

CIT payments with 3D Secure - Direct integration

3D Secure (3DS) is an authentication method that provides additional layer of security against fraud for online card payments. If you are accepting payments within European Economic Area then it is mandatory to authenticate the transactions with 3D secure to comply with PSD2 SCA (Strong customer authentication) requirements.

3D Secure integration options

enables 3DS authentication for all integration types:

Hosted payment page or Hosted forms

If you are using 's Hosted Payment Page or Hosted Forms, you don't need to take any additional steps to implement 3DS. will automatically handle the entire authentication process for you, ensuring compliance with PSD2 SCA requirements.

Direct integration

For merchants using Direct Integration, the implementation of 3DS depends on how you choose to handle the authentication process:

Merchant-managed 3DS authentication

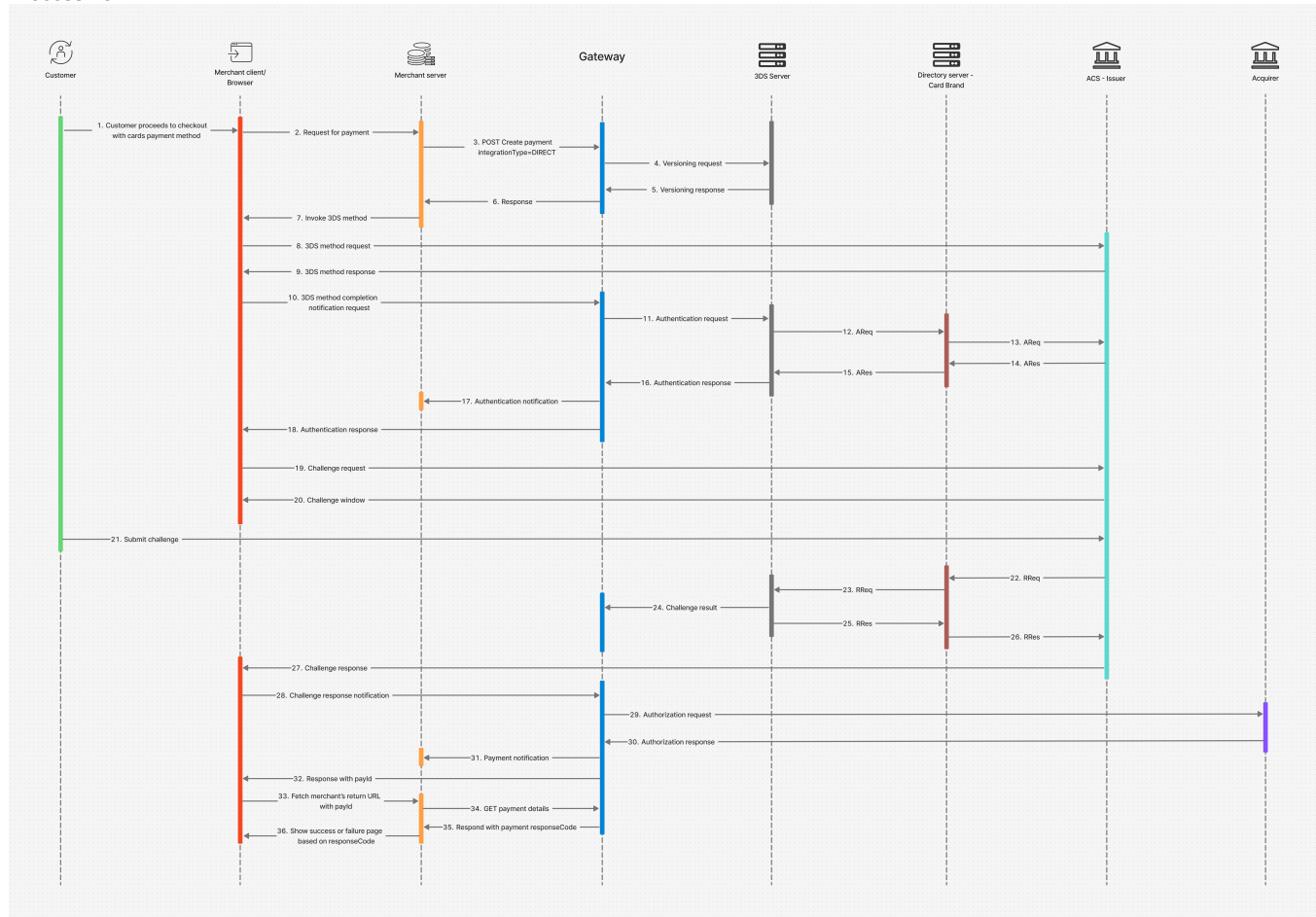
If you manage the 3DS authentication process independently before sending the payment request to , ensure that all relevant authentication data is included in the `paymentMethods.card.threedsdata` object within the [payment request](#).

```
{
  "acsProtocolVersion": "2.2.0",
  "authenticationValue": "AAABBIcWERFgUFgUQk1FQRE=",
  "eci": "02",
  "threeDSServerTransID": "55570cf-bt5b-43fe-bd0d-2trr3427401c",
  "acsXID": "34565040-e95d-4f55-9aa1-d48d1234acd4",
  "dsTransID": "4347f607-d631-4ad5-34564533cadce558",
  "intermediateStatus": "Y",
  "finalStatus": "Y",
  "challengeRequestInd": "01"
}
```

managed 3DS authentication

This section describes the process flow for merchants who want to handle the 3DS authentication process.

