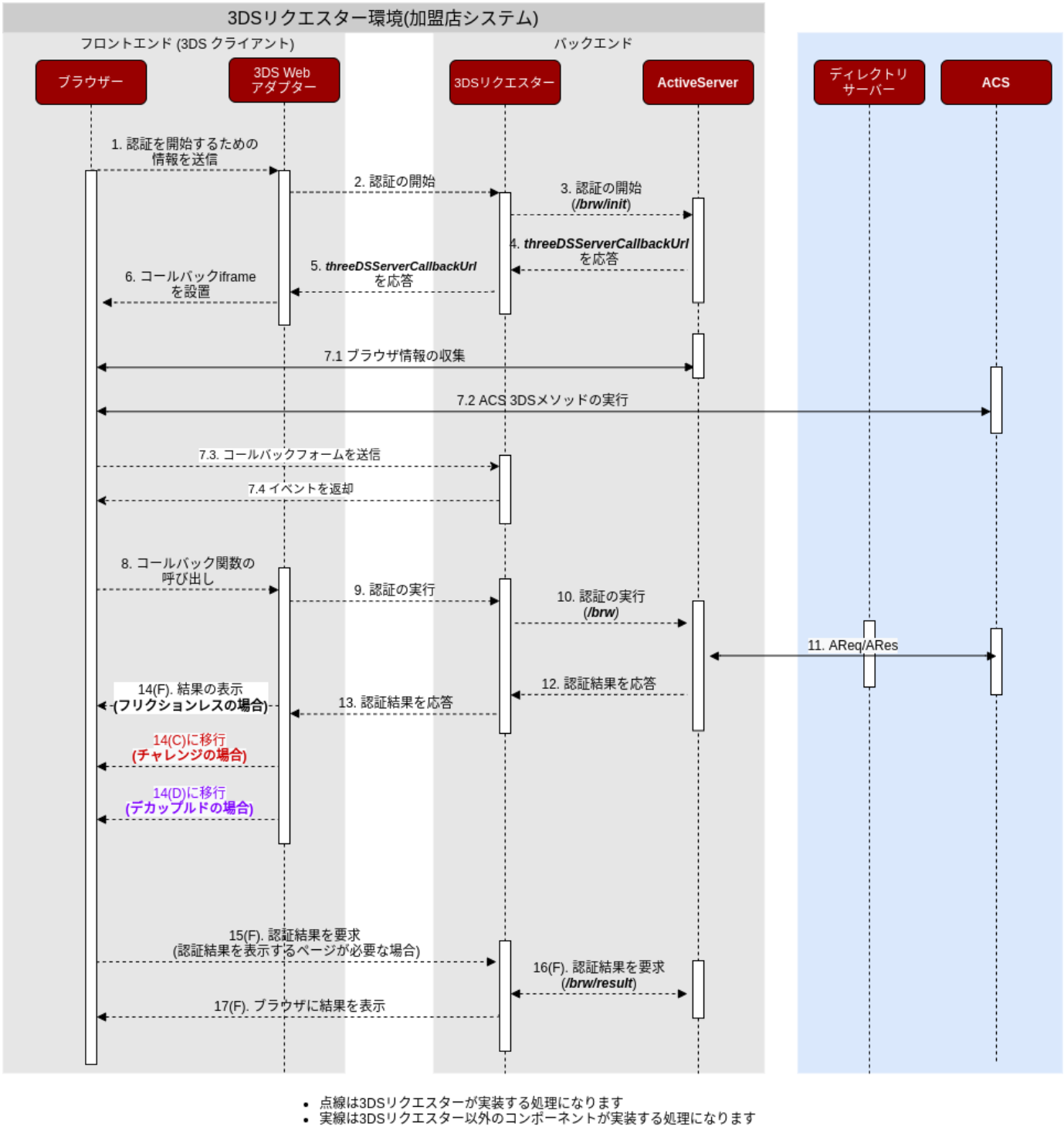
認証シーケンス

次のシーケンス図は、3DS2認証処理について、特に3DS2フローにおける**GPayments**の APIを使用した**3DSリクエスター** の役割に焦点を当てて各ステップを段階を追って示したものです。

> **備考**
>
> ActiveServerを統合するには、フロントエンドに**3DS web adapter**を、バックエンドに**3DSリクエスター**を実装する必要があります。

