POS, following the request structure indicated in point 3 of this document. The endpoints to use, depending on whether you want to make a request in the test environment or real operations, are the following:

URL Connection	Environment
https://sis-t.redsys.es:25443/sis/rest/iniciaPeticionREST	Tests
https://sis.redsys.es/sis/rest/iniciaPeticionREST	Real

The following describes the data that the Ds_MerchantParameters must include to send a Inicia Peticion to the REST Service:

```
{
  "DS_MERCHANT_ORDER":"1552571678",
  "DS_MERCHANT_MERCHANTCODE":"999008881",
  "DS_MERCHANT_TERMINAL":"999",
  "DS_MERCHANT_CURRENCY":"978",
  "DS_MERCHANT_TRANSACTIONTYPE":"0",
  "DS_MERCHANT_AMOUNT":"1000",
  "DS_MERCHANT_PAN":" XXXXXXXXXXXXXXXXX ",
"DS_MERCHANT_EMV3DS": {"threeDSInfo":"CardData"}
}
```

In response, the following will be obtained:

```
{
"Ds_Order":"1552571678",
  "Ds_MerchantCode":"999008881",
  "Ds_Terminal":"2",
  "Ds_TransactionType":"0",
  "Ds_EMV3DS": {
    "protocolVersion":"2.1.0",
    "threeDSServerTransID":"8de84430-3336-4ff4-b18d-f073b546ccea",
    "threeDSInfo":"CardConfiguration",
    "threeDSMethodURL":https://sis.redsys.es/sis-simulador-web/threeDsMethod.jsp
  },
    "Ds_Card_PSD2":"Y"
}
```

The **Ds_EMV3DS** parameter contains information about the card's authentication options. It will be composed of the following fields:

- **protocolVersion**: will always indicate the major version number allowed in the operation. The merchant will be responsible for using the version number for which it is prepared. If the version requires authentication with 3DSecure 1.0, the value "NO_3DS_V2" will be indicated.
- **threeDSServerTransID**: EMV3DS transaction identifier.
- **threeDSInfo**: CardConfiguration.
- **threeDSMethodURL**: URL of the 3DSMethod in case the card issuer has defined it.

The **Ds_Card_PSD2 parameter** would inform the merchant if the card reported in the request is affected or not by PSD2. Possible values will be "Y" to indicate that the card is affected by PSD2, or "N" to indicate the opposite.

6.5.2 3DSMethod execution

The 3DSMethod is a process that allows the issuing bank to capture the information of the device that the cardholder is using. This information, together with the EMV3DS data, which are sent in the authorization, will be used by the bank to make a risk assessment of the transaction. Based on this, the issuer can determine if the transaction is reliable and therefore does not require the intervention of the cardholder to verify his/her identity (frictionless).

The data capture of the device is done through an **iframe hidden** in the client's browser, which will establish a connection directly with the issuing bank in a transparent way for the user. The merchant will receive a notification when the information capture is finished and in the next step, when making the authorization request to the Virtual POS, the merchant must send the threeDSCompInd parameter **indicating** the execution of the 3DSMethod.

Steps for the execution of the 3DSMethod:

1. In the response received with the card configuration (iniciaPeticion) the following data are received to run the 3DSMethod:
    a. threeDSMethodURL: 3DSMethod url
    b. threeDSServerTransID: EMV3DS transaction identifier

    If threeDSMethodURL is not received in the response, this step ends, and the merchant must send threeDSCompInd = N in the authorization request.
2. Build the JSON Object with the following parameters:
    a. threeDSServerTransID: value received in the card query response
    b. threeDSMethodNotificationURL: url of the merchant to which the completion of 3DSMethod from the bank will be notified.
3. Encode the previous JSON in Base64url encode
4. A hidden iframe must be included in the client's browser, and send a threeDSMethodData field with the value of the previous json object, in an http post form to the url obtained in the initial threeDSMethodURL **query**

5. The issuing bank interacts with the browser to proceed to capture information. At the end, it will send the threeDSMethodData field in the html browser iframe via http post to the threeDSMethodNotificationURL url (indicated in step 2) and the 3DSMethod will end.
6. If the 3DSMethod has been completed in less than 10 seconds, threeDSCompInd **= Y** will be sent in the authorization. If it has not been completed within 10 seconds you must stop the wait and send the authorization with **threeDSCompInd = N**

6.5.3 Authorization request with EMV 3DS data

The authorization request is made through another REST request to the Virtual POS with the structure indicated in point 3 of this document.

The authorization request must be sent to the following endpoints depending on whether you want to make a request in the test environment or real operations:

URL Connection	Environment
https://sis-t.redsys.es:25443/sis/rest/trataPeticionREST	Tests
https://sis.redsys.es/sis/rest/trataPeticionREST	Real

The following describes the data the Ds_MerchantParameters must include to send an authorization request with EMV3DS authentication to the REST Service:

```
{
  "DS_MERCHANT_ORDER":1552572812,
  "DS_MERCHANT_MERCHANTCODE":"999008881",
  "DS_MERCHANT_TERMINAL":"2",
  "DS_MERCHANT_CURRENCY":"978",
  "DS_MERCHANT_TRANSACTIONTYPE":"0",
  "DS_MERCHANT_AMOUNT":"1000",
  "DS_MERCHANT_PAN":" XXXXXXXXXXXXXXXXX ",
  "DS_MERCHANT_EXPIRYDATE":"XXXX",
  "DS_MERCHANT_CVV2":"XXX",
"DS_MERCHANT_EMV3DS":
        {
                "threeDSInfo":"AuthenticationData",
                "protocolVersion":"2.1.0",
                "browserAcceptHeader":"text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0
        .8,application/json",
                "browserUserAgent":"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/71.0.3578.98 Safari/537.36",
                "browserJavaEnabled": "false",
                "browserJavaScriptEnabled": "true",
                "browserLanguage":"ES-es",
                "browserColorDepth":"24",
                "browserScreenHeight":"1250",
                "browserScreenWidth":"1320",
                "browserTZ":"52",
                "threeDSServerTransID":"8de84430-3336-4ff4-b18d-f073b546ccea",
                "notificationURL":"https://comercio-inventado.es/recibe-respuesta-autenticacion",
```

```
                    "threeDSCompInd":"Y"
                }
}
```

NOTE: The value of the version indicated in the protocolVersion field can't be higher than that returned by the Virtual POS in the previous call to iniciaPeticion.

