3D Secure 2 Integration Guide

Document Control

This is a controlled document.

DESCRIPTION	3D Secure 2 Integration Guide		
CREATION DATE	06/10/2021	**CREATED BY**	SecurePay
VERSION	1.5	**DATE UPDATED**	22/10/2024
CHANGES	1.5 - Updated 3D Secure 2 protocol version to 2.2.0 1.4 - State field is now conditional 1.3 - Updated testing section and test cards, section 2.4 - Added descriptions to Trans Status Reasons in appendix 3.2 1.2 - Added Fiserv FDMSA as an acquirer supporting Visa, Mastercard and Amex - Added LSI details for authorisation request 1.1 - Added merchantRiskData under 2.3.1. - Added section 2.3.1.5 merchantRiskData		

Table of Contents

1 Introduction

1.1 What is 3D Secure 2 (3DS2)?

3D Secure 2 (3DS2) is an additional layer of security that aids fraud prevention. A protocol being managed by EMVCo, this is used to verify the identity of the person doing the purchase to ensure that they are the legitimate cardholder even before the payment is done.

This is done by providing the card issuer high volume of contextual data to aid in the decision which authentication flow to be used. The card issuer checks whether the payment being requested is from the legitimate cardholder. This results to a more frictionless experience where customers are verified, and payment then proceeds. In cases where the issuer identifies that the transaction is high risk, they can prompt challenge questions for the customer to answer as an additional verification.

With SecurePay 3DS2 which currently supports protocol version 2.2.0, you will be able to authenticate before submitting payment authorisation requests. Depending on the result of authentication, merchants or issuers can bear responsibility during chargebacks when the payment authorised is found to be fraudulent and not authorised by the cardholder.

To allow you to have the flexibility to decide whether to proceed with the payment authorisation or not, you will receive the Liability Shift Indicator along with other 3DS results. You can then submit your payment authorisation request along with the authentication details via XML API or Direct Post, depending on your integration.

1.2 About this Guide

This guide provides technical information on how to integrate with the SecurePay 3DS2. It is intended for developers who will be configuring their own applications or websites to execute authentication process in your environment using SecurePay 3DS2.

1.3 Requirements for implementation

To utilise 3DS2 on your existing SecurePay eCommerce account, there are 2 steps to complete.

- Firstly, you must be enabled through your acquiring facility for the card schemes you support. Please contact your acquirer to complete this step.
- Secondly, after your acquirer has enabled 3DS2 for you, you will then need to enable 3DS2 on your SecurePay account. You can complete the application steps via your SecurePay account login or contact our support team.

1.4 Supported Integrations

The SecurePay 3DS2 supports the following integration:

- XML API
- Direct Post
- SecurePay API

1.5 Supported Transaction Types

SecurePay 3DS2 supports the following transaction types:

- Payment
- Preauthorisation (including Initial Authorisation)
- Preauthorisation Complete

1.6 Supported Banks and Card Types

SecurePay 3DS2 is currently enabled on the following acquiring bank links with corresponding card brands.

Bank	Card Brands
NAB	Visa, Mastercard
Westpac	Visa, Mastercard, American Express
ANZ	Visa, Mastercard
CBA	Visa, Mastercard, American Express
Fiserv FDMSA	Visa, Mastercard

2 Implementation

2.1 General Information

2.1.1 *Parameter Types and Constraints*

The value format descriptions in the following sections use keys from the following table:

Type	Constraint	Description
String	A	• Alphabetic characters • Value in the element is valid if it only contains characters in the specified set (alphabetic)
	N	• Numeric characters • Value in the element is valid if it only contains characters in the specified set (numeric)
	S	• Special characters • Will be followed with a list of allowed characters. • Value in the element is valid if it only contains characters in the specified set (special characters)
	LEN	• Number of characters in the string • Value in the element is valid if the length of the value is equal to the defined length.
	MINLEN	• Number of characters in the string • Value in the element is valid if the length of the value is equal to the defined length.
	MAXLEN	• Maximum number of characters in the string • Value in the element is valid if the length of the value is less than or equal to the defined maximum length.
Integer	DIGNO	• Number of digits in the integer value • Value in the element is valid if the number of digits in the value is less than or equal to the defined digits number.
	MINVAL	• Minimum numerical value • Value in the element is valid if it is numerically greater than or equal to the defined minimum value.
	MAXVAL	• Maximum numerical value • Value in the element is valid if it is numerically less than or equal to the defined maximum value.

2.2 How to do 3DS2 Authentication

Authentication can be accomplished by using the 3DS2 JavaScript SDK, using either of the two methods below:

2.2.1 *Recommended method of 3DS2 authentication*

2.2.1.1 *Step 1: Create a 3DS2 order.*

A 3DS2 Order will need to be created before 3DS2 authentication can be triggered.

Send a POST message to the endpoint to create an 3DS2 order.