- **処理 1: 認証の初期化**
  - ステップ1～ステップ7
- **処理 2: 認証の実行**
  - フリクションレス・フロー: ステップ8～ステップ13、およびステップ14(F)
  - チャレンジ・フロー: ステップ8～ステップ13、およびステップ14(C)～ステップ19(C)
  - デカップルド・フロー: ステップ８～ステップ１３、およびステップ14(D)、ステップ17
- **処理 3: 認証結果の取得**
  - フリクションレス・フロー: ステップ15(F)～ステップ17(F)
  - チャレンジ・フロー: ステップ20(C)～ステップ22(C)
  - デカップルド・フロー: ステップ15(D)からステップ22(D)



- 点線は3DSリクエスターが実装する処理になります
- 実線は3DSリクエスター以外のコンポーネントが実装する処理になります

1. **認証の初期化用の情報を送信** ←3DSリクエスターの処理
   - カード番号やカード会員の氏名など、決済ページで得られたカード会員情報が、**3DS web adapter** に送信されます。これは加盟店のフロントエンドシステムをシミュレートした簡潔なJavaScriptのコードです。
2. **認証の初期化** ←3DSリクエスターの処理
   - **3DS web adapter**は決済ページから収集した情報を使用して**3DSリクエスター**へのPOSTリクエストを行い、**3DSリクエスター** に認証の初期化を要求します。

3. **認証の初期化 (`initAuth` ←3DSリクエスターの処理**
   - **3DSリクエスター**はフロントエンドから情報を取得し、 **`initAuth`** へのPOST API呼び出しを行って認証を初期化します。
   - ここで送信される重要なフィールドは `eventCallbackUrl` であり、**ActiveServer** がこのURLへのコールバックを行ってブラウザーの情報収集完了を通知できるようステップ8を開始するために必要です。
4. **`threeDSServerCallbackUrl` を応答**
   - **`initAuth`** からの正常な応答には `threeDSServerCallbackUrl` と `threeDSServerTransID` が含まれています。

> **RBC**
>
> `skipAutoBrowserInfoCollect` が `true` または3DSメソッドが利用できなかったり `acctNumber` が提供されなかった場合、レスポンスに含まれる `threeDSServerCallbackUrl` と `monUrl` は `null` になる可能性があります。この場合 `threeDSMethodAvailable` には `false` が設定されます。

5. **`threeDSServerCallbackUrl` を応答** ←3DSリクエスターの処理
   - **3DSリクエスター**は**3DS web adapter**に `threeDSServerCallbackUrl` を返します。
6. **コールバック `iframe` を設置** ←3DSリクエスターの処理
   - `src` 属性を `threeDSServerCallbackUrl` に設定した非表示の `iframe` を挿入します。これにより、**ActiveServer**は **3DSリクエスター**に接続できる状態になります。**ActiveServer**は、この `iframe` へのコールバックを行います。

> **RBC**
>
> `threeDSServerCallbackUrl` が `null` の場合、リクエスター側での `iframe` の設定をスキップできます。その際は10番目の手順に進んでください。

7. **ブラウザ情報の収集/3DSメソッドを実行**

   7.1. **ブラウザ情報の収集**
   - **ActiveServer**は `iframe` を通してブラウザー情報を収集します。このフィールドはAReqを送信する際に必要になります。

RBC

`skipAutoBrowserInfoCollect` が `true` の場合、ActiveServerは通常自動的に行っているブラウザ情報収集処理をスキップします。

7.2 3DSメソッドを実行

- ActiveServerは3DSメソッドがACSにより利用可能な場合、ACSの3DSメソッドURLをiframe内で実行することにより、**ACS** が3DSメソッドデータを収集できるようにします。**ACS**は、用意された `iframe` を使用して3DSメソッドデータを収集します。

7.3 コールバックフォームを自動送信

- ステップ7.1またはステップ7.2の結果、非表示のHTMLフォームを含むコールバックフォームが返却されます。このフォームは、ブラウザの情報収集または3DSメソッドが終了したことを3DSリクエスターに通知するため、レンダリングされるとすぐに送信されます。
- 3DSメソッドがACSによってサポートされていない場合は、 `event` パラメーターは `3DSMethodSkipped` に設定されます。それ以外の場合は `3DSMethodFinished` に設定されます。 `param` パラメーターにはActiveServerが収集したブラウザ情報をbase64エンコードしたものが設定されます。