In response you will get:

- If a Frictionless is ***done***, the result of the operation will be obtained directly:

```
{
  "Ds_Amount":"1000",
  "Ds_Currency":"978",
  "Ds_Order":"1552572812",
  "Ds_MerchantCode":"999008881",
  "Ds_Terminal":"2",
  "Ds_Response":"0000",
  "Ds_AuthorisationCode":"694432",
  "Ds_TransactionType":"0",
  "Ds_SecurePayment":"2",
  "Ds_Language":"1",
  "Ds_CardNumber":"454881******0004",
  "Ds_Card_Type":"C",
"Ds_MerchantData":"",
  "Ds_Card_Country":"724",
  "Ds_Card_Brand":"1"
}
```

- If ***Challenge*** is required, the data to perform the Challenge will be obtained:

```
 {
   "Ds_Amount":"1000",
   "Ds_Currency":"978",
   "Ds_Order":"1552572812",
   "Ds_MerchantCode":"999008881",
   "Ds_Terminal":"2",
   "Ds_TransactionType":"0",
     "Ds_EMV3DS":{
            "threeDSInfo":"ChallengeRequest",
            "protocolVersion":"2.1.0",
             "acsURL":"https://sis.redsys.es/sis-simulador-web/authenticationRequest.jsp",
              "creq":"eyJ0aHJlZURTU2VydmVyVHJhbnNJRCI6IjhkZTg0NDMwLTMzMzYtNGZmNC1iMThkLWYwNzNiNTQ2Y2NlYSIsImFjc1RyYW5zSUQiOiJkYjVjOTljNC1hMmZkLTQ3ZWUtOTI2Zi1mYTBiMDk0MzUyYTAiLCJtZXNzYWdlVHlwZSI6IkNSZXEiLCJtZXNzYWdlVmVyc2lvbiI6IjIuMS4wIiwiY2hhbGxlbmdlV2luZG93Sl6ZSI6IjA1In0"
   }
}
```

6.5.4 Challenge Execution

We describe this process in 3 steps:

Step 1.- Connection from the merchant to the ACS of the issuing bank

The next step consists in connecting from the merchant with the issuing bank so that the client may authenticate. This connection is made by sending an http POST form to the bank

ACS url. For this connection we use the data received in the parameter Ds_EMV3DS of the previous step (acsURL and creq parameters):

<Ds_EMV3DS>{"threeDSInfo":"ChallengeRequest",
"protocolVersion":"2.1.0",
"acsURL":"https://sis.redsys.es/sis-simulador-web/authenticationRequest.jsp",
"creq":"eyJ0aHJlZURTU2VydmVyVHJhbnNJRCI6ImU5OWMzYzI2LTFiZWItNGY4NS05ZmE3LTI3OTJiZjE5NDZlMiIsImFjc1RyYW5zSUQiOiIyMTQzNDFhYi0wMjlhLTRmMGEtOTEyNi1iMDFkYmE5OTc2MTkiLCJtZXNzYWdlVHlwZSI6IkNSZXEiLCJtZXNzYWdlVmVyc2lvbiI6IjIuMS4wIiwiY2hhbGxlbmdlV2luZG93U2l6ZSI6IjA1In0"}
</Ds_EMV3DS>

Example:
<form action="{acsURL}" method="POST" enctype = "application/x-www-form-urlencoded">
<input type="hidden" name="creq" value="{creq}" ">
</form>

With the data received in <Ds_EMV3DS> it would be:

<form action="https://sis.redsys.es/sis-simulador-web/authenticationRequest.jsp" method="POST" enctype = "application/x-www-form-urlencoded">

<input type="hidden" name="creq"
value="eyJ0aHJlZURTU2VydmVyVHJhbnNJRCI6ImU5OWMzYzI2LTFiZWItNGY4NS05ZmE3LTI3OTJiZjE5NDZlMiIsImFjc1RyYW5zSUQiOiIyMTQzNDFhYi0wMjlhLTRmMGEtOTEyNi1iMDFkYmE5OTc2MTkiLCJtZXNzYWdlVHlwZSI6IkNSZXEiLCJtZXNzYWdlVmVyc2lvbiI6IjIuMS4wIiwiY2hhbGxlbmdlV2luZG93U2l6ZSI6IjA1In0">
</form>

NOTE: It is important to respect upper and lower cases when typing the name of the parameters.

Step 2.-Challenge Execution

The cardholder is authenticated by the methods required by his/her issuing bank: OTP, static password, biometrics, etc.

Step 3.- Receipt of the authentication result

Once the challenge is over, the issuing bank will send the result to the merchant, making an http POST to the url of *the* notificationURL parameter that the merchant previously sent in the authorization request:

"notificationURL":" https://comercio-inventado.es/recibe-respuesta-autenticacion"

The merchant will receive the “cres” parameter that will be used in the final authorization request of the following section.

6.5.5 EMV 3DS authorization confirmation after the challenge

The following describes the data that the Ds_MerchantParameters must include to send an EMV3DS authorisation confirmation request to the REST Service:

{
"DS_MERCHANT_ORDER":1552577128,
"DS_MERCHANT_MERCHANTCODE":"999008881",
"DS_MERCHANT_TERMINAL":"2",
"DS_MERCHANT_CURRENCY":"978",

```
  "DS_MERCHANT_TRANSACTIONTYPE":"0",
  "DS_MERCHANT_AMOUNT":"1000",
  "DS_MERCHANT_PAN":" XXXXXXXXXXXXXXXXXX ",
  "DS_MERCHANT_EXPIRYDATE":"XXXX",
  "DS_MERCHANT_CVV2":"XXX",
  "DS_MERCHANT_EMV3DS":{
                        "threeDSInfo":"ChallengeResponse",
                        "protocolVersion":"2.1.0",
                        "cres":"eyJ0aHJlZURTU2VydmVyVHJhbnNJRCI6IjhkZTg0NDMwLTMzMzYtNGZmNC1i
            MThkLWYwNzNiNTQ2Y2NlYSIsImFjc1RyYW5zSUQiOiJkYjVjOTljNC1hMmZkLTQ3ZWUtOTI2Zi
            1mYTBiMDk0MzUyYTAiLCJtZXNzYWdlVHlwZSI6IkNSZXMiLCJtZXNzYWdlVmVyc2lvbiI6IjIuMS4
            wIiwidHJhbnNTdGF0dXMiOiJZIn0="
            }
}
```


*NOTE: The contents of the **cres** parameter must be sent as a continuous string with no carriage returns or line breaks.*

In response, the result of the operation will be obtained:


```
{
  "Ds_Amount":"1000",
  "Ds_Currency":"978",
  "Ds_Order":"1552572812",
  "Ds_MerchantCode":"999008881",
  "Ds_Terminal":"2",
  "Ds_Response":"0000",
  "Ds_AuthorisationCode":"694432",
  "Ds_TransactionType":"0",
  "Ds_SecurePayment":"2",
  "Ds_Language":"1",
  "Ds_CardNumber":"454881******0004",
  "Ds_Card_Type":"C",
"Ds_MerchantData":"",
  "Ds_Card_Country":"724",
  "Ds_Card_Brand":"1"
}
```


7. Transactions with DCC

Next, the additional features of the DCC operation in merchants using the REST interface are detailed. The merchant must be configured to perform this type of operation.

A payment with DCC in the Rest connection follows the next steps:

- **Step 1: Initiate request**

In the "Inicia Peticion" the merchant makes a query to the Virtual POS to find out if the card allows DCC operations. To make this DCC request, the Ds_MerchantParameters must include the **DS_MERCHANT_DCC** parameter.