Live URL: https://api.securepay.com.au/services/order-management/v2/payments/orders

Test URL: https://test.api.securepay.com.au/services/order-management/v2/payments/orders

2.2.1.1.1 *Request*

Request Header

Element	Comments	
Authorization	**Description:**	Authorization header. A Base64 encoding of your SecurePay Merchant ID and password joined by a single colon ":"
	Format type:	(No Value)
	Format constraints:	(No Value)
	Validated by SecurePay:	Yes
	Value:	For SecurePay Merchant ID ABC00 with password abc123, corresponding Base64 encoding is Basic QUJDMDA6YWJjMTIz
	Sub-elements:	No

Request Body

Element	Comments	
ip	**Description:**	IP address from which the transaction originated.
	Format type:	String
	Format constraints:	NS (Must contain three periods), MAXLEN=15
	Validated by SecurePay:	Yes
	Value:	E.g: "203.89.101.20"
	Sub-elements:	No
merchantOrderReference	**Description:**	Unique merchant transaction identifier.
	Format type:	String
	Format constraints:	ANS, MINLEN=1, MAXLEN=50 (use of '-' allowed)
	Validated by SecurePay:	Yes
	Value:	E.g: "511428f6-356a-4256-9b3d-3f2da91a971d"
	Sub-elements:	No
amount	**Description:**	Transaction amount in cents. **Note:** Ensure that the amount used during 3DS2 authentication is the same as authorisation.
	Format type:	Integer
	Format constraints:	MINVAL=1
	Validated by SecurePay:	Yes
	Value:	Eg: "620" for $6.20
	Sub-elements:	No
currency	**Description:**	Transaction currency.
	Format type:	String, following ISO 4217
	Format constraints:	A, LEN=3
	Validated by SecurePay:	Yes
	Value:	Eg: "AUD" for Australian Dollars
	Sub-elements:	No
orderType	**Description:**	Order Type.
	Format type:	String
	Format constraints:	Allowed value provided
	Validated by SecurePay:	Yes
	Value:	Value should be "PAYMENT"
	Sub-elements:	No
intents	**Description:**	Purpose of creating the order.
	Format type:	String
	Format constraints:	Allowed value provided
	Validated by SecurePay:	Yes
	Value:	Value should be "THREED_SECURE"
	Sub-elements:	No

2.2.1.1.2 Response

Element	Comments	
orderId	**Description:**	A unique id generated for 3DS2 order.
	Format type:	String
	Format constraints:	ANS, LEN=36
	Value:	E.g: "032f6cfc-5fc2-4fe9-9e70-e463bcc56e08"
	Sub-elements:	No
orderToken	**Description:**	A bearer token used to authenticate calls made from browser. The orderToken expires after 10 minutes.
	Format type:	String
	Format constraints:	ANS
	Value:	(No sample provided)
	Sub-elements:	No
simpleToken	**Description:**	A shortened authentication token used to authenticate calls from browser.
	Format type:	String
	Format constraints:	ANS
	Value:	(No sample provided)
	Sub-elements:	No
amount	**Description:**	Transaction amount in cents returned unchanged from the request. The authentication amount should match the amount on your authorisation request.
	Format type:	Integer
	Format constraints:	MINVAL=1
	Value:	Eg: "620" for $6.20
	Sub-elements:	No
currency	**Description:**	Transaction currency returned unchanged from the request.
	Format type:	String, following ISO 4217
	Format constraints:	A, LEN=3
	Value:	Eg: "AUD" for Australian Dollars
	Sub-elements:	No
orderType	**Description:**	Order Type returned unchanged from the request.
	Format type:	String
	Format constraints:	Allowed value provided
	Value:	Value should be "PAYMENT"
	Sub-elements:	No
status	**Description:**	3DS2 Order Status. Returns NEW after 3DS2 order creation
	Format type:	String
	Format constraints:	Allowed value provided
	Value:	"NEW"
	Sub-elements:	No
merchantId	**Description:**	Merchant ID returned from the request.
	Format type:	String
	Format constraints:	ANS
	Value:	E.g. "ABC00"
	Sub-elements:	No
merchantOrderReference	**Description:**	Unique merchant transaction identifier returned unchanged from the request.
	Format type:	String
	Format constraints:	ANS, MINLEN=1, MAXLEN=50 (use of '-' is permitted)
	Value:	E.g: "511428f6-356a-4256-9b3d-3f2da91a971d"
	Sub-elements:	No

Element	Comments	
threedSecure	**Description:**	Details that will be used towards 3DS2 authentication.
	Format type:	(No Value)
	Format constraints:	(No Value)
	Value:	(No Value)
	Sub-elements:	Yes, see threedSecure.
intents	**Description:**	Purpose of creating the order. Returned unchanged from the request.
	Format type:	String
	Format constraints:	Allowed Value Provided
	Value:	"THREED_SECURE"
	Sub-elements:	No

threedSecure

Element	Comments	
providerClientId	**Description:**	The client ID assigned to the merchant during activation of 3DS2 feature.
	Format type:	String
	Format constraints:	ANS.
	Value:	E.g. "w-123456"
	Sub-elements:	No
sessionId	**Description:**	Unique session ID for an order.
	Format type:	String
	Format constraints:	AN
	Value:	e.g. "MTIxOGY2NDgtODJjZi00MzM5LTgwMmQtMDM0OWRlMWI1MjA2"
	Sub-elements:	No

2.2.1.1.3 Samples

Request


```
{
    "amount": 620,
    "currency": "AUD",
    "ip": "203.89.101.20",
    "merchantOrderReference": "511428f6-356a-4256-9b3d-3f2da91a971d",
    "orderType": "PAYMENT",
    "intents": [
        "THREED_SECURE"
    ]
}
```