Process flow



1. Customer proceeds to checkout with cards as a preferred payment method on your web shop and submits all the card data.

2. Your frontend makes a payment request to your backend.

3. Your backend makes a [Create payment](#) call to .

Versioning:

4. makes a versioning request to 3DS server to fetch Access Control Server (ACS) and Directory Server (DS) protocol versions that corresponds to the card account range and optionally a 3D secure method URL.

5. receives the versioning response with the ACS and DS protocol versions.

6. responds with a HTTP 201 response code to your backend with versioning data in `paymentMethods.card.versioningData`

```

{
  "threeDSServerTransID": "f049e8fb-5d93-1234-b995-3eaf05cef06a",
  "acsStartProtocolVersion": "2.2.0",
  "acsEndProtocolVersion": "2.2.0",
  "dsStartProtocolVersion": "2.2.0",
  "dsEndProtocolVersion": "2.2.0",
  "threeDSMethodURL": "https://testacsserver.com/3ds-method",
  "threeDSMethodDataForm":
"eyJ0aHJlZURTZW0aG9kTm90aWZpY2F0aW9uVGVJMIjoiaHR0cHM6Ly93d3cuY29tcHV0b3AtcGF5Z2F0ZS5jb20vY2JUaHJlZURTLMFzcHg_YWNoaW9uPW10aGR0dGZuIiwidGhyZWVlNlcnZlclRyYw5zSUQiOiJmMDQ5ZThmYi01ZDkzLTQ2MTAtYjk5NS0zZWVmMDVjZWYwNmEifQ",
  "threeDSMethodData": {
    "threeDSMethodNotificationURL": "https://www.comutop-paygate.com/cbThreeDS.aspx?action=mthdNtfn",
    "threeDSServerTransID": "f049e8fb-5d93-4610-b995-3eaf05cef06a"
  }
}
  
```

Refer `paymentMethods.card.versioningData.errorDetails` object for the reasons if any parameter in `versioningData` is not populated.

3DS Method: 3DS Method is an optional step in the 3DS authentication flow that allows the card issuer's ACS to collect additional browser/device information before the actual authentication request (AReq) is sent. This data helps the ACS perform a more accurate risk assessment for the transaction.

7. Based on the versioning response received, 3DS method process should be invoked. To do this, your backend should provide `threeDSMethodData` and `threeDSMethodURL` to your frontend as received from versioning response. Your frontend should create a hidden iframe containing a form with a hidden input field for `threeDSMethodData` and submit it to `threeDSMethodURL`. The `threeDSMethodData` should be sent in Base64 encoded format which you can fetch directly from `threeDSMethodDataForm` in the versioning response. Below is an example snippet of the implementation:

```
<form name="threeDSMethodForm" method="POST" action="https://testacsserver.com/3ds-method">
  <input type="hidden" name="threeDSMethodData" value="
eyJ0aHJlZURTU2VydMvYVhJhbnNJRCI6IjNhYzdjYWE3LWFhNDItMjY2My030TFiLTJhYzA1YTU0MmM0YSIsInRocmVlRFNNZXRob2R0b3RpZ
mljYXRpb25VUkw0Ij0aHJlZURTtwV0aG9kTm90awZpY2F0aw9uVGVJMi0">
</form>
```

If the issuer doesn't support 3DS Method process, `threeDSMethodURL` will be populated by so that the 3DS Method process of posting the hidden form can happen regardless. This is necessary to facilitate direct communication between the browser and during the 3DS process flow.



You may use the operations `init3DSMethod` or `createIframeAndInit3DSMethod` at your discretion from the [netcetra3DSWebSDK](#) in order to initiate the 3-D Secure Method.

8. Your frontend submits the hidden form to ACS 3DS Method URL that enables ACS to collect all the required information.

9. ACS responds to the 3DS Method request with an HTML document.

10. The HTML document contains javascript that automatically creates and submits a hidden form with the `threeDSMethodData` to `threeDSMethodNotificationURL` which is hosted by .

Authentication:

11. Upon receiving 3DS completion notification request, proceeds with authentication request to 3DS server.

12. 3DS server submits authentication request to directory server.

13. Directory server submits authentication request to ACS

14. ACS sends back an authentication response to directory server

15. Directory server forwards the response to 3DS server

16. 3DS server forwards the response to . If the authentication response doesn't enforce a challenge then the process skips to step 29, else the subsequent challenge process begins.

