3-D Secure for API

Suggest Edits (/web/20191207095936/https://developers.bluesnap.com/docs/3-d-secure-for-api/edit)

This guide provides information for using 3-D Secure 2.0 with the BlueSnap API.

- For general information about 3-D Secure and 3-D Secure 2.0, refer to the 3-D Secure Guide (https://web.archive.org/web/20191207095936/https://support.bluesnap.com/docs/3d-secure).
- For answers to specific questions related to BlueSnap and 3-D Secure 2.0, PSD2, and SCA, refer to our 3-D Secure 2.0 FAQs page (https://web.archive.org/web/20191207095936/https://support.bluesnap.com/docs/3ds2-faqs).

This guide provides the following information:

- 3-D Secure in Hosted Payment Fields
- 3-D Secure in Client-Side Encryption
- 3-D Secure with Plain Text cards
- 3-D Secure with Returning Shoppers
- Processing payments with data from external Merchant Plug-In (MPI)
- Supported 3-D Secure object properties
- Error codes
- Authentication results

> **Note**
>
> You don't have to apply 3-D Secure to all transactions. If you want to apply 3DS only to EU-issued cards for PSD2, but not US-issued cards, you can do that. Refer to Step 5 (https://web.archive.org/web/20191207095936/https://developers.bluesnap.com/v8976-Basics/docs/3-d-secure-for-api#section-step-5-determine-3-d-secure-result) for examples.

> **Important**
>
> Merchant-initiated transactions are out of scope for 3-D Secure. To make sure your transaction is identified as merchant-initiated, you must use the Merchant Managed Subscription API. Refer to Create Merchant-Managed Subscription (https://web.archive.org/web/20191207095936/https://developers.bluesnap.com/v8976-JSON/docs/create-merchant-managed-subscription) and Create Merchant-Managed Subscription Charge (https://web.archive.org/web/20191207095936/https://developers.bluesnap.com/v8976-JSON/docs/create-merchant-managed-subscription-charge) for details.
>
> BlueSnap applies this out-of-scope exemption to applicable transactions only for merchants who are the BlueSnap Subscription Engine (https://web.archive.org/web/20191207095936/https://support.bluesnap.com/docs/subscription-options) or our merchant-managed subscription (https://web.archive.org/web/20191207095936/https://support.bluesnap.com/docs/subscription-options#section-merchant-managed-subscriptions) feature. If you handle your own **subscriptions** (that is, you are sending straight auth-capture requests ***not*** using the BlueSnap Subscription Engine or our merchant-managed subscription feature), follow the information in the Rules for MITs (https://web.archive.org/web/20191207095936/https://support.bluesnap.com/docs/3d-secure#section-rules-for-mits) for more information.

Before you begin

- Make sure you've implemented the latest version of Hosted Payment Fields (/web/20191207095936/https://developers.bluesnap.com/v8976-Tools/docs/hosted-payment-fields) or Client-Side Encryption (/web/20191207095936/https://developers.bluesnap.com/docs/client-side-encryption).
- Enable 3-D Secure
  To get started, contact Merchant Support (https://web.archive.org/web/20191207095936/https://bluesnap.zendesk.com/hc/en-us/requests/new?ticket_form_id=360000127087) to request that BlueSnap enable 3DS for your account. After it has been enabled, you can activate it in your BlueSnap Console by going to **Settings > Fraud Settings** and selecting **Enable 3D Secure**.
- Enable IPNs for your account (https://web.archive.org/web/20191207095936/https://support.bluesnap.com/docs/ipn-setup). When the shopper's account is debited for the transaction, the Charge IPN is sent with the `3DStatus` parameter, which verifies the 3-D Secure result.

3-D Secure in Hosted Payment Fields

If you're using Hosted Payment Fields, implementing 3DS consists of these steps:

- Step 1: Prevent your shopper from submitting the form during 3-D Secure setup
- Step 2: Enable 3-D Secure usage in the transaction
- Step 3: Account for 3-D Secure errors
- Step 4: Create 3-D Secure object with transaction data
- Step 5: Determine 3-D Secure result
- Step 6: Process the transaction with Hosted Payment Fields token

Step 1: Prevent your shopper from submitting the form during 3-D Secure setup

If the shopper submits the payment form while BlueSnap is setting up 3DS, an error may occur. You must let BlueSnap deactivate the submit button while setting up 3DS. To do this, add the `data-bluesnap` attribute to your submit button element and set the value to `submitButton`, as shown here

HTML ()

```
<button data-bluesnap="submitButton" type="submit" id="submit-button">Pay Now</button>
```

> **Important!**
> You must define the `data-bluesnap` value exactly as shown above; otherwise, BlueSnap cannot properly deactivate the button.