Example of Ds_MerchantParameters in the initiate request for an operation with DCC.


```
{
  "DS_MERCHANT_ORDER":1552580496,
  "DS_MERCHANT_MERCHANTCODE":"999008881",
  "DS_MERCHANT_TERMINAL":"2",
  "DS_MERCHANT_TRANSACTIONTYPE":"0",
  "DS_MERCHANT_PAN":"XXXXXXXXXXXXXXXXXX",
  "DS_MERCHANT_CURRENCY":"978",
  "DS_MERCHANT_AMOUNT":"5785",
  "DS_MERCHANT_DCC":"Y"
}
```


In response, if the card allows DCC, the following will be obtained:


```
{
  "Ds_Order":"1552580496",
  "Ds_MerchantCode":"999008881",
  "Ds_Terminal":"2",
  "Ds_TransactionType":"0",
  "Ds_DCC":{
    "InfoMonedaTarjeta":{
      "monedaDCC":"826",
      "litMonedaDCC":"POUND STERLING.",
      "litMonedaRDCC":"GBP",
      "importeDCC":"53.60",
      "cambioDCC":"1.079288",
      "fechaCambioDCC":"2019-01-16",
      "markUp":"0.03",
      "cambioBCE":"1,136092"
      "porcentajeSobreBCE":"0,05"
    },
    "InfoMonedaComercio":{
      "monedaCome":"978",
      "litMonedaCome":"EUR",
      "importeCome":"57.85"
    }
  }
 "Ds_Card_PSD2":"Y"
}
```


If the card does not allow performing DCC the response will be:


```
{
  "Ds_Order":"1552580496",
  "Ds_MerchantCode":"999008881",
  "Ds_Terminal":"2",
  "Ds_TransactionType":"0",
  "Ds_DCC":{
            "dataDCC":"DCC does not apply"
  }
 "Ds_Card_PSD2":"Y"
}
```


- **Step 2: Authorization request**

The merchant will send the request for authorization of the operation indicating the information of the currency and DCC amount obtained in the previous step.

Example of request authorization with DCC


```
{
  "DS_MERCHANT_ORDER":1552581014,
  "DS_MERCHANT_MERCHANTCODE":"999008881",
  "DS_MERCHANT_TERMINAL":"2",
  "DS_MERCHANT_CURRENCY":"978",
  "DS_MERCHANT_TRANSACTIONTYPE":"0",
  "DS_MERCHANT_AMOUNT":"1000",
  "DS_MERCHANT_PAN":"XXXXXXXXXXXXXXXXXX",
  "DS_MERCHANT_EXPIRYDATE":"XXXX",
  "DS_MERCHANT_CVV2":"XXX",
"DS_MERCHANT_DCC":{
                    "monedaDCC":"840",
                    "importeDCC":"11.50"
                    }
}
```


In response, the result of the operation will be obtained:


```
{
  "Ds_Amount":"1000",
  "Ds_Currency":"978",
  "Ds_Order":"1552572812",
  "Ds_MerchantCode":"999008881",
  "Ds_Terminal":"2",
  "Ds_Response":"0000",
  "Ds_AuthorisationCode":"694432",
  "Ds_TransactionType":"0",
  "Ds_SecurePayment":"1",
  "Ds_Language":"1",
  "Ds_CardNumber":"454881******0004",
  "Ds_Card_Type":"C",
"Ds_MerchantData":"",
  "Ds_Card_Country":"724",
  "Ds_Card_Brand":"1"
}
```


NOTE: The maximum time from the beginning of the request and the authorization request is 1 hour. After this time, the request is given as lost and the flow must be carried out again from the beginning.

NOTE: *In this DCC operation, it must be taken into account that, by regulations of the European Central Bank (ECB), in operations involving the following currencies: Lev, Croatian Kuna, Danish Krone, Hungarian Forint, Zloty, Czech Krone, Romanian Leu, Swedish Krona, British Pound, it is necessary to inform the client of the% of increase between the applied change and the ECB change. This information will be returned, in operations in these currencies, in the parameters* cambioBCE *and* percentageOverBCE.

8. DCC transactions with authentication

Next, we will detail all those additional characteristics for an authentication transaction in which you want to use the DCC operation for merchants using the REST interface. The merchant must be configured to perform this type of operation.

Starting from the necessary steps to carry out a transaction with authentication, we will include the specific part of an operation with:

- **Step 1: Initiate request**

The merchant must consult the Virtual POS to know if the card is registered in EMV3DS to be able to start the authentication process. In this request the merchant can also check if the card offers **DCC**.

The following describes the data that the Ds_MerchantParameters must include to send a request for initiate request to the REST Service with authentication and DCC:

```
{
  "DS_MERCHANT_ORDER":1552580496,
  "DS_MERCHANT_MERCHANTCODE":"999008881",
  "DS_MERCHANT_TERMINAL":"2",
  "DS_MERCHANT_TRANSACTIONTYPE":"0",
  "DS_MERCHANT_PAN":"XXXXXXXXXXXXXXXXXX",
  "DS_MERCHANT_CURRENCY":"978",
  "DS_MERCHANT_AMOUNT":"1000",
 "DS_MERCHANT_DCC":"Y"
 "DS_MERCHANT_EMV3DS": {"threeDSInfo":"CardData"}
 }
```

If the card requires authentication and allows DCC, the response obtained will be as follows:

```
{
  "Ds_Order":"1552580496",
  "Ds_MerchantCode":"999008881",
  "Ds_Terminal":"2",
  "Ds_TransactionType":"0",
  "Ds_DCC":{
    "InfoMonedaTarjeta":{
"monedaDCC":"840",
      "litMonedaDCC":"DOLAR U.S.A.",
      "litMonedaRDCC":"USD",
      "importeDCC":"11.50",
      "cambioDCC":"0.869841",
      "fechaCambioDCC":"2019-01-16",
      "markUp":"0.03",
      "cambioBCE":"0,869835"
      "porcentajeSobreBCE":"0,05”
    },
    "InfoMonedaComercio":{
      "monedaCome":"978",
      "litMonedaCome":"EUR",
      "importeCome":"10.00"
    }
   },
   "Ds_EMV3DS": {
     "protocolVersion":"2.1.0",
     "threeDSServerTransID":"8de84430-3336-4ff4-b18d-f073b546ccea",
     "threeDSInfo":"CardConfiguration",
```

```
      "threeDSMethodURL":"https://sis.redsys.es/sis-simulador-web/threeDsMethod.jsp"
    },
      "Ds_Card_PSD2":"N"

}
```


- **Step 2: 3DSMethod (If applicable)**

The merchant executes the 3DSMethod so that the issuer captures the device information.

- **Step3: Authorisation request**

The merchant will send the operation authorization request including the result of the 3DSMethod and other additional data of the EMV3DS protocol. In addition, including the **DCC** information obtained in step 1.

The following describes the data that the Ds_MerchantParameters must include to send an authorization request with DCC to the REST Service:


```
{
  "DS_MERCHANT_ORDER":1552581014,
  "DS_MERCHANT_MERCHANTCODE":"999008881",
  "DS_MERCHANT_TERMINAL":"2",
  "DS_MERCHANT_CURRENCY":"978",
  "DS_MERCHANT_TRANSACTIONTYPE":"0",
  "DS_MERCHANT_AMOUNT":"1000",
  "DS_MERCHANT_PAN":"XXXXXXXXXXXXXXXXXX",
  "DS_MERCHANT_EXPIRYDATE":"XXXX",
  "DS_MERCHANT_CVV2":"XXX",
"DS_MERCHANT_DCC":{
                        "monedaDCC":"840",
                        "importeDCC":"11.50"
                                },
 "DS_MERCHANT_EMV3DS":
          {
                        "threeDSInfo":"AuthenticationData",
                        "protocolVersion":"2.1.0",
                        "browserAcceptHeader":"text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0
          .8,application/json",
                        "browserUserAgent":"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
 (KHTML, like Gecko) Chrome/71.0.3578.98 Safari/537.36",
                        "browserJavaEnabled":"false",
                        "browserJavaScriptEnabled": "true",
                        "browserLanguage":"ES-es",
                        "browserColorDepth":"24",
                        "browserScreenHeight":"1250",
                        "browserScreenWidth":"1320",
                        "browserTZ":"52",
                        "threeDSServerTransID":"8de84430-3336-4ff4-b18d-f073b546ccea",
                        "notificationURL":"https://comercio-inventado.es/recibe-respuesta-autenticacion",
                        "threeDSCompInd":"Y"
          }

}
```


As a result of this request, you may have one of the following options:

a. Authentication OK (Frictionless): the operation has been authenticated and the Virtual POS will continue the authorization process.

b. Challenge Required: The issuer needs to verify the identity of the client

c. Other result: Authentication not available, authentication rejected, authentication failed, etc.