17. sends a notification with `payId` in the body to `ur1s.webhook` that you submitted in the initial request (Step 3). You can call [Retrieve payment details](#) with `payId` to get all authentication response data.

18. additionally sends the authentication response (base64 encoded) to the browser as a response to the 3DS method completion notification request (Step 10). Below is a snippet of decoded authentication response sent to the browser.

```
{
  "threeDSServerTransID": "f049e8fb-5d93-1234-b995-3eaf05cef06a",
  "acsChallengeMandated": true,
  "acsDecConInd": false,
  "acsOperatorID": "10024942",
  "acsReferenceNumber": "3DS_LOA_ACS_HIIN_020200_00553",
  "acsTransID": "70283ff7-df85-44be-b157-3bc6f8a31b89",
  "acsURL": "https://testacsserver.com/3ds-method",
  "dsReferenceNumber": "VISA.V 17 0003",
  "dsTransID": "2f15b09e-b915-4d5e-9f04-239ed985f3bb",
  "messageType": "Ares",
  "messageVersion": "2.2.0",
  "transStatus": "C",
  "challengeRequest": {
    "threeDSServerTransID": "f049e8fb-5d93-1234-b995-3eaf05cef06a",
    "acsTransID": "70283ff7-df85-44be-b157-3bc6f8a31b89",
    "challengeWindowSize": "04",
    "messageVersion": "2.2.0",
    "messageType": "CReq"
  },
  "base64EncodedChallengeRequest":
  "eyJtZXNzYWdlVHlwZSI6IkNSZXElLCJ0aHJlZURTU2VydmVyVHJhbnNJRCI6ImYwNDl1OGZiLTk0MTtNDYxMC1iOTk1LTNlYWYwNWw1ZjA2YSIsImFjc1RyYW5zSUQiOiI3MDI4M2ZmNy1kZjg1LTQ0YmUtYjE1Ny0zYmM2ZjhhMzFiODkiLCJjaGFsbGVuZ2VXaw5kb3dTaXplIjoIMDQiLCJtZXNzYWdlVmVyc2lubiI6IjIuMi4wIn0"
}
```

Challenge:

19. Based on the authentication response received, your frontend should create a challenge request. To do this, Base64 decode the authentication response and collect the value present in base64EncodedChallengeRequest and acsURL. Submit base64EncodedChallengeRequest to acs URL via a hidden HTML iframe containing a form with hidden input field. Below is an example snippet of the implementation:

```
<form name="challengeRequestForm" method="POST" action="https://testacsserver.com/3ds-method">
  <input type="hidden" name="creq" value="
  eyJtZXNzYWdlVHlwZSI6IkNSZXElLCJ0aHJlZURTU2VydmVyVHJhbnNJRCI6ImYwNDl1OGZiLTk0MTtNDYxMC1iOTk1LTNlYWYwNWw1ZjA2YSIsImFjc1RyYW5zSUQiOiI3MDI4M2ZmNy1kZjg1LTQ0YmUtYjE1Ny0zYmM2ZjhhMzFiODkiLCJjaGFsbGVuZ2VXaw5kb3dTaXplIjoIMDQiLCJtZXNzYWdlVmVyc2lubiI6IjIuMi4wIn0">
</form>
```



You may use the operations `init3DSChallengeRequest` or `createIFrameAndInit3DSChallengeRequest` at your discretion from the [netcetra3DSWebSDK](#) in order to submit challenge request.

20. ACS renders the challenge window on the browser.
21. Customer completes the challenge.
22. ACS notifies the outcome of the challenge to directory server via result request (RReq) message.
23. Directory server forwards RReq to 3DS server.
24. 3DS server forwards challenge result to .
25. 3DS server acknowledges to RReq with a result response (RRes) to directory server.
26. Directory server forwards RRes to ACS.
27. ACS additionally provides the challenge response to the browser as a response to "Submit challenge" (Step 21). Your frontend should listen to the challenge response returned to the iframe.
28. Once challenge response is received your frontend submits challenge response notification request to . Below is the sample decoded JSON:

```
{
  "messageType": "CRes",
  "messageVersion": "2.2.0",
  "threeDSSTransID": "f049e8fb-5d93-1234-b995-3eaf05cef06a",
  "acsTransID": "70283ff7-df85-44be-b157-3bc6f8a31b89",
  "challengeCompletionInd": "Y",
  "transStatus": "Y"
}
```

Authorization:

29. proceeds to authorization with the acquirer.

30. Acquirer provides the authorization response to .

31. sends a notification with payId in the message body to `urls.webhook` that you submitted initially (step 3). You can call [Retrieve payment details](#) with payId to get all authorization response data.

32. additionally responds to the browser with payId in the message body as a response to step 28.

33. Your frontend fetches your shop's return URL with payId from your backend.

34. Your backend makes [Retrieve payment details](#) with payId to to fetch the final responseCode of the transaction.

35. responds with the responseCode of the transaction.

36. Your backend renders success or failure page depending on the responseCode of the transaction.