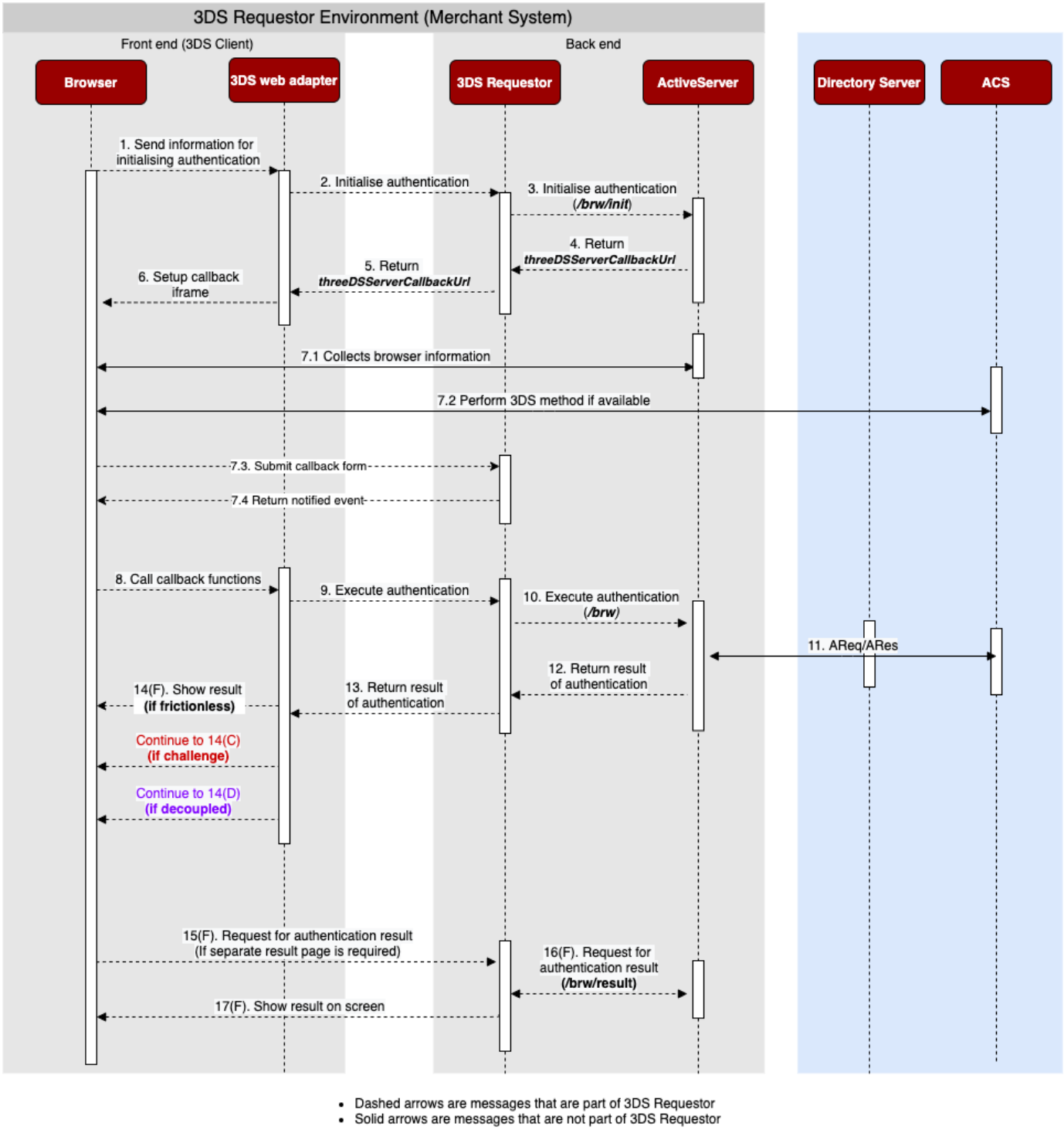
Authentication sequence

The sequence diagram below breaks down the process of a 3DS2 authentication, step by step, explicitly focusing on how the **3DS Requestor** functions inside the 3DS2 flow, using **GPayments'** APIs.