The Virtual POS will decide whether to authorize or reject the operation.

- **Step 4: Authentication (In the event that the issuing bank has requested challenge)**

The issuing bank requires the cardholder to verify his/her identity (through OTP, static password, biometrics, etc.).

- **Step 5: Authorization Confirmation**

The merchant will send the authorization, with the result of the challenge, to the Virtual POS to finalize the authorization process. All the information for the payment, as well as the DCC fields in the same way as in step 3, must also be included.

NOTE: We recommend that in step 3 the merchant provides all the additional information to increase the probability of frictionless flow and a greater authorization rate.

9. PSD2 adaptations

According to the PSD2 regulation (entered into force on September 14, 2019), a European directive that aims to improve security and reinforce client authentication in electronic commerce operations. As a basic rule, the cardholder's authentication shall be required in all operations (SCA), however, it is also considered the option that the merchant, in the payment request, ask for an exemption to avoid said authentication. To request an exemption, the merchant must include the following parameter in its requests.

PARAMETER	POSSIBLE VALUES
DS_MERCHANT_EXCEP_SCA	LWV, TRA, MIT, COR, ATD

- LWV (Low value transaction): Low amount exemption (up to € 30, with a maximum of 5 operations or € 100 accumulated per card, these counters are controlled at the card issuer level)
- TRA (Transaction Risk Analysis): This exemption is based on a transaction risk analysis by the acquirer / merchant.
- MIT (Merchant Initiated Transaction): Operation initiated by the merchant without being associated with an action or event of the client, that is, without any possible interaction with the client. These operations are outside the scope of PSD2. This is the case of subscription, recurring payments operations, etc. and, in general, almost all those that require the storage of the client's payment credentials (COF) or its equivalent "payment by reference". All payment operations initiated by the merchant (MIT) require that the initial operation, when the client grants permission to the merchant to use his/her payment credentials, be done through an operation authenticated with SCA.
- COR (Secure Corporate Payment): exemption restricted to the case of transactions between companies, not consumers.
- ATD: Delegated authentication exemption. Delegated Authentication is a brand-specific program (for more information, refer the brand's documentation on this subject)

NOTE: It should be considered that to improve the user experience in the LWV, TRA and COR exemptions, the first option will be to mark the exemption in the authentication step. This allows that if the issuer does not accept the exemption proposal, an authentication at the same time without having to reject the operation can be requested (challenge required EMV3DS).

9.1 Examples of requests with exemptions.

As indicated in the previous point, the regulations contemplate different exemptions that can be marked in the payment request to propose not to carry out the authentication process of the cardholder. It must be considered that when proposing an exemption, the merchant would be responsible for possible fraud in the operation

Two types of messages are considered where we mark an exemption:

- **Request with EMV3DS data**. The exemptions requested in requests with sending of EMV3DS data, will be marked in the authentication. If this exemption is not accepted by the issuer, a CHALLENGE request will be returned for the cardholder to authenticate with SCA. In this way, the request is not lost, and the usual flow will continue, without the cardholder being affected. THIS OPTION IS RECOMMENDED.
- **Request without EMV3DS data.** The exemptions requested in the requests in which the EMV3DS data have not been reported, will be marked in the authorization. If this exemption is not accepted by the issuer, a denial will be made with **Ds_Response = 0195**

("soft-decline" requires SCA). In this case, the merchant can decide to start the operation again with EMV3DS data, but a new request must be sent.

Inicia Peticion message (to know exemptions allowed to the merchant)

The allowed exemptions will depend on the configuration of the merchant. The activation of these exemptions is carried out by the acquiring bank.

To know which exemptions the merchant may apply in each of the transactions, the DS_MERCHANT_EXCEP_SCA parameter with the value "Y" in the call to '*Inicia Peticion*' must be sent.

NOTE: For the TRA exemption, a maximum amount is established which will also be reported in the response.

Example of Inicia Peticion:

```
{
  "DS_MERCHANT_ORDER":"1552571678",
  "DS_MERCHANT_MERCHANTCODE":"999008881",
  "DS_MERCHANT_TERMINAL":"999",
  "DS_MERCHANT_CURRENCY":"978",
  "DS_MERCHANT_TRANSACTIONTYPE":"0",
  "DS_MERCHANT_AMOUNT":"1000",
  "DS_MERCHANT_EXCEP_SCA":"Y",
  "DS_MERCHANT_PAN":" XXXXXXXXXXXXXXXXX ",
"DS_MERCHANT_EMV3DS": {"threeDSInfo":"CardData"}
}
```

An example response would be the following:

```
{
"Ds_Order":"1552571678",
  "Ds_MerchantCode":"999008881",
  "Ds_Terminal":"2",
  "Ds_TransactionType":"0",
  "Ds_EMV3DS": {
    "protocolVersion":"2.1.0",
    "threeDSServerTransID":"8de84430-3336-4ff4-b18d-f073b546ccea",
    "threeDSInfo":"CardConfiguration",
    "threeDSMethodURL":https://sis.redsys.es/sis-simulador-web/threeDsMethod.jsp
  },
  "Ds_Excep_SCA":"LWV;TRA[30.0];COR;MIT",
   "Ds_Card_PSD2":"Y"
}
```

Trata Peticion message (with EMV3DS)

The DS_MERCHANT_EXCEP_SCA parameter is included with the value of the proposed exception. As the EMV3DS fields are also reported, the exemption will be requested at the time of requesting authentication.

```
{
  "DS_MERCHANT_ORDER":1552572812,
  "DS_MERCHANT_MERCHANTCODE":"999008881",
  "DS_MERCHANT_TERMINAL":"2",
  "DS_MERCHANT_CURRENCY":"978",
  "DS_MERCHANT_TRANSACTIONTYPE":"0",
```

```
  "DS_MERCHANT_AMOUNT":"100",
  "DS_MERCHANT_PAN":" xxxxxxxxxxxxxxxxx ",
  "DS_MERCHANT_EXPIRYDATE":"XXXX",
  "DS_MERCHANT_CVV2":"XXX",
"DS_MERCHANT_EXCEP_SCA:"LWV",
  "DS_MERCHANT_EMV3DS":
          {
                  "threeDSInfo":"AuthenticationData",
                  "protocolVersion":"2.1.0",
                  "browserAcceptHeader":"text/html,application/xhtml+xml,application/xml;q=0.9,*/*;q=0.8
        ,application/json",
                  "browserUserAgent":"Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36
(KHTML, like Gecko) Chrome/71.0.3578.98 Safari/537.36",
                  "browserJavaEnabled": "false",
                  "browserJavaScriptEnabled": "true",
                  "browserLanguage":"ES-es",
                  "browserColorDepth":"24",
                  "browserScreenHeight":"1250",
                  "browserScreenWidth":"1320",
                  "browserTZ":"52",
                  "threeDSServerTransID":"8de84430-3336-4ff4-b18d-f073b546ccea",
                  "notificationURL":"https://comercio-inventado.es/recibe-respuesta-autenticacion",
                  "threeDSCompInd":"Y"
          }
}
```