Step 2: Enable 3-D Secure usage in the transaction

Enable 3DS usage in the transaction by adding the property `'3DS'` to `bsObj` and set its value to `true`.

JavaScript ()

```
var bsObj = {
  ...
  '3DS': true
};
```

Step 3: Account for 3-D Secure errors

BlueSnap passes 3DS error codes and descriptions (shown here) to your `onError` callback function as the `errorCode` and `errorDescription` parameters. Update your `onError` callback function to account for these errors.

Step 4: Create 3-D Secure object with transaction data

Create a 3DS object with at least the transaction amount and currency, as shown in the code below.
If you want to submit the credentials and activate 3DS, pass your object as an additional parameter to `bluesnap.hostedPaymentFieldsSubmitData` (shown in Step 5). You can also decide not to activate 3DS and not pass the object.

When the shopper hits the submit button, the data you provide within your object is sent to the issuer during the lookup and they decide if shopper authentication is required. In some cases, the issuer may authenticate the transaction (and assume fraud chargeback liability) without requiring the shopper to enter their credentials in the popup screen, making the 3DS flow entirely transparent.

> ☑ **Provide as much transaction data as possible**
>
> Provide as much transaction data as possible to improve the issuer's ability to authenticate the transaction without requiring the shopper to enter their credentials during checkout (refer to Supported 3-D Secure object properties).

JavaScript ()

```
var threeDSecureObj = {
  amount: 35.75,
  currency: 'USD'
  ...
};
```

Step 5: Determine 3-D Secure result

BlueSnap passes the 3DS result to your purchase function callback as `callBack.threeDSecure.authResult`, which has one of the values shown here. Configure your purchase function callback to determine the 3DS result and take appropriate action. (Refer to the code below for a simple `bluesnap.hostedPaymentFieldsSubmitData` call with the changes from Steps 4 and 5.)

> ☑ **Note**
>
> `callBack.threeDSecure` also includes `threeDSecureReferenceId`, a reference to the 3DS authentication.

The code calls `bluesnap.hostedPaymentFieldsSubmitData` with a simple purchase callback function. Here you can decide if you want to activate 3DS by either including the `threeDSecureObj` in `bluesnap.hostedPaymentFieldsSubmitData` or you can decide not to activate 3DS and not pass the object.

with threeDSecureObj () | without threeDSecureObj ()

```
bluesnap.hostedPaymentFieldsSubmitData(function(callback) {
        if (null != callback.cardData) {
                console.log('Card type is ' + callback.cardData.ccType +
               ', Last 4 digits are ' + callback.cardData.last4Digits +
               ', Exp is ' + callback.cardData.exp +
               ', Issuing Country is ' + callback.cardData.issuingCountry +
               ', 3D Secure result is ' + (callback.threeDSecure != null ? callback.threeDSecure.authResult : null));
               // submit form to server & process transaction
        } else {
               .
               .
               .
        }
}, threeDSecureObj);
```

Step 6: Process the transaction with Hosted Payment Fields token

Process 3-D Secure transactions (or save shopper payment details) from your server the same way you handle regular card payments—by including your token within the `pfToken` property in the request. Refer to the Hosted Payment Fields guide (/web/20191207095936/https://developers.bluesnap.com/v8976-Tools/docs/hosted-payment-fields#section-in-the-payment-api) for Payment API code samples with `pfToken`.

Back to Top

3-D Secure in Client-Side Encryption

If you're using Client-Side Encryption, implementing 3-D Secure consists of these steps:

- Step 1: Add data-bluesnap attributes
- Step 2: Obtain 3-D Secure token for the session
- Step 3: Call bluesnap.init3DS to set up 3-D Secure
- Step 4: Create a 3-D Secure object with transaction data
- Step 5: Call bluesnap.encrypt with 3-D Secure object and callback function
- Step 6: Process the transaction with encrypted data

Step 1: Add data-bluesnap attributes

Add `data-bluesnap` attributes to the expiration month and year fields on your checkout form and set the values to **expirationMonth** and **expirationYear**, respectively (shown below). This allows BlueSnap to access these fields during checkout, which are required as part of the 3DS authentication flow.