Response


```
{
    "orderId": "032f6cfc-5fc2-4fe9-9e70-e463bcc56e08",
    "orderToken": "eyJjdHkiOiJKV1QiLCJlbmMiOiJBMjU2R0NNIiwiYWxnIjoiUlNBLU9BRVAtMjU2In0.mhZKn8fvJtzIusZ54xDfgYun8eG6SzbpIpSvAFu9GrsxO1NY7Tg0u4_ahXK6fALwvr1PGiXsTX_VKG-WY3t5zjKb0G6WP5Qd1EF9uYJmIlMjMTCCk9BQM-pH8EbmpujNPyfNEgsC8Bsdi2JCIIAcgRhlcmZvTxlES-pmFXX1awnLm-oMx5DnHHh1l3AFPktVyx_Mx4Vzic9RnOasxmd9DBA4WYQ47bRSGnIm2ukQXyNGcE7Nm_MMgcVjZuxB2aro78Vy6xh22-4IfU5tbG-uNn_18aCfHMSbakjAo4MV23YhUvoaN9TvwlL35tMos-32aMUF776pxvmS-wtS94PA5w.HTmy8iO6Sv6txQ4L.QXfqKVyl99jibwP-PP9UnZfHBDJErmZj_ey7Ltp7sH3sZQOve3c6lNrPzO-Eriv1h9aTb959V8X2gTcuXe2JwmM9h2D1RhiUWNcJ2xsomkPp_cuHIZsWLEvvcbXcD_AoWpYGz7Nki2ketfptXYnjz0N_Wxu4WHS8ajQVsEKKZwituUqKodjvq8GBQq-lD4SHwEFJmLFJUEKibuuA9H04bQjicqGswljWlbLDnYNbDhbZ1G2IZUhhXFD6d4yfsRvGPQZr9pM9oJ1DwrvD2zxiZs_JmRS1z68_ePe_OeegLQsHUfRdCykZMsq3zk2QoJrGnKzMQyTToCDikep667IBp9mMg2YNQBg5NQGzcAKampe6F_jCaZvMVuBitklzCf6Qt2c9Yy7iTX3JoA6zUovdh2yeGR_Uk88baB0NG1qkQQag5GriTbuW75zfdh9lOewUt1zPKmMh75fLxVPiJbb2YZfk3JayqVd8tU5tALbX6NgW01pY8M0zpK6heJuDIyGGmFbWJDrS4j4YQhs85fHH4UhSs8hACAiZDHwCIzLnOQ6W1d0KKwGYJOJ4MtqPKjFnzg75J1u4mI4ZVLmrPJfglcMX5aZbB4iqldW7Qi_Eqsf577UF_EazJT-87RieKUQsFVCOLhKrhfQM64qI2nZDuuXIMmefTy_saiUjFtkbt0M-yLOPLoQ75mZkm4Zuv7gzDV6sbM8k0nMPm1FykYeJkbSTWS4tPD2W_H2BtsUF4kdmOGTY79nrfQb8LSZPxfKLm2K7GoJLsJck0KS-JqXMx42y5RE7jyq4hbRKBH3s2pKEmj2RsGhxajht6w.AbcTt7rK6UGYH6HdDqoCIg",
    "simpleToken": "wtS94PA5wHTmy8iO6Sv6txQ4LQXfqKVyl99jibwPP9UnZfHBDJErmZjey7Ltp7sH3sZQOve3c6lNrPzOEriv1h9aTb959V8X2gTcuXe2JwmM9h2D1RhiUWNcJ2xsomkPpcuHIZsWLEvvcbXcDAoWp"
    "amount": 620,
    "currency": "AUD",
    "status": "NEW",
    "merchantId": "ABC00",
    "merchantOrderReference": "511428f6-356a-4256-9b3d-3f2da91a971d",
    "threedSecure": {
        "providerClientId": "w-123456",
        "sessionId": "MTIxOGY2NDgtODJjZi00MzM5LTgwMmQtMDM0OWRlMWI1MjA2"
    },
    "intents": [
        "THREED_SECURE"
    ]
}
```


Please note:

The 3DS2 order ID is used to track all authentication requests for a payment or preauthorisation transaction. Each authentication request is kept in our system and includes the result of the authentication.

The 3DS2 order ID can be reused for maximum of three authentication attempts. Succeeding attempts to use the same 3DS order will result to authentication request being rejected. This is excluding the challenge attempts done by customers.

2.2.1.2 *Step 2: Load 3DS2 JavaScript*

The **securepay-threeds.js** client library should be included in your HTML source as shown in the sample code:

Live Environment:

```
<script th:id="sp-threeds-js" th:src=" https://api.securepay.com.au/threeds-js/securepay-threeds.js" type="text/javascript"></script>
```

Test Environment:

```
<script th:id="sp-threeds-js" th:src=" https://test.api.securepay.com.au/threeds-js/securepay-threeds.js" type="text/javascript"></script>
```

The 3DS2 will need a challenge window for some calls to get customer's input for the challenge. Hence create an element for this window to be published when required.

```
<iframe id=”3ds-v2-challenge-iframe” name=”3ds-v2-challenge-iframe” style=”width: 500px; height: 500px; visibility:hidden;”>
```

2.2.1.3 *Step 3: Initialise 3DS2 JavaScript*

Once loaded, the script can be initialised with the following parameters.

This can be done during loading of your checkout page.