Trata Peticion message (without EMV3DS)

The DS_MERCHANT_EXCEP_SCA parameter is included with the value of the proposed exception. As EMV3DS fields are not reported, the exemption will be requested directly in the authorization request, so if the issuer does not accept it, it can deny it with a 0195 "soft-decline" to indicate that it requires authentication. If the terminal has secure payment methods configured, the DS_MERCHANT_DIRECTPAYMENT parameter must also be added with the value true.


```
{
  "DS_MERCHANT_ORDER":1552572812,
  "DS_MERCHANT_MERCHANTCODE":"999008881",
  "DS_MERCHANT_TERMINAL":"2",
  "DS_MERCHANT_CURRENCY":"978",
  "DS_MERCHANT_TRANSACTIONTYPE":"0",
  "DS_MERCHANT_AMOUNT":"100",
  "DS_MERCHANT_PAN":" xxxxxxxxxxxxxxxxx ",
  "DS_MERCHANT_EXPIRYDATE":"XXXX",
  "DS_MERCHANT_CVV2":"XXX",
  “DS_MERCHANT_DIRECTPAYMENT”:”true”,
"DS_MERCHANT_EXCEP_SCA:"LWV"
}
```


9.2 MIT transaction example (Merchant Initiated Transaction)

A MIT transaction is the one initiated by the merchant itself without any possible interaction with the client. For example, monthly payment of a receipt or subscription fee. This type of

operation, since the client is not present and his/her authentication is not possible, will not require authentication by the cardholder (SCA).

To correctly identify this type of transaction, the merchant must include in the payment request, the **DS_MERCHANT_EXCEP_SCA** parameter with the value **MIT** and, in addition, send the **DS_MERCHANT_DIRECTPAYMENT parameter** with the value **true**.

These MIT transactions may be associated with an initial payment request (Initial **COF Operation**) in which the owner is present and grants permission to the merchant to use its payment data in subsequent charges, according to a service provided continuously over time. This initial operation must be authenticated with SCA and must be marked following the COF specifications. The MIT operation also requires that the COF indicator be correctly marked, according to the specific use that is being made of the stored credentials.

It must be considered that not all operations in which stored card data / credentials (COF) are used can be considered MIT. For example, the **1-click payment** operation, where the client's credentials are stored or tokenized (payment by reference), payment requests that are made using the stored credentials **CAN'T** be considered transactions initiated by the merchant since the cardholder is present and thus he/she can be authenticated. In this case, according to PSD2, and if no other exemption is applied, the use of enhanced authentication (SCA) is required.

NOTE 1: For more information on Credentials on File (COF) specifications please refer to Especificaciones COF Ecom Guide.

NOTE 2: The complete list of all SIS input parameters is available in the document "TPV-Virtual Parámetros Entrada-Salida.xlsx".

10. Advanced 3RI and OTA functionalities

As for this point of R3I operations, the specifications are not yet closed by the card brands. This point is temporarily deleted until we have the requirements of the card brands that allow us to implement the definitive solution for this type of transaction.

11. Other REST integrations: PSPs/ External MPI / PUCE

11.1 Integration for PSP

If you are a merchant aggregator or a PSP there is a specific integration so that, with a secret key for PSPs, they can operate on behalf of the merchants at the terminal level.

The parameters to send in these payment requests are the same as in the request from a merchant but using a specific signature version for PSP (ANSI X9.19).

No specific flows are defined for the processing of operations by the PSP since they are the same as those already referred in this document.

NOTE: To carry out this integration with the Redsys Virtual POS, activation by the Acquiring Bank is required.

11.1.1 Configuration

It is necessary that a merchant-terminal to process requests through a PSP be associated with this PSP. This configuration must be requested by the merchant to its Bank.

11.1.2 Request and receipt of keys

Two private keys will be received with the protocol set forth by Redsys.

- Key to perform 3DES encryption of the DS_MERCHANT_PAN field
- Key to sign the (DS_MERCHANTPARAMETERS) request according to the ordinary X9.19

11.1.3 Submitting a request to the Virtual POS

As in the payment request sent by a merchant, the PSP must send a REST request with a JSON made up of the following three fields:

- **Ds_SignatureVersion**: Constant value indicating the signature version that is being used. For this PSP integration the value to be used will be **T25V1.**
- **Ds_MerchantParameters**: String in JSON format with all the parameters of the request encoded in Base 64 and without carriage returns (the list of parameters that can be sent in a payment request is included in the Annex section). If the DS_MERCHANT_PAN field is sent this must be signed with 3DES.
- **Ds_Signature**: Signature of the data sent. It is the result of the Mac X9.19 of the JSON string encoded in Base 64 sent in the previous parameter. The value to be sent in the Ds_Signature field is obtained by converting the Mac from the previous step to hexadecimal and taking the first 4 bytes (8 characters) starting from the left of the result.

These parameters must be sent to the following endpoints, depending on whether you want to make a request in the test environment or actual operations:

URL Connection initiates request	Environment
https://sis-t.redsys.es:25443/sis/rest/iniciaPeticionPSPREST	Tests
https://sis.redsys.es/sis/rest/iniciaPeticionPSPREST	Real
URL Connection processes request	
https://sis-t.redsys.es:25443/sis/rest/trataPeticionPSPREST	Tests
https://sis.redsys.es/sis/rest/trataPeticionPSPREST	Real

11.1.4 Receiving the result

The reception of the result will be signed in the same way as the request for submission, according to the ANSI x9.19 standard

11.1.5 Example of requests

JSON string

*{"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_PAN":"454881********04","DS_MERCHANT_EXPIRYDATE":"1512","DS_MERCHANT_CVV2":"123"}*

We encrypt the DS_MERCHANT_PAN parameter in 3DES

Card number	*454881********04*
3DES encrypted hexadecimal card number CBC mode without vector, key F180E06B7A89A88F4A2A52C8EC1C5D1C	*377f34989cf0ae79857a70aa45f3e4c3*

JSON string with encrypted PAN:

{"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_PAN":"377f34989cf0ae79857a70aa45f3e4c3","DS_MERCHANT_EXPIRYDATE":"1512","DS_MERCHANT_CVV2":"123"}

The JSON object encoded in BASE64, DS_MERCHANT_PARAMETERS field, is shown below:

eyJEU19NRVJDSEFOVF9BTU9VTlQiOiIxNDUiLCJEU19NRVJDSEFOVF9PUkRFUiI6IjE0NDYwNjg1ODEiLCJEU19NRVJDSEFOVF9NRVJDSEFOVENPREUiOiI5OTkwMDg4ODEiLCJEU19NRVJDSEFOVF9DVVJSRU5DWSI6Ijk3OCIsIkRTX01FUkNIQU5UX1RSQU5TQUNUSU9OVFlQRSI6IjAiLCJEU19NRVJDSEFOVF9URVJNSU5BTCI6IjEiLCJEU19NRVJDSEFOVF9NRVJDSEFOVFVSTCI6Imh0dHA6XC9cL3d3dy5wcnVlYmEuY29tXC91cmxOb3RpZmljYWNpb24ucGhwIiwiRFNfTUVSQ0hBTlRfUEFOIjoiMzc3ZjM0OTg5Y2YwYWU3OTg1N2E3MGFhNDVmM2U0YzMiLCJEU19NRVJDSEFOVF9FWFBJUllEQVRFIjoiMTUxMiIsIkRTX01FUkNIQU5UX0NWVjIiOiIxMjMifQ==

Sign the request details

On the whole string obtained in the previous step (DS_MERCHANTPARAMETERS) the complete signature is calculated based on the x9.19 standard.