HTML ()

```
<div>
  <label for="Expiration">Exp. (MM/YYYY)</label>
    <input data-bluesnap="expirationMonth" type="text" name="exp-month" id="exp-month" size="2">
    <span> / </span>
    <input data-bluesnap="expirationYear" type="text" name="exp-year" id="exp-year" size="4">
</div>
```

Add a `data-bluesnap` attribute to your submit button element and set its value to **submitButton** (shown below). This allows BlueSnap to deactivate your button during the 3DS setup process, which you begin in Step 3 by calling `bluesnap.init3DS`. The shopper is prevented from submitting the payment form during the setup process, eliminating the potential for any errors that could result.

HTML ()

```
<button data-bluesnap="submitButton" type="submit" id="submit-button">Pay Now</button>
```

Step 2: Obtain 3-D Secure token for the session

Obtain the 3-D Secure token for the session by sending a Create 3D Secure Token (/web/20191207095936/https://developers.bluesnap.com/v8976-Tools/docs/create-3d-secure-token) request, which is a server-to-server POST request sent to:

`/services/2/threeDSecure`

The response provides the token as the value of the `threeDSecureToken` property (shown below). Grab the token and send it to the client for the next step.

Create 3D Secure Token Response ()

```
{
  "threeDSecureToken": "eyJhbGciOiJIU..."
}
```

Step 3: Call bluesnap.init3DS to set up 3-D Secure

After the checkout page has been loaded, begin the 3DS setup process by calling `bluesnap.init3DS` with the following parameters:

1. The 3-D Secure token (obtained in the prior step)
2. A callback function to handle an error if it occurs

When `bluesnap.init3DS` is called, BlueSnap disables the submit button (which is possible because you added the `data-bluesnap` attribute in Step 1) and set up 3-D Secure. When the process is complete, BlueSnap re-enables the button and calls the provided function.

The below code calls `bluesnap.init3DS` with its required parameters. Note: The `result` is always empty.

JavaScript ()

```
bluesnap.init3DS(threeDSecureToken, function(result, error) {
  if (error != null) {
    // Handle the error. See below.
  } else {
    // 3D Secure setup was a success. No further action is required.
    return;
  }
});
```

Handling setup errors

If an error occurs during the setup process, `error` has the following format. Handle the error by calling `bluesnap.init3DS` with a valid token.

Error example ()

```
{
  "errorCode": "14102",
  "errorDescription": "3D Secure object is missing required fields { threeDSecureToken }"
}
```

> **Disabling 3-D Secure for a specific transaction**
>
> To disable 3-D Secure for a specific transaction, remove the `bluesnap.init3DS` call.

Step 4: Create a 3-D Secure object with transaction data

Create a 3-D Secure object with at least the transaction amount and currency (shown below). In the next step, you call `bluesnap.encrypt` with your object (in addition to other parameters), which allows BlueSnap to send this information to the card issuer to consider as part of their decision whether shopper authentication is required.

Provide as much transaction data as possible

Provide as much transaction data as possible because it can improve the issuer's ability to authenticate the transaction (and assume fraud chargeback liability) without requiring the shopper to enter their credentials in the popup screen, making the 3-D Secure flow entirely transparent and reducing checkout friction (refer to Supported 3-D Secure object properties).

JavaScript ()

```
var threeDSecureObj = {
  amount: 3575,
  currency: 'USD'
  ...
};
```

Note

The amount in the example above actually processes "35.75" (with the decimal).

Step 5: Call bluesnap.encrypt with 3-D Secure object and callback function

When the shopper hits the submit button, call `bluesnap.encrypt` like you would normally, but add two additional parameters:

1. The 3-D Secure object (created in the prior step)
2. A callback function to handle the authentication result (or error)

When the shopper submits their payment and `bluesnap.encrypt` is called, BlueSnap sends the 3D Secure object to the issuer and they decide whether shopper authentication is required. After the authentication process is complete, BlueSnap encrypts the payment form and calls the provided function with the authentication result (or error).

The below code calls `bluesnap.encrypt` with its required parameters.

JavaScript ()

```
bluesnap.encrypt("formID", threeDSecureObj, function(result, error) {
  if (error != null) {
    // Handle the error. See below.
  } else {
    // Handle the authentication result. See below.
  }
});
```

Handling authentication errors

If an error occurs during the authentication process, `error` has the format of the following example and it contains a set of values (shown here), which explains how to handle the error accordingly.