Element	Comments	
`clientId`	**Description:**	Client Id that is assigned by 3DS2 provider to merchant. The value can be obtained from create order response, “threedSecure.providerClientId”.
	Format type:	String
	Value:	Value obtained from “threedSecure.providerClientId”.
	Sub-elements:	No
`iframe`	**Description:**	The iframe element that can be used to display 3DS2 challenge form. The iframe element should be defined in the checkout/payment page and passed into the JavaScript during initialization.
	Format type:	String
	Value:	For example: In Add an iframe element, value will be: **document.getElementById(“3ds-v2-challenge-iframe”)**
	Sub-elements:	No
`token`	**Description:**	An order token that was returned in create order response as “orderToken”.
	Format type:	String
	Value:	Value obtained from “orderToken”.
	Sub-elements:	No

Element	Comments	
simpleToken	**Description:**	A simple authentication token that was returned in create order response as "simpleToken".
	Format type:	String
	Value:	Value obtained from "simpleToken".
	Sub-elements:	No
threeDSSessionId	**Description:**	A unique session id that was returned in create order response.
	Format type:	String
	Value:	Value obtained from "threedsecure.sessionId".
	Sub-elements:	No
onRequestInputData	**Description:**	A JS call-back method to provide 3DS2 data for authentication.
	Sub-elements:	Yes. See "onRequestInputData callback"
onThreeDSResultsResponse	**Description:**	A JS call-back method to update merchant when 3DS2 authentication has completed.
	Sub-elements:	Yes. See "onThreeDSResultsResponse callback"
onThreeDSError	**Description:**	A JS call-back method to update merchant when there is an error during the authentication process.
	Sub-elements:	Yes. See "onThreeDSError callback"

Below is an example of how the scripts can be initialised.

```
var sp3dsConfig = {
    clientId: providerClientId,
    iframe: iframeElement,
    token: orderToken,
    simpleToken: simpleToken,
    threeDSSessionId: sessionId,
    onRequestInputData: onRequestInputDataCallback,
    onThreeDSResultsResponse: onThreeDSResultsResponseCallback,
    onThreeDSError: onThreeDSErrorCallback
};

var securePayThreedsUI = new window.securePayThreedsUI();
window.securePayThreedsUI = securePayThreedsUI;
securePayThreedsUI.initThreeDS(sp3dsConfig);
```

Please note:
If any of the config objects are missing, error "VALIDATION_ERROR - missing mandatory config" will be received.
If the clientId provided is incorrect, error "INITIALIZATION_ERROR - JavaScript initialization error" will be received.
If invalid type of iframe is used, then "WRONG_CONFIG_TYPE_ERROR - config is not in expected format" will be received.

2.2.1.4 *Step 4: Start 3DS2 Process.*

Once the SP 3DS2 JavaScript has been initialised, the authentication process can be triggered.

This can be done when payment or checkout commenced.

Example to start 3DS2 authentication process:

```
var spThreedsUI = window.securePayThreedsUI;
spThreedsUI.startThreeDS();
```

During this step, onRequestInputData callback will be used to collect necessary information to prepare an 3DS2 authentication request.

Please note:
If 3DS2 authentication process is triggered before the initialisation has completed,
error GENERIC_THREEDS_START_ERROR - Failed to start Threeds Authentication will be received.

2.2.1.5 *Step 5: Handle Authentication Outcome*

Once the 3DS2 authentication process has finished, a callback will be done via onThreeDSResultsResponse to notify the authentication outcome. This callback includes a Liability Shift Indicator (whether you or the card issuer is liable for fraudulent chargebacks) and other supporting parameters.

See “onThreeDSResultsResponse callback”.

If an error has occurred during the authentication process, a callback will be done via onThreeDSError. See “onThreeDSError callback”.

2.2.1.6 *Step 6: Trigger Authorisation*

Once the 3DS2 authentication process has finished and you have received the authentication outcome, you can now trigger your payment authorisation or preauthorisation. To send the 3DS2 details with your payment authorisation or preauthorisation request, include the 3DS2 orderId in the request. This allows our system to include the resulting 3DS2 details with your transaction before sending to the acquirer.

Please refer to the XML API Integration guide or Direct Post Integration guide depending on your integration method to see how 3DS2 OrderId is used in payment or preauthorisation.

Liability Shift Indicator in Authorisation Request

To confirm that you accept the value returned for the Liability Shift Indicator in the **Authentication Outcome**, it is recommended to send an optional `liabilityShiftIndicator` (LSI) value in the authorisation or preauthorisation request.

If you do not include an LSI value in the authorisation or preauthorisation request and the liability has not shifted (i.e., you are liable), the transaction will be declined.

Accepted values are Y or N.

The purpose of this optional field in the authorisation/preauthorisation request is to:

1) Provide control (and explicit acceptance) of whether to proceed with payments even if the 3DS2 authentication did not pass.

2) Ensure that there has been no modification to the Liability Shift Indicator you receive in the authentication outcome, and your decision to continue with payment (or cease the payment flow) is based on correct authentication information.

Example #1: After your customer completes the 3DS2 authentication process, if the Liability has not been shifted, it means that you are responsible for any potential fraudulent chargebacks. If you decide to proceed with the payment based on your business reasoning, you need to acknowledge and accept that the liability has not been shifted by placing a 'N' in the liabilityShiftIndicator field in the authorization request. This will allow the payment to proceed while acknowledging that you accept the liability for any potential chargebacks.

Example #2: Your customer completes the 3DS2 authentication process and the Liability was not shifted. However, a malicious customer modifies the authentication outcome, and you receive an indicator that the liability was shifted to the issuer. You proceed with the payment as you understood it was shifted (but you are still liable for fraudulent chargebacks). As you have sent no value in the liabilityShiftIndicator field, the payment is rejected, protecting you and your customer.

The LSI value in the authorisation request must match the Liability Shift Indicator value we have stored for the Authentication outcome. This gives control of proceeding or not proceeding with unauthenticated transactions.