If any of the steps are part of the **3DS Requestor environment** process, they will be marked as ←3DS Requestor process for your reference, as these steps are provided as demo code and may require customisation to fit your requirements.

> **Note**
>
> Implementation of the **3DS web adapter** at the front end, and the **3DS Requestor** at the back end, is required to integrate ActiveServer.

- **Process 1: Initialise Authentication**
  - Step 1 to Step 7
- **Process 2: Execute Authentication**
  - Frictionless flow - Step 8 to Step 13, and Step 14(F)
  - Challenge flow - Step 8 to Step 13, and Step 14(C) to Step 21(C)
  - Decoupled flow - Step 8 to Step 13, and Step 14(D) Step 17.1(D)
- **Process 3: Get Authentication Result**
  - Frictionless flow - Step 15(F) to Step 17(F)
  - Challenge flow - Step 22(C) to Step 28(C)
  - Decoupled flow - Step 15(D) to Step 22(D)



1. **Send information for initialising authentication** ←3DS Requestor process
    - Information for initialising authentication obtained from the checkout page, such as the card number, is sent to the **3DS web adapter**. This is a simple JavaScript simulating how the front end system of 3DS Requestor works.

2. **Initialise authentication** ←3DS Requestor process
    - **3DS web adapter** makes a POST request to the **3DS Requestor** with information collected from the checkout page and requests the **3DS Requestor** to initialise authentication.
3. **Initialise authentication** ←3DS Requestor process
    - The **3DS Requestor** obtains the information from the front end and makes a POST API call to the `initAuth` end point to initialise authentication.
    - An important field sent here is the `eventCallbackUrl`, which will be required to start Step 8 to allow **ActiveServer** to callback to this URL to notify the end of the browser information collection.
4. **Return** `threeDSServerCallbackUrl`
    - A successful response from `initAuth` end point contains `threeDSServerCallbackUrl` and `threeDSServerTransID`.

> **RBC**
>
> The `threeDSServerCallbackUrl` and `monUrl` can potentially be `null` in the response when `skipAutoBrowserInfoCollect` is `true`, and 3DS Method is not available or the `acctNumber` was not provided. In this scenario `threeDSMethodAvailable` will also be `false`.

5. **Return** `threeDSServerCallbackUrl` ←3DS Requestor process
    - The **3DS Requestor** returns the `threeDSServerCallbackUrl` back to the **3DS web adapter**.
6. **Setup callback** `iframe` ←3DS Requestor process
    - Insert a hidden `iframe`, with `src` set to `threeDSServerCallbackUrl`. This will allow **ActiveServer** to connect to the **3DS Requestor**. **ActiveServer** will callback to this `iframe`.

> **RBC**
>
> If the `threeDSServerCallbackUrl` is `null`, you may skip inserting the `iframe`. The requestor should proceed to Step 10.

7. **Info collecting page flow (browser and/or 3DS Method)**

7.1 **Collects browser information**

- **ActiveServer** collects the browser information via the `iframe`, this a required field to be sent in AReq process.

> **RBC**
>
> When `skipAutoBrowserInfoCollect` is `true`, ActiveServer will skip collecting the browser info automatically.

7.2 **Perform 3DS method**

- ActiveServer facilitates the optional 3DS method data collecting for the **ACS** if the 3DS method is available. The **ACS** then collects the 3DS method data via the prepared `iframe`.

7.3 **Submits callback form**

- As a result of Step. 7.1 or Step 7.2, the callback form is returned containing a hidden HTML form. This form is submitted immediately as soon as it renders which notifies the requestor that browser information collecting or 3DS method has finished.
- If 3DS method was not performed due to being unavailble, the `event` will be set to `3DSMethodSkipped`. Otherwise, `3DSMethodFinished` will be returned. Along with `param` attribute set to the base64 encoded browser information collected by ActiveServer.