Error example ()

```
{
  "errorCode": "14101",
  "errorDescription": "3D Secure authentication was failed because the shopper did not enter correct credentials"
}
```

Handling authentication results

If no errors occur during the authentication process, `result` has the format of the following example and `authResult` contains one of the values (shown here), which explains how to process the transaction accordingly.

Result example ()

```
{
  "threeDSecure":{
    "authResult":"AUTHENTICATION_SUCCEEDED",
    "threeDSecureResultToken":"eyJhbGciOiJIU..."
  },
  "encryptedCardNumber":"$bsjs_1_0_3$odPj3U+Ap98...",
  "encryptedSecurityCode":"$bsjs_1_0_3$a4jfpzZVitf...",
  "ccLast4Digits":"0002",
  "expirationYear":"2020",
  "expirationMonth":"01"
}
```

Step 6: Process the transaction with encrypted data

To process 3-D Secure payments from your server, include the `threeDSecure` object in the request and set its `threeDSecureResultToken` property to the 3-D Secure result token (obtained in the last step).

Sample Auth Capture request:

JSON ()

```
{
  "amount": 11,
  "softDescriptor": "DescTest",
  "cardHolderInfo": {
    "firstName": "test first name",
    "lastName": "test last name",
    "zip": "123456"
  },
  "currency": "USD",
  "creditCard": {
    "expirationYear": 2023,
    "encryptedCardNumber": "$bsjs_1_0_3$odPj3U+Ap98...",
    "encryptedSecurityCode": "$bsjs_1_0_3$a4jfpzZVitf...",
    "expirationMonth": "07"
  },
  "cardTransactionType": "AUTH_CAPTURE",
  "threeDSecure": {
    "threeDSecureResultToken": "eyJhbGciOiJIUzI1NiJ9.eyJqdG..."
  }
}
```

Back to Top

3-D Secure with Plain Text cards

If you're using plain text cards, implementing 3-D Secure consists of these steps:

- Step 1: Obtain the 3DS Payments token for the session
- Step 2: Add the BlueSnap JavaScript file to your checkout form
- Step 3: Add a script to initiate the 3DS Payments with your 3DS Payments token
- Step 4: Activate 3DS Payments Submit Data function
- Step 5: Process a transaction with the 3DS Reference ID

Step 1: Obtain the 3DS Payments token for the session

Get your 3DS Payments token by sending a server-to-server POST request.

- For a card that is new on your website:

  `BLUESNAPDOMAINPATH/services/2/payment-fields-tokens`

- For a saved card:

  `BLUESNAPDOMAINPATH/services/2/payment-fields-tokens/prefill`

  `{ "ccNumber":"4111 1111 1111 1111", "expDate" : "01/2022" }`

> **Insert the domain for either Sandbox or Production**
>
> In all steps, replace the BLUESNAPDOMAINPATH with the relevant domain for either the BlueSnap Sandbox or Production environment, as follows:
>
> - Sandbox: `https://sandbox.bluesnap.com`
> - Production: `https://ws.bluesnap.com`
>
> For example, the Hosted Fields token request (step 1) should be sent to `https://sandbox.bluesnap.com/services/2/payment-fields-tokens` on Sandbox and `https://ws.bluesnap.com/services/2/payment-fields-tokens` on Production.

The response provides the token in the location header. For example:

`BLUESNAPDOMAINPATH/services/2/payment-fields-tokens/PAYMENTFIELDTOKENID`

Note: Secured Payments Token expires after 60 minutes.

Step 2: Add the BlueSnap JavaScript file to your checkout form

In your checkout page, call the BlueSnap JavaScript file by adding the following script:

JavaScript ()

```
<script type="text/javascript" src="BLUESNAPDOMAINPATH/web-sdk/REQUIRED_VERSION/bluesnap.js">
</script>
// Where REQUIRED_VERSION should be
// 4.1.1 for a specific version
// or 4 for the exact major version
```