To see more details on how the field is sent, please refer to the integration guide for your specific integration in the developer resources on the SecurePay website.

Please refer to the table below for the transaction flow scenarios relating to the liabilityShiftIndicator field:

Optional Liability Shift Indicator Processing:

SecurePay | Liability shift indicator from Authentication Outcome	Merchant | Liability Shift Indicator Sent in Authorisation Request 'liabilityShiftIndicator'	Result
Y	Y	Payment continues for processing
N	N	
Y	Not provided	
N	Y	3DS2 Payment is declined • Response Code: 517 • Response Text: Liability Shift Indicator Error
Y	N	
N	Not provided	

2.2.2 *Alternate Method of 3DS2 authentication*

This method allows the initialise 3DS2 JS and trigger 3DS2 authentication steps together using the method initAndStartThreeDS.

Please note that this may cause an increase in the load time. Also, this can only be done once you have filled all the 3DS2 required fields.

Step 1: Initiate a 3DS2 order – same with recommended method.

Step 2: Load the SecurePay 3DS2 JavaScript client library – same with recommended method.

Step 3: Initialise and trigger 3DS2 authentication together using the method initAndStartThreeDS

Example of initialising and starting 3DS2.

```
var sp3dsConfig = {
    clientId: providerClientId,
    iframe: iframeElement,
    token: orderToken,
    simpleToken: simpleToken,
    threeDSSessionId: sessionId,
    onRequestInputData: onRequestInputDataCallback,
    onThreeDSResultsResponse: onThreeDSResultsResponseCallback,
    onThreeDSError: onThreeDSErrorCallback
};


var securePayThreedsUI = new window.securePayThreedsUI();
window.securePayThreedsUI = securePayThreedsUI;
securePayThreedsUI.initAndStartThreeDS(sp3dsConfig);
```

Step 4: Handle the authentication outcome – same with recommended method

Step 5: Trigger Authorisation.

2.3 Callback methods

2.3.1 *onRequestInputData callback*

onRequestInputData callback will collect relevant information from customer to be used in an 3DS2 Authentication request. onRequestInputData callback is executed prior to an HTTP request initiated by 3DS2 javascript. The callback should return an object with the following details.

Element	Comments	
`cardholderInfo`	**Description:**	Card holder details associated with the issued card used for the transaction. Mandatory field for authentication to proceed.
	Format type:	(No Value)
	Validated by SecurePay:	Yes
	Value:	(No Value)
	Sub-elements:	Yes. Please see cardholderInfo.
`accountData`	**Description:**	Additional Information related to the card holder. Mandatory field for authentication to proceed.
	Format type:	(No Value)
	Validated by SecurePay:	Yes
	Value:	(No Value)
	Sub-elements:	Yes. Please see accountData.
`billingAddress`	**Description:**	Billing Address associated with the card holder. Mandatory field for authentication to proceed.
	Format type:	(No Value)
	Validated by SecurePay:	Yes
	Value:	(No Value)
	Sub-elements:	Yes. Please see Address.
`shippingAddress`	**Description:**	Shipping Address used for the transaction. Non-mandatory field.
	Format type:	(No Value)
	Validated by SecurePay:	Yes
	Value:	(No Value)
	Sub-elements:	Yes. Please see Address.
`threeDSInfo`	**Description:**	Additional information associated with the authentication request as required by different card brands. Required for Visa cards only.
	Format type:	(No Value)
	Validated by SecurePay:	Yes
	Value:	(No Value)
	Sub-elements:	Yes. Please see threeDSInfo.
`merchantRiskData`	**Description:**	The Merchant Risk Data contains information about the specific purchase by the Cardholder as required by different card brands. Optional for Amex Cards.
	Format type:	(No Value)
	Validated by SecurePay:	Yes
	Value:	(No Value)
	Sub-elements:	Yes. Please see merchantRiskData.

2.3.1.1 cardholderInfo

Element	Comments	
cardholderName	**Description:**	Cardholder name. The full name of the cardholder.
	Format type:	String
	Format constraints:	ANS, MINLEN=2, MAXLEN=45 (allows space " ")
	Validated by SecurePay:	Yes
	Value:	E.g: John Smith
	Sub-elements:	No
cardNumber	**Description:**	Card Number. The cardholder account number or PAN used to complete the purchase.
	Format type:	String
	Format constraints:	N, MINLEN=13, MAXLEN=16
	Validated by SecurePay:	Yes
	Value:	E.g: 4444333322221111
	Sub-elements:	No
cardExpiryMonth	**Description:**	Card expiry month.
	Format type:	String
	Format constraints:	N, LEN=2
	Validated by SecurePay:	Yes
	Value:	E.g: 10
	Sub-elements:	No
cardExpiryYear	**Description:**	Card expiry year.
	Format type:	String
	Format constraints:	N, LEN=2
	Validated by SecurePay:	Yes
	Value:	E.g: 25
	Sub-elements:	No

2.3.1.2 accountData

Element	Comments	
mobilePhone	**Description:**	The mobile phone number associated with the cardholder. Optional field.
	Format type:	(No Value)
	Format constraints:	(No Value)
	Validated by SecurePay:	Yes
	Value:	(No Value)
	Sub-elements:	Yes. See ContactNumber.
workPhone	**Description:**	The work phone number associated with the cardholder. Optional field.
	Format type:	(No Value)
	Format constraints:	(No Value)
	Validated by SecurePay:	Yes
	Value:	(No Value)
	Sub-elements:	Yes. See ContactNumber.
homePhone	**Description:**	The home phone number associated with the cardholder. Optional field.
	Format type:	(No Value)
	Format constraints:	(No Value)
	Validated by SecurePay:	Yes
	Value:	(No Value)
	Sub-elements:	Yes. See ContactNumber.