> **3DS method monitoring**
>
> ActiveServer also provides a fail safe mechanism if the 3DS method failed to be performed by the ACS. The 3DS requestor can use the `monUrl` to setup an monitoring iframe which will notify an event `InitAuthTimedOut` if 3DS method is availble but did not complete within 10 seconds. It is recommended to continue with authentication when the 3DS requestor receive `InitAuthTimedOut` event, and `param` attribute will be set to the base64 encoded browser information.

7.4 **Return notified event**

- The 3DS Requestor delivers the events to the front-end. When the event is notified and it is either `3DSMethodFinished`, `3DSMethodSkipped` or `InitAuthTimedOut`, the 3DS requestor can now continue with `Execute authentication` process as the ActiveServer is now ready to perform authentication.
- Alternatively, **ActiveServer** will send `3DSMethodHasError` event to the 3DS Requestor when the 3DS Method monUrl is implemented, and there was a unexpected 3DS Method

notification by the ACS after the 3DS Method timeout period. This event is only for troubleshooting and logging purposes, and the 3DS Requestor should not deliver it to the front-end, you can check the demo code for more details.

8. **Call callback function** ←3DS Requestor process
   - During authentication initialisation, the **3DS Requestor** sends the `eventCallBackUrl` so that the ACS can notify the **3DS Requestor** when the 3DS method is finished or skipped. This is done through the `iframe` setup in Step. 7.
   - Once the **3DS Requestor** receives a notification, it will pass required parameters including `callbackFn` and render `notify-3ds-events.html` to the `iframe`. When `notify_3ds_events.html` is rendered it will simply call the `callbackFn` defined in the `3ds-web-adapter`.
9. **Execute authentication** ←3DS Requestor process
   - `callbackFn` can be either `_on3DSMethodSkipped()`, `_onInitAuthTimedOut()` or `_on3DSMethodFinished()`, all of which will end up calling `doAuth()`. 3DS-web-adapter will call `doAuth()`, which will ask the 3DS Requestor to execute the authentication.
   - `_onInitAuthTimedOut` means that ACS did not perform the 3DS Method process in time.
10. **Execute authentication** ←3DS Requestor process
    - The **3DS Requestor** will make a call to the `auth` end point, which will initiate the authentication processes.

> **RBC**
>
> The requestor must collect the browser information, the put the data into `browserInfoCollected` when making a call to the `auth` end point. Please check the requestor demo for details.

11. **AReq/ARes**
    - An Authentication Request (AReq) is sent from **ActiveServer** via the Directory Server to an ACS. An Authentication Response (ARes) containing the authentication results is sent from the ACS to **ActiveServer**.
12. **Return result of authentication**
    - `auth` end point returns a `tranStatus` to the **3DS Requestor**.

13. **Return result of authentication** ←3DS Requestor process

    - Return the result of authentication back to the web adaptor.

> **Info**
>
> If the `transStatus` is "C" go to `Step 14(C)`, if "D" go to `Step 14(D)`, otherwise `Step 14(F)` is optional as you already have the authentication result.

Frictionless flow specific

14(F). **Show Result (if frictionless)** ←3DS Requestor process

- If the authentication result has a `transStatus` of "Y", `authSuccess()` is called, which redirects the page to `/auth/result?transId`.

15(F). **Request for authentication result (if separate result page is required)** ←3DS Requestor process

- The steps outlined here is **optional** as you have obtained the final authentication result in `Step.13` following steps may be taken if you need the authentication result on a separate screen.
- The browser notifies the **3DS Requestor** with the `transId`, and the transaction result is available for request.