Step 3: Add a script to initiate the 3DS Payments with your 3DS Payments token

To call the 3DS Payments script, add the following script to your checkout page

JavaScript ()

```
<script>
    //Run the following command after Document Object Model (DOM) is fully loaded
    // and bluesnap script was loaded
    bluesnap.threeDsPaymentsSetup("PAYMENTFIELDTOKENID", function(sdkResponse){
        var code = sdkResponse.code;
        if (code == 1){  // on Success
                var cardData = sdkResponse.cardData;
                var threeDSecure = sdkResponse.threeDSecure;
                    console.log('card type: ' + cardData.ccType
                    + ', last4digits: '
                    + cardData.last4Digits
                    + ', issuing Country: '
                    + cardData.issuingCountry
                    + ', is Regulated Card: '
                    + cardData.isRegulatedCard
                    + ', card Sub Type: '
                    + cardData.cardSubType
                    + ', bin Category: '
                    + cardData.binCategory
                    + ', cc bin: '
                    + cardData.ccBin
                    + ', 3D Secure result: '
                    + threeDSecure.authResult
                    + ', 3D Secure ReferenceId: '
                    + threeDSecure.threeDSecureReferenceId
                    + ', after that I can call final submit');
                    // SUBMIT THE FORM HERE!
        } else {
                // in case some errors occurred
                var errorsArray = sdkResponse.info.errors;
                // in case some warnings occurred
                var warningsArray = sdkResponse.info.warnings;
        }
    });
</script>
```

Replace `PAYMENTFIELDTOKENID` with the token you obtained in Step 1.

The `SdkResponse` object contains:

JSON ()

```
{
  "status": STATUS,
  "code": ERROR_CODE,
  "info": {   // if an error or warning occurred
     "errors": ["Some errors"],
     "warnings": ['Some warning']
   },
  "cardData": {
    "binCategory": "CONSUMER",
    "ccBin": "411111",
    "cardSubType": "CREDIT",
    "ccType": "VISA",
    "last4Digits": "0002",
    "isRegulatedCard": "Y",
    "issuingCountry": "us"
  },
  "threeDSecure": {
    "authResult": “AUTHENTICATION_SUCCEEDED”,
    "threeDSecureReferenceId": "12345"
  }
}
```

The `"status"` can have the following values:

- `Success`
- `Invalid Data` — Identifies issues with the jsonData provided from the merchant.
- `Inner Error` — Identifies issues in the solution itself.
- `Server Error` — Identifies issues with the BlueSnap server.

The `"code"` field can have the following error codes:

- `1` — On a status of `Success`
- `10` — On a status of `Invalid Data` when an error prevents the process from continuing. For example: Invalid provided currency.
- `15` — On a status of `Invalid Data` when an error does not prevent the process from continuing. (The callback is called with a warning array. The process continues and callback is called again on completion or another error). For example: Invalid billingFirstName.
- Other codes are the BlueSnap server HTTP errors or generic server errors (400, 500). For example:
  - `22013` is `CC_TYPE_IS_NOT_SUPPORTED_BY_THE_MERCHANT`
  - `14040` is `TOKEN_IS_EXPIRED`
  - `14042` is `TOKEN_IS_NOT_ASSOCIATED_WITH_A_PAYMENT_METHOD`
  - `14104` is `SHOPPER_CC_NOT_FOUND`
  - `14102` is `THREE_D_SECURE_MISSING_REQUIRED_FIELDS`

The `"info"` field is present if an error or warning occurred and will contain the `errors` and `warnings` .

- The following are possible values for `errors` :
  - `Invalid amount: <given_amount>` — An invalid amount was given

- `Amount is mandatory and must be of type number` — An amount wasn't provided or provided not as a number
- `Currency <given_currency> is not supported` — The given currency is not supported
- `Currency is mandatory` — An amount wasn't provided
- `Invalid ccNumber`
- `Invalid cvv`
- `Invalid expDate`
- `Invalid ccType`
- `Invalid last4Digits`

- The following is a possible value for `warnings` :
  - `Parameter <given_key> with value of <given_value> is invalid` — Keys verified through this process are: email, all shipping details, and all of the billing details.
    For example: `Parameter shippingFirstName with value of 123 is invalid`

Step 4: Activate 3DS Payments Submit Data function

To activate 3DS Payments Submit Data function when the user presses the submit button and then wait for response before activating the submit form, add the script below into your checkout form:

for a new card () for a saved card ()

```
<script>
        var newCard = {
                "ccNumber":"CREDIT CARD NUMBER",
                "cvv":"CVV", //Optional
                "expDate":"MM/YYYY",
                "amount": 10,
                "currency": "USD"
        }
        bluesnap.threeDsPaymentsSubmitData(newCard);
</script>
```

This function should be activated when the shopper clicks the **Submit** button. This submits the credit card data to BlueSnap, where it is associated with the 3DS Payments Token.