We obtain the MAC 5D823A402DE70705 with the encryption key 269289DA6EAD0B20928C8F2F2F6BC752 and the ds_merchantparameters.

The ds_signature field will be obtained by taking the first 4 bytes (8 characters) starting from the left of the MAC result from the previous step ***5D823A40***

Compose the request message

- *Ds_SignatureVersion: T25V1*
- *Ds_MerchantParameters:*
  eyJEU19NRVJDSEFOVF9BTU9VTlQiOiIxNDUiLCJEU19NRVJDSEFOVF9PUkRFUiI6IjE0NDYwNjg1ODEiLCJEU19NRVJDSEFOVF9NRVJDSEFOVENPREUiOiI5OTkwMDg4ODEiLCJEU19NRVJDSEFOVF9DVVJSRU5DWSI6Ijk3OCIsIkRTX01FUkNIQU5UX1RSQU5TQUNUSU9OVFlQRSI6IjAiLCJEU19NRVJDSEFOVF9URVJNSU5BTCI6IjEiLCJEU19NRVJDSEFOVF9NRVJDSEFOVFVSTCI6Imh0dHA6XC9cL3d3dy5wcnVlYmEuY29tXC91cmxOb3RpZmljYWNpb24ucGhwIiwiRFNfTUVSQ0hBTlRfUEFOIjoiMzc3ZjM0OTg5Y2YwYWU3OTg1N2E3MGFhNDVmM2U0YzMiLCJEU19NRVJDSEFOVF9FWFBJUllEQVRFIjoiMTUxMiIsIkRTX01FUkNIQU5UX0NWVjIiOiIxMjMifQ==
- *Ds_Signature: 5D823A40*

11.2 External MPI / 3DSServer

If the PSP or integrator has its own certification to carry out authentication management externally to Redsys, it may carry out transactions indicating that the operation has already been authenticated. The PSP must ask the Acquiring Bank for the configuration to carry out this transaction.

It is an essential requirement that the PSP be certified with EMVCO with the card brands.

The requests will be made as indicated in this document, adding to the DS_MERCHANTPARAMETERS field the DS_MERCHANT_MPIEXTERNAL parameter, of the JSON Object type.

Fields to include in an operation authenticated with external 3DSServer in version 2:

- **threeDSServerTransID:** identifier of the transaction used in authentication messages.
- **authenticacionValue:** value returned by the card issuer's ACS.
- **dsTransID:** Directory identifier used in authentication messages.
- **protocolVersion:** version of the EMV 3DS protocol with which the authentication was performed.
- **Eci:** value of the Eci field returned by the ACS in the authentication.
- **authenticationFlow**: optional field to indicate if the authentication was performed with Challenge (C) or with Frictionless (F). If nothing is indicated, it will be considered to have been done with Challenge.

Below is an example with the data to include in the Ds_MerchantParameters to send an authentication request with external 3DSServer in version 2:

```
{
  "DS_MERCHANT_ORDER":1552572812,
  "DS_MERCHANT_MERCHANTCODE":"999008881",
  "DS_MERCHANT_TERMINAL":"2",
  "DS_MERCHANT_CURRENCY":"978",
  "DS_MERCHANT_TRANSACTIONTYPE": "0",
  "DS_MERCHANT_AMOUNT":"1000",
  "DS_MERCHANT_PAN":" xxxxxxxxxxxxxxxxx ",
  "DS_MERCHANT_EXPIRYDATE":"XXXX",
"DS_MERCHANT_CVV2":"XXX"",
  " DS_MERCHANT_MPIEXTERNAL ":{
    "threeDSServerTransID":"7bb7e07d-d2e5-467e-9e96-69da4542b759",
    "dsTransID:" 7bb7e07d-d2e5-467e-9999-69da4542b770",
    "authenticacionValue":"jBXyoyIcA+KlCREAAA/aAAkAAAA=" ,
    "protocolVersion":"2.1.0",
    "Eci":"05",
    "authenticationFlow":"C"
    }
}
```


11.3 PUCE (authentication)

This type of integration allows merchants / integrators that make electronic commerce payment requests through the PUC protocol (PRICE merchants), to authenticate the cardholder using the 3DSecure protocol (EMV 3DS). To perform this type of integration, see the *PUCE-REST integration manual*.

These merchants / integrators can use the REST protocol to authenticate the cardholder using the 3DSecure protocol and, once the cardholder has been authenticated, send the authorization by PUC / PRICE connection. This sending of financial authorization through PUC is not strictly necessary, since with this same REST integration it is also possible to request authorization, that is, with this REST integration the merchant / integrator can do the entire operation flow, the authentication of the cardholder and the authorization request. In this case the integration process described in this guide must be followed.

12. Test environment

The merchant can use the test environment to carry out the tests that needs to verify the correct operation of its integration before implementing in the real environment.

This guide provides generic test data that can be used by any client, if the merchant is interested in carrying out these tests with its own data, it should contact its bank to provide access data.

The URLs to access the test environment are:

URL Connection initiates request	Integration type
https://sis-t.redsys.es:25443/sis/rest/iniciaPeticionREST	Merchant

https://sis-t.redsys.es:25443/sis/rest/iniciaPeticionPSPREST	PSP
URL Connection processes request	
https://sis-t.redsys.es:25443/sis/rest/trataPeticionREST	Merchant
https://sis-t.redsys.es:25443/sis/rest/trataPeticionPSPREST	PSP
URL access to the Management portal	
https://sis-t.redsys.es:25443/canales/portal	Merchant / PSP

NOTE: The operations flow in the test environment is the same than in the production environment with the only difference that payments made in this environment will not have accounting validity.

GENERIC TEST DATA

- Merchant number (Ds_Merchant_MerchantCode): Here you must enter the number provided by your bank (example 999008881)
- Terminal (Ds_Merchant_Terminal): Here the number provided by the bank must be provided (example 01)
- Merchant integration secret key: sq7HjrUOBfKmC576ILgskD5srU870gJ7

Test card	Description
4918019160034602 Expiration: not validated CVV2: other than 999	EMV3DS 2.1 with *threeDSMethodURL and* Frictionless authentication
4548814479727229 Expiration: not validated CVV2: other than 999	EMV3DS 2.1 without *threeDSMethodURL and* Frictionless authentication
4918019199883839 Expiration: not validated CVV2: other than 999	EMV3DS 2.1 with *threeDSMethodURL and* Challenge authentication
4548817212493017 Expiration: not validated CVV2: other than 999	EMV3DS 2.1 without *threeDSMethodURL and* Challenge authentication
Denied operations	