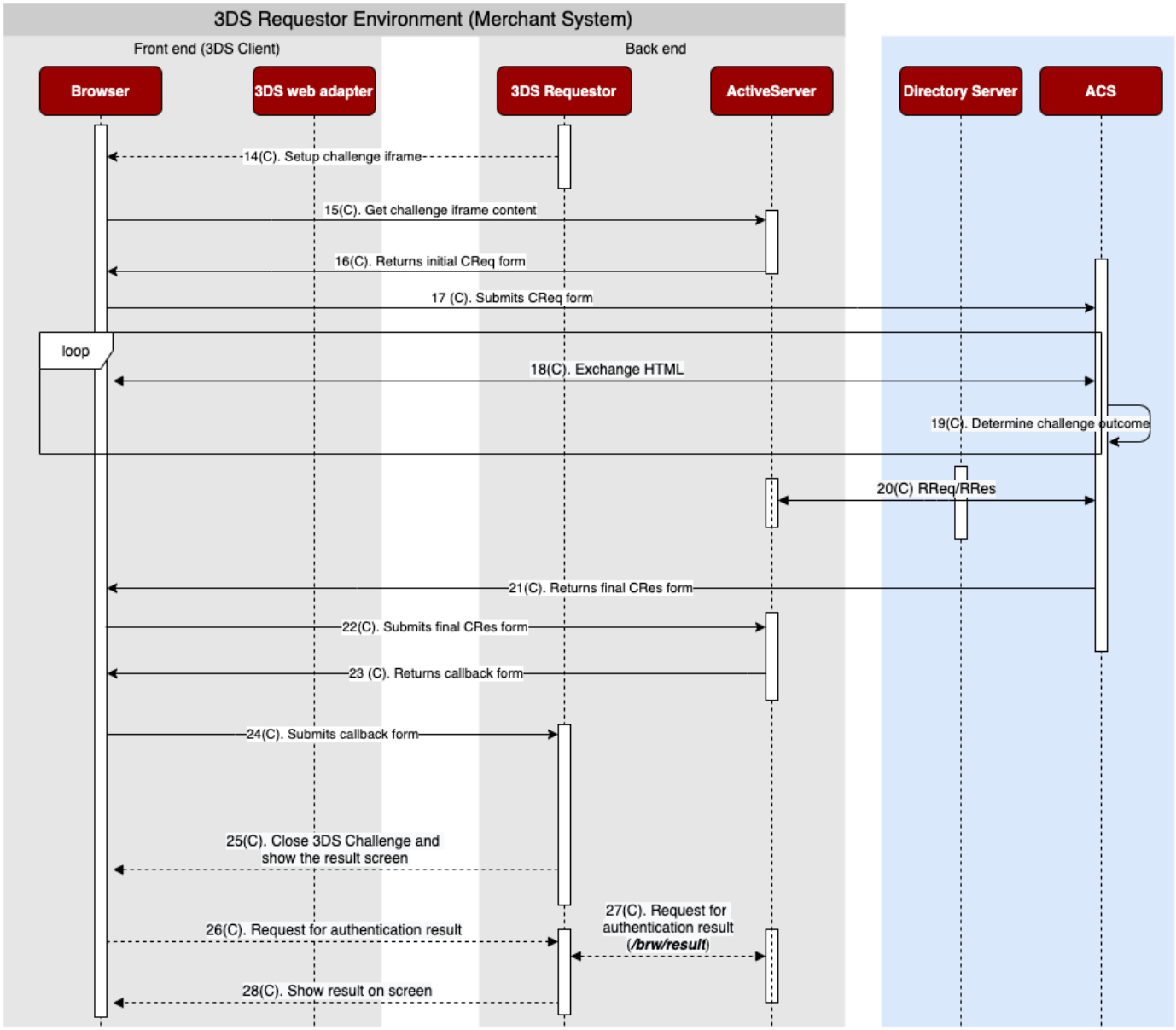
16(F). **Request for authentication result** ←3DS Requestor process

- The **3DS Requestor** will ask for a result receipt from **ActiveServer** by calling the `result` end point.

17(F). **Show result on screen** ←3DS Requestor process

- Use the authentication result obtained to show the result on the UI or authentication value and eci can be used to continue with the authorization process.

Challenge flow specific



14(C). **Setup `iframe` (if challenge)** ←3DS Requestor process

- If the authentication result has a `transStatus` of "C", the iframe with attribute `src` set to the `challengeUrl` must be created.

15(C). **Get challenge iframe content**

- By setting the `src` attribute on the `iframe`, the browser will request for `iframe` content from ActiveServer.

16(C). **Returns initial CReq form**

- The `iframe` content contains an hidden initial CReq form which will be submitted automatically when the form renders inside a browser.