Step 5: Process a transaction with the 3DS Reference ID

Process 3-D Secure transactions (or save shopper payment details) from your server the same way you handle regular card payments—by including the `threeDSecureReferenceId` within the `threeDSecure` property in the request. Refer to the Auth-Capture section (https://web.archive.org/web/20191207095936/https://developers.bluesnap.com/v8976-JSON/docs/auth-capture) for Payment API code samples with 3-D Secure.

JSON Request ()

```
curl -v -X POST https://sandbox.bluesnap.com/services/2/transactions \
-H 'Content-Type: application/json' \
-H 'Accept: application/json' \
-H 'Authorization: Basic dXNlcm5hbWU6cGFzc3dvcmQ=' \
-d '
{
  "cardTransactionType": "AUTH_CAPTURE",
  "recurringTransaction": "ECOMMERCE",
  "softDescriptor": "DescTest",
  "amount": 10,
  "currency": "USD",
  "cardHolderInfo": {
    "firstName": "Will",
    "lastName": "Smith"
  },
  "creditCard": {
    "cardNumber": "4012000033330026",
    "securityCode": "123",
    "expirationMonth": "07",
    "expirationYear":"2019"
  },
  "threeDSecure": {
       "threeDSecureReferenceId":"4759"
  }
}'
```

Back to Top

3-D Secure with Returning Shoppers

If you're using 3-D Secure for returning shoppers, implementation consists of these steps:

Step 1: Obtain the 3DS Payments token for this session
Step 2: Add the BlueSnap JavaScript file to your checkout page
Step 3: Add the script to initiate the 3DS Payments with your 3DS Payments token

Step 4: Activate 3DS Payments Submit Data function
Step 5: Process the transaction with the 3DS Reference ID

Step 1: Obtain the 3DS Payments token for this session

Get your 3DS Payments token by sending a server-to-server POST request to:

```
BLUESNAPDOMAINPATH/services/2/payment-fields-tokens?shopperId=SHOPPERID
```

If you have saved the shopper using your `sellerShopperId`, you can use that instead of `shopperId`.

The response provides the token in the location header. For example:

```
BLUESNAPDOMAINPATH/services/2/payment-fields-tokens/PAYMENTFIELDTOKENID
```

> **Insert the domain for either Sandbox or Production**
>
> In all steps, replace the BLUESNAPDOMAINPATH with the relevant domain for either the BlueSnap Sandbox or Production environment, as follows:
>
> - Sandbox: `https://sandbox.bluesnap.com`
> - Production: `https://ws.bluesnap.com`
>
> For example, the Hosted Fields token request (step 1) should be sent to `https://sandbox.bluesnap.com/services/2/payment-fields-tokens` on Sandbox and `https://ws.bluesnap.com/services/2/payment-fields-tokens` on Production.

Note: Secured Payments Token expires after 60 minutes.

Step 2: Add the BlueSnap JavaScript file to your checkout page

In your checkout page, call the BlueSnap JavaScript file by adding the following script:

JavaScript ()

```
<script type="text/javascript" src="BLUESNAPDOMAINPATH/web-sdk/REQUIRED_VERSION/bluesnap.js">
</script>
// Where REQUIRED_VERSION should be
// 4.1.1 for a specific version
// or 4 for the exact major version
```

Step 3: Add the script to initiate the 3DS Payments with your 3DS Payments token

To call the 3DS Payments script, add the following script to your checkout page

JavaScript ()

```
<script>
    //Run the following command after Document Object Model (DOM) is fully loaded
    // and bluesnap script was loaded
    bluesnap.threeDsPaymentsSetup("PAYMENTFIELDTOKENID", function(sdkResponse){
        var code = sdkResponse.code;
        if (code == 1){  // on Success
                var cardData = sdkResponse.cardData;
                var threeDSecure = sdkResponse.threeDSecure;
                    console.log('card type: ' + cardData.ccType
                    + ', last4digits: '
                    + cardData.last4Digits
                    + ', issuing Country: '
                    + cardData.issuingCountry
                    + ', is Regulated Card: '
                    + cardData.isRegulatedCard
                    + ', card Sub Type: '
                    + cardData.cardSubType
                    + ', bin Category: '
                    + cardData.binCategory
                    + ', cc bin: '
                    + cardData.ccBin
                    + ', 3D Secure result: '
                    + threeDSecure.authResult
                    + ', 3D Secure ReferenceId: '
                    + threeDSecure.threeDSecureReferenceId
                    + ', after that I can call final submit');
                    // SUBMIT THE FORM HERE!
        } else {
                // in case some errors occurred
                var errorsArray = sdkResponse.info.errors;
                // in case some warnings occurred
                var warningsArray = sdkResponse.info.warnings;
        }
    });
</script>
```