Element	Comments	
emailAddress	**Description:**	The email address associated with the cardholder. Mandatory field for authentication to proceed.
	Format type:	String
	Format constraints:	ANS, MAXLEN=254
	Validated by SecurePay:	Yes
	Value:	johnsmith@somedomain.com
	Sub-elements:	No

2.3.1.2.1 ContactNumber

Element	Comments	
cc	**Description:**	The country code of the contact number. Mandatory if subscriber is provided.
	Format type:	String
	Format constraints:	NS, (Only a prefix of '+' is allowed for special character), MINLEN=1, MAXLEN=3
	Validated by SecurePay:	Yes
	Value:	Eg. "+61" for Australia
	Sub-elements:	No
subscriber	**Description:**	Subscriber number. Mandatory if cc is provided and should follow string length according to country code.
	Format type:	String
	Format constraints:	N, MAXLEN=15
	Validated by SecurePay:	Yes
	Value:	Eg. "432123456" for Australia with cc +61
	Sub-elements:	No

2.3.1.3 Address

Element	Comments	
streetAddress	**Description:**	Street address. This is mandatory for billing address.
	Format type:	String
	Format constraints:	ANS, MAXLEN=50
	Validated by SecurePay:	Yes
	Value:	Eg. "1/2 Grover Street"
	Sub-elements:	No
detailedStreetAddress	**Description:**	Detailed street address. This is an optional field for billing and shipping address.
	Format type:	String
	Format constraints:	ANS, MAXLEN=50
	Validated by SecurePay:	Yes
	Value:	Eg. "Destination drive"
	Sub-elements:	No
detailedStreetAddressAdditional	**Description:**	Additional detailed street address. This is an optional field for billing and shipping address.
	Format type:	String
	Format constraints:	ANS, MAXLEN=50
	Validated by SecurePay:	Yes
	Value:	Eg. "Camberwell"
	Sub-elements:	No
city	**Description:**	City.
	Format type:	String
	Format constraints:	AN, MAXLEN=50
	Validated by SecurePay:	Yes
	Value:	Eg. "Melbourne"
	Sub-elements:	No

Element	Comments	
state	**Description:**	Conditional field. The state or province as per Country subdivision code defined in ISO 3166-2. Required unless State is not applicable for this country i.e. Aruba. Do not send field if not applicable.
	Format type:	String
	Format constraints:	AN, MAXLEN=3
	Validated by SecurePay:	Yes
	Value:	Eg. "VIC"
	Sub-elements:	No
country	**Description:**	Country code. Can contain the 3-digit numeric ISO code or the 2 or 3 alpha character ISO code
	Format type:	String
	Format constraints:	N, LEN = 3 or A, MINLEN = 2, MAXLEN = 3
	Validated by SecurePay:	Yes
	Value:	Eg. "AU"
	Sub-elements:	No
zipCode	**Description:**	ZIP or postal code.
	Format type:	String
	Format constraints:	AN, MAXLEN=16
	Validated by SecurePay:	Yes
	Value:	Eg. "3000"
	Sub-elements:	No

2.3.1.4 threeDSInfo

Element	Comments	
threeDSReqAuthMethodInd	**Description:**	Method used to authenticate the user with the client web application. Mandatory for Visa cards.
	Format type:	String
	Format constraints:	LEN=2, Accepted Values: "01" - NO_AUTHENTICATION "02" - OWN_CREDENTIALS "03" - FEDERATED_ID "04" - ISSUER_CREDENTIALS "05" - THIRD_PARTY "06" - FIDO "07" - FIDO_ASSURANCE_DATA_SIGNED "08" - SRC_ASSURANCE_DATA
	Validated by SecurePay:	Yes
	Value:	E.g. "01" for NO_AUTHENTICATION
	Sub-elements:	No

2.3.1.5 merchantRiskData

Element	Comments	
deliveryTimeframeType	**Description:**	The merchandise delivery timeframe. Optional for Amex cards.
	Format type:	String
	Format constraints:	LEN=2, Accepted Values: "01" – Electronic delivery "02" – Same Day shipping "03" – Overnight shipping "04" – Two day or more shipping
	Validated by SecurePay:	Yes
	Value:	E.g. "01" for Electronic Delivery
	Sub-elements:	No

Element	Comments	
reOrderType	**Description:**	Indicates whether the cardholder is reordering previously purchased merchandise. Optional for Amex cards.
	Format type:	String
	Format constraints:	LEN=2, Accepted Values: "01" – First time ordered "02" – Reordered
	Validated by SecurePay:	Yes
	Value:	E.g. "01" for First time ordered
	Sub-elements:	No
shippingMethodType	**Description:**	Indicates shipping method chosen for the transaction. You must choose the code that most accurately describes the cardholder's specific transaction. If one or more items are included in the sale, use the type for the physical goods, or if all digital goods, use the type that describes the most expensive item. Optional for Amex cards.
	Format type:	String
	Format constraints:	LEN=2, Accepted Values: "01" - Ship to cardholder's billing address "02" - Ship to another verified address on file with merchant "03" - Ship to address that is different than the cardholder's billing address "04" - "Ship to Store" / Pick-up at local store (Store address shall be populated in shipping address fields) "05" - Digital goods (includes online services, electronic gift cards and redemption codes) "06" - Travel and Event tickets, not shipped "07" - Other (for example, Gaming, digital services not shipped,
	Validated by SecurePay:	Yes
	Value:	E.g. "01" for Ship to cardholder's billing address
	Sub-elements:	No
deliveryEmailAddress	**Description:**	For Electronic delivery, the email address to which the merchandise was delivered. Conditional -Mandatory for Amex Card when shippingMethodType = "05"
	Format type:	String
	Format constraints:	ANS, MAXLEN=254
	Validated by SecurePay:	Yes
	Value:	johnsmith@somedomain.com
	Sub-elements:	No