17(C). **Submits CReq form**

- ActiveServer will automatically submit the initial CReq form, and sends the initial CReq to the `acsURL` returned in ARes to initiate the challenge process with the ACS.

18(C). **Exchange HTML**

- In response to the initial CReq, ACS will render the challenging `iframe` content on the browser.
- The cardholder interacts with the challenge process and completes authentication with the ACS.

19(C). **Determine challenge outcome**

- The ACS determines if the challenge performed is successful or not.

20(C). **RReq/RRes**

- The ACS sends a Result Request (RReq) containing the authentication results via the Directory Server to **ActiveServer**. **ActiveServer** will then acknowledge its receipt with a Result Response (RRes).

21(C). **Returns final CRes form**

- The ACS will render a hidden final CRes form on the browser.

22(C). **Submits final CRes form**

- The final CRes form is submitted as soon as it has been rendered on the browser. ActiveServer will validate the final CRes according to the EMVCo specification requirements and returns a callback form to the browser `iframe` .

23(C). **Returns callback form**

- Callback form is returned from ActiveServer. This is an HTML containing a hidden form, which will be submitted automatically as soon as it renders on the browser.

24(C). **Submits callback form**

- The callback form is submitted automatically with an event, `AuthResultReady` to the 3DS requestor backend which notifies the 3DS requestor side that final authentication result is available for request from ActiveServer.

25(C). **Close 3DS Challenge and show the result screen** ←3DS Requestor process

- Since the challenge is finished, the **3DS Requestor** redirects the page to `/auth/result?transId`.

26(C). **Request for authentication result** ←3DS Requestor process

- The browser notifies the **3DS Requestor** with the `transId`, and the transaction result is available for request.

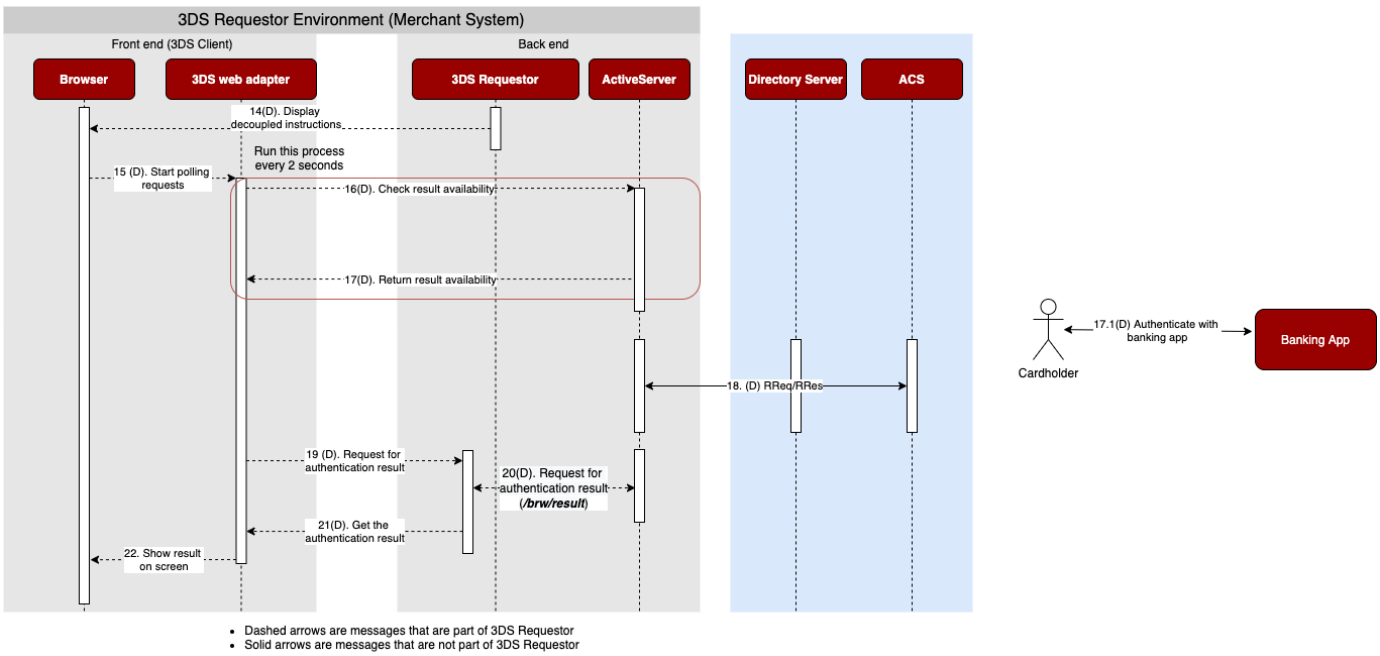
27(C). **Request for authentication result (brw/result)** ←3DS Requestor process