Any operation with a value of CVV2 = 999 or amount ended in 96 To perform a denial of operation test by response code 172, 173, or 174, simply use the code value as CVV or set an amount ending in 72, 73, or 74. This will only work in test environments.	
PSD2 tests	
4548816134581156 Expiration: not validated CVV2: other than 999	EMV3DS 2.2 *without threeDSMethodURL with Frictionless authentication*
4548816131164386 Expiration: not validated CVV2: other than 999	EMV3DS 2.2 *without threeDSMethodURL with Challenge authentication*
4548815324058868 Expiration: not validated CVV2: other than 999	EMV3DS 2.2 *without threeDSMethodURL. If an exemption is sent, Frictionless authentication will be performed. If no exemptions are sent it will ask for Challenge*
4548815374025114 Expiration: not validated CVV2: other than 999	EMV3DS 2.2 *without threeDSMethodURL. If MIT is sent it will return Frictionless, in any other case it will ask for Challenge*
5576441563045037 Expiration: not validated CVV2: other than 999	EMV3DS 2.2 *without threeDSMethodUR. 3RI-OTA payments are accepted*
4548817212493017 Expiration: not validated CVV2: other than 999	Card with soft decline. Denial 195, if the authorization is not authenticated.
DCC tests	
5424180805648190 Expiration: not validated CVV2: other than 999	Master card with DCC and EMV3DS 2 Frictionless
4117731234567891 Expiration: not validated CVV2: other than 999	Visa card with DCC and EMV3DS 2 Challenge
5409960031405146 Expiration: not validated CVV2: other than 999	Master card with Norwegian Korona currency and EMV3DS 2 Challenge
5171471234567894 Expiration: not validated CVV2: other than 999	Master card with Pound currency and EMV3DS 2 Challenge
4001151234567891 Expiration: not validated CVV2: other than 999	Visa card with Pound currency and EMV3DS 2 Challenge

Other card brands	
36849800000018 Expiration: not validated CVV2: other than 999	Diners Club card
36849800000000 Expiration: not validated CVV2: other than 999	Diners Club card
376674000000008 Expiration: not validated CVV2: other than 999	Amex card
376674000000016 Expiration: not validated CVV2: other than 999	Amex card
3587870000000001 Expiration: not validated CVV2: other than 999	JCB card
3587870000000019 Expiration: not validated CVV2: other than 999	JCB card

13. Examples of the most common types of operation

The following points indicate the parameters to include in some of the most common types of operations. Examples of the fields included in the Ds_MerchantParameters parameter are included. The details of how to configure a REST request to the Virtual POS are indicated in point 3 of this document.

Request for payment/ pre-authorization (with sending card data without authentication)

The DS_MERCHANT_TRANSACTIONTYPE field indicates the type of operation to be carried out:

* DS_MERCHANT_TRANSACTIONTYPE":"0" for PAYMENT
* DS_MERCHANT_TRANSACTIONTYPE":"1" for PRE-AUTHORIZATION

The value of Ds_MerchantParameters would be:

{ "DS_MERCHANT_ORDER":"1552565870",
"DS_MERCHANT_MERCHANTCODE":"999008881",
"DS_MERCHANT_TERMINAL":"999",
"DS_MERCHANT_CURRENCY":"978",
"DS_MERCHANT_TRANSACTIONTYPE":"0",

"DS_MERCHANT_AMOUNT":"1000",
"DS_MERCHANT_PAN":"XXXXXXXXXXXXX",
"DS_MERCHANT_EXPIRYDATE":"XXXX",
"DS_MERCHANT_CVV2":"XXX"}

Request for Confirmation / Refund / Cancellation

In this case, it is not necessary to provide the card details and the operation must be indicated in the DS_MERCHANT_ORDER field. The DS_MERCHANT_TRANSACTIONTYPE field indicates the type of operation to be carried out:

* DS_MERCHANT_TRANSACTIONTYPE":"2" for CONFIRMATION
* DS_MERCHANT_TRANSACTIONTYPE":"3" for RETURN
* DS_MERCHANT_TRANSACTIONTYPE":"9" for CANCELLATION

The value of Ds_MerchantParameters would be:

{ "DS_MERCHANT_ORDER":"1552565870",
"DS_MERCHANT_MERCHANTCODE":"999008881",
"DS_MERCHANT_TERMINAL":"999",
"DS_MERCHANT_CURRENCY":"978",
"DS_MERCHANT_TRANSACTIONTYPE":"3",
"DS_MERCHANT_AMOUNT":"1000"}

Request for Tokenization (Payment by Reference - 1-Click Payment)

The DS_MERCHANT_IDENTIFIER = REQUIRED field must be included to request the generation of the reference. The DS_MERCHANT_COF_TYPE field must also be included to indicate the use that will be made of said reference.

The value of Ds_MerchantParameters would be:

{ "DS_MERCHANT_ORDER":"1552565870",
"DS_MERCHANT_MERCHANTCODE":"999008881",
"DS_MERCHANT_TERMINAL":"999",
"DS_MERCHANT_CURRENCY":"978",
"DS_MERCHANT_TRANSACTIONTYPE":"0",
"DS_MERCHANT_AMOUNT":"1000",
"DS_MERCHANT_PAN":"XXXXXXXXXXXXX",
"DS_MERCHANT_EXPIRYDATE":"XXXX",
"DS_MERCHANT_CVV2":"XXX",
" DS_MERCHANT_IDENTIFIER ":" REQUIRED "}
" DS_MERCHANT_COF_TYPE ":" R "
}

NOTE: as a requirement of PSD2, SCA authentication must be requested, so the fields related with the authentication must also be included.

Request for Payment with Tokenization (Payment by Reference - 1-Click Payment)

The DS_MERCHANT_IDENTIFIER is included with the reference value instead of the card data

The value of Ds_MerchantParameters would be:

{ "DS_MERCHANT_ORDER":"1552565870",

"DS_MERCHANT_MERCHANTCODE":"999008881",
"DS_MERCHANT_TERMINAL":"999",
"DS_MERCHANT_CURRENCY":"978",
"DS_MERCHANT_TRANSACTIONTYPE":"0",
"DS_MERCHANT_AMOUNT":"1000",
" DS_MERCHANT_IDENTIFIER ":"XXXXXXXXXXXXXXXXXXXXXXXX"}

14. Timeout

What to do if the Virtual POS does not respond to a request?

This problem can have two possible causes:

- The request was not received, so the Virtual POS will not respond to the request message.
- The Virtual POS has received the request message but is not able to contact the Authorization Centre. This connection has a 30-second timeout defined, so if after that time, no response is received from the Authorizing Centre, a response message with code 9912/912 “Issuer not available” will be returned. The client application must therefore establish a longer timeout (about 40 or 50 seconds), to guarantee that the Virtual POS will always respond.

What to do in case of timeout?

For requests for payment, pre-authorization, or confirmation the relevant cancellation operation must be sent.

In the case of return operations or cancellation operations, the request can be made again.

15. Frequent errors

Signature error (SIS0042)

When there is a signature error, the merchant must verify:

- That the data that have been used to make the signature are the same as those sent in the form, considering that any modification of the value or format of a field after the signature calculation makes it incorrect.
- That the secret key used by the merchant matches with the key loaded by the merchant in the management module (merchant section).
- It should be checked that merchants are not sending blank spaces in the signature. If the request is made through cURL or through the Safari browser, the "+" symbols might become blank spaces. To avoid this, the "+" symbols of the signature must be replaced by "% 2B" (URL encoded value).
- If the merchant fails to locate which parameter is wrong, it should contact the Redsys Client Service Centre, or the Redsys Integration Support department if its bank has provided the contact.

I have in my merchant refusals by repeated number (SIS0051), though I am not aware of having repeated them.

This usually occurs because the merchant platform is generating repeated order numbers only when it receives denials or authorizations, but it is repeating them when the transactions are half completed. Given this, there are two options:

- Request the support service to configure the POS so that you can repeat order numbers. Maximum of one authorized operation per day and no limit for those denied.
- Always generate different order numbers, not only for authorized and denied operations, but for those that have not finished after some time.