Replace `PAYMENTFIELDTOKENID` with the token you obtained in Step 1.

The `SdkResponse` object contains:

JSON ()

```
{
  "status": STATUS,
  "code": ERROR_CODE,
  "info": {   // if an error or warning occurred
     "errors": ["Some errors"],
     "warnings": ['Some warning']
   },
  "cardData": {
    "binCategory": "CONSUMER",
    "ccBin": "411111",
    "cardSubType": "CREDIT",
    "ccType": "VISA",
    "last4Digits": "0002",
    "isRegulatedCard": "Y",
    "issuingCountry": "us"
  },
  "threeDSecure": {
    "authResult": “AUTHENTICATION_SUCCEEDED”,
    "threeDSecureReferenceId": "12345"
  }
}
```

The `"status"` can have the following values:

- `Success`
- `Invalid Data` — Identifies issues with the jsonData provided from the merchant.
- `Inner Error` — Identifies issues in the solution itself.
- `Server Error` — Identifies issues with the BlueSnap server.

The `"code"` field can have the following error codes:

- `1` — On a status of `Success`
- `10` — On a status of `Invalid Data` when an error prevents the process from continuing. For example: Invalid provided currency.
- `15` — On a status of `Invalid Data` when an error does not prevent the process from continuing. (The callback is called with a warning array. The process continues and callback is called again on completion or another error). For example: Invalid billingFirstName.
- Other codes are the BlueSnap server HTTP errors or generic server errors (400, 500). For example:
  - `22013` is `CC_TYPE_IS_NOT_SUPPORTED_BY_THE_MERCHANT`
  - `14040` is `TOKEN_IS_EXPIRED`
  - `14042` is `TOKEN_IS_NOT_ASSOCIATED_WITH_A_PAYMENT_METHOD`
  - `14104` is `SHOPPER_CC_NOT_FOUND`
  - `14102` is `THREE_D_SECURE_MISSING_REQUIRED_FIELDS`

The `info` field is present if an error or warning occurred and will contain the `errors` and `warnings` .

Step 4: Activate the 3DS Payments Submit Data function

To activate 3DS Payments Submit Data function when the user presses the submit button and then wait for response before activating the submit form, add the script below into your checkout form:

JavaScript ()

```
<script>
        var previouslyUsedCard = {
                "last4Digits": "1111",
                "ccType": "VISA",
                "amount": 10,
                "currency": "USD"
        }
  bluesnap.threeDsPaymentsSubmitData(previouslyUsedCard);
</script>
```

This function should be activated when the shopper clicks the **Submit** button. This submits the credit card data to BlueSnap, where it is associated with the 3DS Payments Token.

Step 5: Process the transaction with the 3DS Reference ID

Process 3-D Secure transactions (or save shopper payment details) from your server the same way you handle regular card payments—by including the `threeDSecureReferenceId` within the `threeDSecure` property in the request. Refer to the Auth-Capture section (https://web.archive.org/web/20191207095936/https://developers.bluesnap.com/v8976-JSON/docs/auth-capture) for Payment API code samples with 3-D Secure.

JSON Request ()

```
curl -v -X POST https://sandbox.bluesnap.com/services/2/transactions \
-H 'Content-Type: application/json' \
-H 'Accept: application/json' \
-H 'Authorization: Basic dXNlcm5hbWU6cGFzc3dvcmQ=' \
-d '
{
        "cardTransactionType": "AUTH_CAPTURE",
        "softDescriptor": "DescTest",
        "amount": 10,
        "currency": "USD",
        "vaultedShopperId":"19574268",
        "creditCard": {
                "cardLastFourDigits": "1111",
                "cardType": "VISA"
                },
        "threeDSecure": {
                "threeDSecureReferenceId":"5303"
        }
}'
```

Back to Top

Processing payments with data from external Merchant Plug-In (MPI)

If you're using an external Merchant Plug-In (MPI) for 3-D Secure and you only want to process the payment through BlueSnap, first contact Merchant Support (https://web.archive.org/web/20191207095936/https://bluesnap.zendesk.com/hc/en-us/requests/new?ticket_form_id=360000127087) and inform them you're using an external MPI so that your account can be configured appropriately.