2.3.2 *onThreeDSResultsResponse callback*

The onThreeDSResultsResponse callback provides the outcome of the 3DS2 authentication process. When the 3DS2 authentication process has finished, this callback provides a liability shift indicator and other supporting parameters.

Element	Comments	
liabilityShiftIndicator	**Description:**	Liability Shift Indicator. This indicates whether the chargeback liability is shifted to the issuer. Refer to 3.1 Appendix A for more information.
	Format type:	String
	Format constraints:	A, LEN=1, Values: Y - Liability for relevant Chargeback reason codes is shifted to the issuer. N - Liability for relevant Chargeback reason codes is not shifted to the issuer.
	Value:	E.g. "Y"
	Sub-elements:	No
transStatus	**Description:**	Transaction Status. This value indicates whether a transaction qualifies as an authenticated transaction. It is received from the 3DS2 server and follows the EMVCo specification for 3DS2 data elements.
	Format type:	String
	Format constraints:	A, LEN=1
	Value:	E.g. "Y"
	Sub-elements:	No
transStatusReason	**Description:**	Transaction Status Reason. It provides additional information to support the transStatus. It is received from the 3DS2 server and follows the EMVCo specification v2.2.0 for 3DS2 data elements. See appendix 3.2 for more details on the descriptions.
	Format type:	String
	Format constraints:	LEN=2, values: 01 - Card authentication failed 02 - Unknown Device 03 - Unsupported Device 04 - Exceeds authentication frequency limit 05 - Expired card 06 - Invalid card number 07 - Invalid transaction 08 - No Card record 09 - Security failure 10 - Stolen card 11 - Suspected fraud 12 - Transaction not permitted to cardholder 13 - Cardholder not enrolled in service 14 - Transaction timed out at the ACS 15 - Low confidence 16 - Medium confidence 17 - High confidence 18 - Very High confidence 19 - Exceeds ACS maximum challenges 20 - Non-Payment transaction not supported 21 - 3RI transaction not supported
	Value:	E.g. "01" for Card Authentication failed
	Sub-elements:	No
eci	**Description:**	E-Commerce Indicator/Security Level Indicator (SLI). This is a two-digit value returned by the Directory Server that indicates the result of the authentication.
	Format type:	String
	Format constraints:	LEN=2
	Value:	E.g. "05"
	Sub-elements:	No

Element	Comments	
authenticationValue	**Description:**	Authentication Value. Also known as Cardholder Authentication Verification Value (CAVV). This is a value returned by the Issuer ACS that performed the authentication.
	Format type:	String
	Format constraints:	(no value)
	Value:	E.g. "owbocveo5tA7DyHugsy+79oukPI="
	Sub-elements:	No
cardDescription	**Description:**	Card Brand Description used during authentication.
	Format type:	String
	Format constraints:	(no value)
	Value:	E.g. "Visa"
	Sub-elements:	No

2.3.3 *onThreeDSError callback*

The onThreeDSError callback is used when there is a failure during the 3DS2 authentication process. It provides details of the failure/error. In each error, the following information will be provided. Refer to the SecurePay Response Codes document via the Developer documentation link on the SecurePay website for more information about 3DS2 errors.

Element	Comments	
id	**Description:**	A unique error id which can be used for troubleshooting.
	Format type:	String
	Format constraints:	ANS
	Value:	E.g. "1a909ec1-c96c-4ced-a471-d145a0e517ef"
	Sub-elements:	No
code	**Description:**	An error code indicating the error.
	Format type:	String
	Format constraints:	N
	Value:	E.g. "176"
	Sub-elements:	No
detail	**Description:**	Error details.
	Format type:	String
	Format constraints:	AN
	Value:	E.g. "Merchant Not Enrolled in EMV 3D Secure."
	Sub-elements:	No

2.4 Testing

To test 3DS2 in the sandbox environment, use merchant ID of 5AR00. You are not able to use your Merchant ID and must use merchant ID 5AR00. Please contact support if you need an alternate testing set up.

To use the test cards below you must use:

- Merchant ID: **5AR00**
- Card holder name - value must either be **Test Card** or an **empty value**
- Expiry date (YYMM) - value must either be **2508** or an **empty value**

The Merchant ID, cardholder name, and expiry date must be set as above otherwise the authentication will fail.