- The **3DS Requestor** asks for a result receipt from the **`result`** end point, same as Step 16(F).

28(C). **Show result on screen** ←3DS Requestor process

- Use the authentication result obtained to show the result on the UI or authentication value and eci can be used to continue with the authorization process.

Decoupled flow specific



14(D). **Display decoupled instructions** ←3DS Requestor process

- If the authentication result has a `transStatus` of "D" and `acsDecConInd` is "Y". Then ACS has agreed to perform decoupled authentication with the cardholder. How the decoupled authentication is performed is outside the scope of 3DS.
- The ACS will return the `cardholderInfo` text containing information for the cardholder to follow. For example, "Please open TestBank app to continue the transaction.", asking the cardholder to a banking app and performing biometric authentication. The 3DS requestor should display this message on the UI so that the cardholder is notified on what to do next.

> **Important**
>
> `transStatus` of "D" and `acsDecConInd` of "Y" will only be returned if `threeDSReqDecInd` is set to **Y** in Step. 9. In other words, the decouple flow will only be executed if requested by 3DS requestor and ACS confirms the use of decoupled authentication.

15(D). **Start polling for result** ←3DS Requestor process

- `resultMonUrl` is also returned in `Step.13`, this is a URL which allows the 3DS requestor to know when to ask ActiveServer for the authentication result. This URL may be called at an interval, from the browser which can be used to poll the status of result availability from the ActiveServer.

16(D). **Check result availability** ←3DS Requestor process

- The `resultMonUrl` is called to check for authentication result availability from ActiveServer.
- This process is performed at an interval, as decoupled authentication is performed outside the 3DS flow.

17(D). **Return result availability** ←3DS Requestor process

- The status of result availability is returned, if authentication result is available, the `event` field will be set to `AuthResultReady`, if not then `AuthResultNotReady`.

17.1(D). **Authentication with banking app**

- This step may be performed outside the 3DS flow and how this authentication is performed is outside the scope of 3DS. The cardholder may authenticate with their registered banking app for example. The authentication process may be performed at any time frame as long as

the 3DS requestor specified `threeDSReqDecMaxTime` is not exceeded, which may be as long as 10080 minutes (168 hours).

18(D). **RReq/RRes**

- The ACS sends a Result Request (RReq) containing the authentication results via the Directory Server to **ActiveServer** after decoupled authentication or `threeDSReqDecMaxTime` has been exceeded. **ActiveServer** will then acknowledge its receipt with a Result Response (RRes).

19(D). **Request for authentication result** ←3DS Requestor process

- The polling result in `Step. 17(D)` returns `AuthResultReady` as now ActiveServer has received the RReq and final authentication result is available for request.

20(D). **Request for authentication result (brw/result)** ←3DS Requestor process

- The **3DS Requestor** asks for a result receipt from the `result` end point, same as Step 16(F) or Step. 27(C).

21(D). **Get authentication result** ←3DS Requestor process

- The authentication result is returned to the **3DS web adapter**.

22(D). **Show result on screen** ←3DS Requestor process

- Use the authentication result obtained to show the result on the UI or authentication value and eci can be used to continue with the authorization process.

> **What's next?**
>
> Access the **Integration guide** and go through the process of integrating a merchant checkout process with **ActiveServer**.