To process the payment, include the `threeDSecure` object (https://web.archive.org/web/20191207095936/https://developers.bluesnap.com/v8976-JSON/docs/threedsecure).

Sample Auth Capture request:

JSON ()

```
{
  "amount": 11,
  "softDescriptor": "DescTest",
  "cardHolderInfo": {
    "firstName": "test first name",
    "lastName": "test last name",
    "zip": "123456"
  },
  "currency": "USD",
  "creditCard": {
    "expirationYear": 2019,
    "securityCode": 111,
    "expirationMonth": "07",
    "cardNumber": 4012000033330026
  },
  "cardTransactionType": "AUTH_CAPTURE",
  "threeDSecure": {
    "eci": "05",
    "cavv": "AAABAWFlmQAAAABjRWWZEEFgFz+A",
    "xid": "MGpHWm5ZWVpKclo0aUk0VmltVDA=",
     "dsTransactionId" : "e08da266-b58d-45c9-a1f8-570b7fb80e30",
     "threeDSecureVersion" : "2.1.0"
  }
}
```

Back to Top

Supported 3-D Secure object properties

`amount` *number* **required**
`currency` *string* **required**
`billingFirstName` *string* optional
`billingLastName` *string* optional
`billingCountry` *string* optional
`billingState` *string* optional
`billingCity` *string* optional
`billingAddress` *string* optional
`billingZip` *string* optional
`shippingFirstName` *string* optional
`shippingLastName` *string* optional
`shippingCountry` *string* optional

`shippingState` *string* optional
`shippingCity` *string* optional
`shippingAddress` *string* optional
`shippingZip` *string* optional
`email` *string* optional
`phone` *string* optional

Back to Top

Error codes

Error Code	Description
`14100`	**Problem**: 3-D Secure is not enabled **Resolution**: Enable 3-D Secure
`14101`	**Problem**: 3-D Secure authentication failed **Resolution**: Inform the shopper to try a different payment method
`14102`	**Problem**: 3-D Secure object is missing required fields { ... } **Resolution**: Make sure 3-D Secure object has both transaction amount and currency
`14103`	**Problem**: 3-D Secure client error **Resolution**: Continue without 3DS

Back to Top

Authentication results

`AUTHENTICATION_BYPASSED`

- **Meaning**: 3-D Secure authentication was bypassed due to the merchant's configuration.
- **To Proceed**: Process the transaction without 3-D Secure
  - in Hosted Payment Fields
  - in Client-Side Encryption

`AUTHENTICATION_SUCCEEDED`

- **Meaning**: 3-D Secure authentication was successful because the shopper entered their credentials correctly or the issuer authenticated the transaction without requiring shopper identity verification. This transaction is eligible for a 3-D Secure chargeback liability shift (https://web.archive.org/web/20191207095936/https://support.bluesnap.com/docs/3d-secure#section-chargeback-liability-shift).
- **To Proceed**:
  - In Hosted Payment Fields: Process the 3-D Secure transaction
  - In Client-Side Encryption: Send `result.threeDSecure.threeDSecureResultToken` to your server and process the 3-D Secure transaction

`AUTHENTICATION_UNAVAILABLE`

- **Meaning**: 3-D Secure is unavailable for this card or a technical error occurred.
- **To Proceed**: Process the transaction without 3-D Secure
  - in Hosted Payment Fields
  - in Client-Side Encryption

`AUTHENTICATION_FAILED`

- **Meaning**: The card on the transaction could not be authenticated..
- **To Proceed**: If the option **Process failed 3DS transactions** is enabled in **Settings > Fraud Settings** in the BlueSnap Console, then the transaction can be processed.

> **Note**
>
> The authentication result above is included in the Order Locator information for any order processed using 3-D

> Secure.

Back to Top

Testing 3-D Secure

For information about Sandbox testing of 3-D Secure, refer here (https://web.archive.org/web/20191207095936/https://support.bluesnap.com/docs/3d-secure#section-sandbox-testing-of-3-d-secure).