I need to return an operation, but the return option does not appear in the management module.

This means that the user used to access the Management Portal does not have permission to make returns. If you need this permission you should contact your bank.

16. Frequently asked questions

I am a merchant and I need to know the encryption key of my Virtual POS

In point 3 of this same document it is indicated how to access the key value.

My access merchant user to the Channel management module is blocked. How can I unlock it?

Under the username and password boxes there is a "I forgot my password" link. After clicking it, you must enter your username and confirm the email address for the new password.

ANNEXES

1. Help libraries to calculate the signature

In the previous sections the way of accessing the SIS using the REST input and the signature system based on HMAC SHA256 has been described. This section explains how to use the libraries available in PHP, JAVA and.NET to allow the development and generation of payment form fields. The use of the libraries provided by Redsys is optional, although they simplify the developments to be made by the merchant.

1.1 PHP Library

Below are the steps that a merchant must follow to use the PHP library provided by Redsys:

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

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

The merchant must decide if the import will be performed with the “include” or “required” function, depending on the developments made.

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

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

3. Calculate the Ds_MerchantParameters parameter. To carry out the calculation of this parameter, initially all the parameters of the payment request to be sent must be added, 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);
```

Finally, to calculate the Ds_MerchantParameters parameter, the library function "createMerchantParameters ()" must be named, as shown below:

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

4. Calculate the Ds_Signature parameter. To carry out the calculation of this parameter, the library function "createMerchantSignature ()" must be named with the key obtained from the management module, as shown below:

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

5. Once the values of the Ds_MerchantParameters and Ds_Signature parameters have been obtained, the REST request must be filled with these values and the Ds_SignatureVersion parameter.

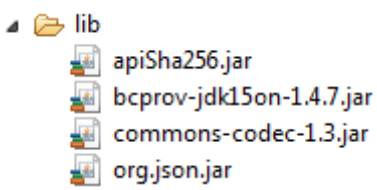
1.2 JAVA Library

Below, the steps that a merchant must follow to use the JAVA library provided by Redsys are described:

1. Import the library, as shown below:

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

The merchant must include in the project construction route all the libraries (JARs) that are provided:



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

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

3. Calculate the Ds_MerchantParameters parameter. To carry out the calculation of this parameter, initially all the parameters of the payment request to be sent must be added, 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);
```

Finally, the library function "createMerchantParameters ()" must be named, as shown below:

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

4. Calculate the Ds_Signature parameter. To carry out the calculation of this parameter, the library function "createMerchantSignature ()" must be named with the key obtained from the management module, as shown below:

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

Once the values of the Ds_MerchantParameters and Ds_Signature parameters have been obtained, the REST request must be filled with these values and the Ds_SignatureVersion parameter.

1.3 Library.NET

Below, the steps that a merchant must follow to use the library are described. NET provided by Redsys:

1. Import the RedsysAPI and Newronsoft.Json library in the project.
2. Calculate the **Ds_MerchantParameters parameter**. To carry out the calculation of this parameter, initially all the parameters of the payment request to be sent must be added, 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);
```

Finally, the library function “createMerchantParameters ()” must be named, as shown below:

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

3. Calculate the **Ds_Signature** parameter. To carry out the calculation of this parameter, the library function “createMerchantSignature ()” must be named with the key obtained from the management module, as shown below:

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

4. Once the values of the Ds_MerchantParameters and Ds_Signature parameters have been obtained, the REST request must be filled with these values and the Ds_SignatureVersion parameter.

2. Help libraries response request for payment

In the previous sections the way of accessing the SIS using REST connection has been described. This section explains how to use the libraries available in PHP, JAVA and .NET to allow the developments for the reception of the parameters in the response of the REST service. The use of the libraries provided by Redsys is optional, although they simplify the developments to be made by the merchant.

2.3.1 PHP Library

Below are the steps that a merchant must follow to use the PHP library provided by Redsys:

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

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

The merchant must decide if the import will be performed with the "include" or "required" function, depending on the developments made.

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

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

2. Capture the response parameters:

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

3. Decode the **Ds_MerchantParameters** parameter. To perform the decoding of this parameter, the library function "decodeMerchantParameters ()" must be named, as shown below:

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

4. Once the "decodeMerchantParameters ()" function call has been executed, the value of any parameter that is likely to be included in the response (Annex **¡Error! No se encuentra el origen de la referencia.**). To obtain the value of a parameter, the "getParameter ()" function of the library must be named with the parameter name to obtain the response code, as shown below:

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

IMPORTANT NOTE: It is important to carry out the validation of all the parameters that are sent in the communication.

5. Validate the **Ds_Signature** parameter. To carry out the validation of this parameter, the signature must be calculated and compared with the captured **Ds_Signature** parameter. To do this, the library function "createMerchantSignatureNotif ()" must be named with the key obtained from the management module and the **captured Ds_ MerchantParameters parameter,** as shown below:

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

6. Once this is done, it can already be validated if the value of the sent signature matches with the value of the calculated signature, as shown below:

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

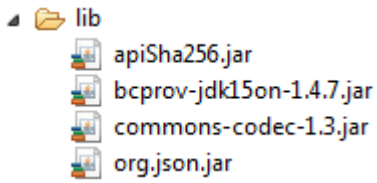

2.2 JAVA Library

Below, the steps that a merchant must follow to use the JAVA library provided by Redsys are described:

1. Import the library, as shown below:

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

   The merchant must include in the project construction route all the libraries (JARs) that are provided:

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

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

3. Capture the request return parameters:

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

Decode the **Ds_MerchantParameters parameter**. To perform the decoding of this parameter, the library function "decodeMerchantParameters ()" must be named, as shown below:

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

Once the "decodeMerchantParameters ()" function call has been executed, the value of any parameter that is likely to be included in the response (Annex **¡Error! Origin of the reference not found**). To obtain the value of a parameter, the "getParameter ()" function of the library must be named with the parameter name to obtain the response code, as shown below:

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

IMPORTANT NOTE: It is important to carry out the validation of all the parameters that are sent in the communication.

4. Validate the **Ds_Signature** parameter. To carry out the validation of this parameter, the signature must be calculated and compared with the captured **Ds_Signature** parameter. To do this, the library function "createMerchantSignatureNotif ()" must be named with the key obtained from the management module and the captured **Ds_** MerchantParameters parameter, as shown below:

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

Once this is done, it can already be validated if the value of the sent signature matches with 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");
}
```

2.3 Library.NET

Below, the steps that a merchant must follow to use the library are described. NET provided by Redsys:

1. Import the library, as shown below:

```
using RedsysAPIPrj;
```

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

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

3. Capture return parameters:

```
string version = Request.QueryString["Ds_SignatureVersion"];
string parms = Request.QueryString["Ds_MerchantParameters"];
string signatureRecibida = Request.QueryString["Ds_Signature"];
```

IMPORTANT NOTE: It is important to carry out the validation of all the parameters that are sent in the communication.

4. Validate the **Ds_Signature** parameter. To carry out the validation of this parameter, the signature must be calculated and compared with the captured **Ds_Signature** parameter. To do this, the library function "createMerchantSignatureNotif ()" must be named with the key obtained from the management module and the **captured Ds_ MerchantParameters** parameter, as shown below:

```
var kc = "sq7HjrUOBfKmC576ILgskD5srU870gJ7";
string signatureCalculada =  r.createMerchantSignatureNotif(kc, parms);
```

Once this is done, it can already be validated if the value of the sent signature matches with 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";
}
```