Test cards and outcomes:

Authentication Outcome	transStatus	Card Type	Card Number	Challenge password	ECI
Frictionless	Y	VISA	4100000000000100	N/A	05
Frictionless	Y	MasterCard	5100000000000107	N/A	02
Frictionless	Y	AMEX	340000000000108	N/A	05
Challenge	Y	VISA	4100000000005000	123456	05
Challenge	Y	MasterCard	5100000000005007	123456	02
Challenge	Y	AMEX	340000000005008	123456	05
Challenge failed	N	VISA	4100000000300005	111111	Per card scheme
Challenge failed	N	MasterCard	5100000000300002	111111	Per card scheme
Challenge failed	N	AMEX	340000000300003	111111	Per card scheme
Unavailable	U	VISA	4100000000400003	N/A	Per card scheme
Unavailable	U	MasterCard	5100000000400000	N/A	Per card scheme
Unavailable	U	AMEX	340000000400001	N/A	Per card scheme
Rejected	R	VISA	4100000000500000	N/A	Per card scheme
Rejected	R	MasterCard	5100000000500007	N/A	Per card scheme
Rejected	R	AMEX	340000000500008	N/A	Per card scheme
Attempted	A	VISA	4100000000100009	N/A	06
Attempted	A	MasterCard	5100000000100006	N/A	01
Attempted	A	AMEX	340000000100007	N/A	06

3 Appendices

3.1 Appendix A: Liability Shift Indicator

SecurePay receives eci (SLI) and authentication value (CAVV) from the 3D Secure server as part of the authentication process. SecurePay uses the value of these parameters along with the card brand used during authentication to determine if the chargeback liability shifts to the issuer or stays with the merchant. The shift indicates who will be responsible for relevant fraudulent chargebacks. Along with the transStatus, transStatusReason, eci and authentication value, the Liability Shift Indicator is included in the response callback for merchant visibility as a guide, allowing the flexibility to decide whether to proceed with the payment or not.

Note: Liability shift according to scheme rules for relevant chargeback reason codes only.

Liability Shift Indicator Matrix:

Card Brand	Authentication Value Present	eci	Trans Status	Authentication Result	Liability Shift Indicator
MasterCard	Yes	02	Y	Fully Authenticated	Y (Liability shifts to Issuer)
MasterCard	Yes	01	A	Attempted Authentication	Y (Liability shifts to Issuer)
MasterCard	No	00	N	Not Authenticated	N (Liability stays with Merchant)
MasterCard	No	00	U	Unable to authenticate.	N (Liability stays with Merchant)
MasterCard	No	00	R	Rejected	N (Liability stays with Merchant)
MasterCard	Yes	07	Y	Fully Authenticated	Y (Liability shifts to Issuer)
Visa	Yes	05	Y	Fully Authenticated	Y (Liability shifts to Issuer)
Visa	Yes	06	A	Attempted Authentication	Y (Liability shifts to Issuer)
Visa	No	06	A	Attempted Authentication	N (Liability stays with Merchant)
Visa	No	07	N	Not Authenticated	N (Liability stays with Merchant)
Visa	No	07	R	Rejected	N (Liability stays with Merchant)
Visa	Yes	07	Y	Fully Authenticated	N (Liability stays with Merchant)
Amex	Yes	05	Y	Fully Authenticated	Y (Liability shifts to Issuer)
Amex	Yes	06	A	Attempted Authentication	Y (Liability shifts to Issuer)
Amex	No		N	Not Authenticated	N (Liability stays with Merchant)

Amex	No		R	Rejected	N (Liability stays with Merchant)
Amex	No	07	U	Unable to authenticate	N (Liability stays with Merchant)

3.2 Appendix B: TransStatusReason

As part of the response for the 3DS2 result, a Transaction Status Reason (`transStatusReason`) will be included. The TransStatusReason provides information on why the Transaction Status (`transStatus`) field has the specified value.

Trans Status Reason Code	Detail	Description
1	Card authentication failed	If this occurs multiple times for the same card, contact your Acquirer.
2	Unknown Device	The device the cardholder used for authentication is not the same as the registered device. The cardholder needs to use the same device. If this information is not known, then the cardholder will need to contact their card issuer.
3	Unsupported Device	The cardholder used a device that the ACS does not support e.g., unsupported OS, device not considered secure.
4	Exceeds authentication frequency limit	The maximum limit of authentications has been exceeded. Wait to try again or try a different card. If issue persists, then the cardholder will need to contact the card Issuer.
5	Expired card	Please check the expiry date of your card and try again.
6	Invalid card number	The card number is invalid. Please check and re-enter the details or try a different card.
7	Invalid transaction	There is an issue with the transaction. The cardholder will need to contact their card Issuer.
8	No Card record	The card does not support 3DS or is not enrolled in 3DS from the Issuer. Please try a different card.
9	Security failure	There is a security failure on this card. Please try a different card.
10	Stolen card	Records show that the card used has been flagged as stolen. The cardholder will need to contact their card Issuer.
11	Suspected fraud	Suspected fraudulent activity has been detected. The cardholder will need to contact their card Issuer.
12	Transaction not permitted to cardholder	This type of transaction is not permitted. The cardholder will need to contact their card Issuer.

Trans Status Reason Code	Detail	Description
13	Cardholder not enrolled in service	The card used is currently not enrolled for 3DS.
14	Transaction timed out at the ACS	An authentication response has not been received within a given time, please try again shortly. If the problem persists, the cardholder will need to contact their card issuer.
15	Low confidence	
16	Medium confidence	
17	High confidence	
18	Very High confidence	
19	Exceeds ACS maximum challenges	Number of failed challenges exceeded the maximum set by the ACS. Please wait and try again or try a different card.
20	Non-Payment transaction not supported	ACS does not allow/support payment transactions (if received on a PA - Payment transaction).
		ACS does not allow/support non-payment transactions (if received on a NPA - Non Payment transaction).
21	3RI transaction not supported	ACS does not allow/support merchant-initiated transactions.
22 - 79	Reserved for EMVCo newer 3DS2 versions	
80 - 99	Reserved for DS use	