3DSメソッドモニタリング

ActiveServerは、ACSが3DSメソッドの実行に失敗した場合のフェールセーフ機能も提供しています。3DSリクエスターは `monUrl` を用いて、3DSメソッドが利用可能であるにも関わらず10秒以内に完了しなかった場合に `InitAuthTimedOut` というイベントを通知する監視用 `iframe` を設定することができます。 `InitAuthTimedOut` イベントが3DSリクエスターに通知された場合は認証を続行することが推奨されます。 `param` パラメーターにはActiveServerが収集したブラウザ情報をbase64エンコードしたものが設定されます。

7.4 通知されたイベントを返却する

- 3DSリクエスターはイベントをフロントエンドに配信します。 イベントが通知され、それが `3DSMethodFinished` 、 `3DSMethodSkipped` 、または `InitAuthTimedOut` のいずれかである場合、ActiveServerが認証を実行する準備ができているため、3DSリクエスターは「認証実行」プロセスを続行できます。
- また、3DSメソッドmonUrlが実装され **、かつ3DSメソッドのタイムアウト期間後にACSから予期しない3DSメソッド通知があった場合、ActiveServer** は `3DSMethodHasError` イベントを3DSリクエスターに送信します。このイベントはトラブ

ルシューティングとログ記録のみを目的としておりますので、3DSリクエスターからフロントエンドには配信しないでください。詳細についてはデモコードを確認してください。

8. **コールバック関数の呼び出し** ←3DSリクエスターの処理
   - 認証の初期化中、3DSメソッドが終了またはスキップされたときにACSが**3DSリクエスター**に通知できるよう、**3DSリクエスター** は `eventCallBackUrl` を送信します。 ステップ７で設置した `iframe` からPOST要求がこの `eventCallBackUrl` に行われます。
   - **3DSリクエスター**は、この要求を受け `iframe` 内に必要な `callbackFn` 含めたパラメーターと一緒に `notify-3ds-events.html` を表示します。 `notify_3ds_events.html` は表示後 `3ds-web-adater` に定義された `callbackFn` を呼び出します。
9. **認証の実行** ←3DSリクエスターの処理
   - `callbackFn` は `_on3DSMethodSkipped()` 、 `_onInitAuthTimedOut()` または `_on3DSMethodFinished()` のいずれかであり、どちらも `doAuth()` を呼び出します。**3DS web adapter**は `doAuth()` を呼び出し、認証を実行するよう3DSリクエスターに要求します。
   - `_onInitAuthTimedOut` はACSの3DSメソッドプロセス処理がタイムアウトしたことを意味します。
   - `_on3DSMethodSkipped` はブラウザの情報がなんらかの理由によってACSが取得できなかったことを意味します。なので、もしこのコールバック関数が呼ばれた場合加盟店は認証を続行しない選択をすることもできます。
10. **認証の実行** (`auth`)
    - **3DSリクエスター**は `auth` を呼び出して認証処理を開始します。

> **RBC**
>
> ブラウザ情報を**3DSリクエスター**で収集した場合、 `auth` 呼び出し時に `browserInfoCollected` にデータを設定する必要があります。詳細はサンプルコードをご参照ください。

11. **AReq/ARes**
    - 認証リクエスト（AReq）は、ディレクトリ・サーバーを介して**ActiveServer**からACSに送信されます。ACSからは、認証結果を含む認証応答（ARes）が **ActiveServer**に送信されます。
12. **認証結果を応答**
    - `auth` は、**3DSリクエスター**に `tranStatus` を返します。

13. **認証結果を応答** ←3DSリクエスターの処理
    - 認証結果を3DS webアダプターに返します。

    "C"の場合は `ステップ 14(C)` に、"D"の場合は `ステップ 15(D)` に進んでください。
    `transStatus` が"Y"の場合は `ステップ 14(F)` に進むこともできますが、既に認証結果を取得しているのでこのステップは任意となります。

フリクションレス・フローの場合

14(F). **結果の表示（フリクションレスの場合)** ←3DSリクエスターの処理

- `ステップ.13` で最終的な認証結果を得たので、ここで説明する手順は**任意**です。別の画面で認証結果を確認する必要がある場合は、以下の手順を実行することができます。
- 認証結果の `transStatus` が"Y"の場合は `authSuccess()` が呼び出され、ページを `/auth/result?transId` にリダイレクトします。

15(F). **認証結果を要求 (認証結果を表示するページが必要な場合)** ←3DSリクエスターの処理

- ブラウザーが `transId` で**3DSリクエスター**に通知し、取引結果がリクエストに使用できる状態になります。

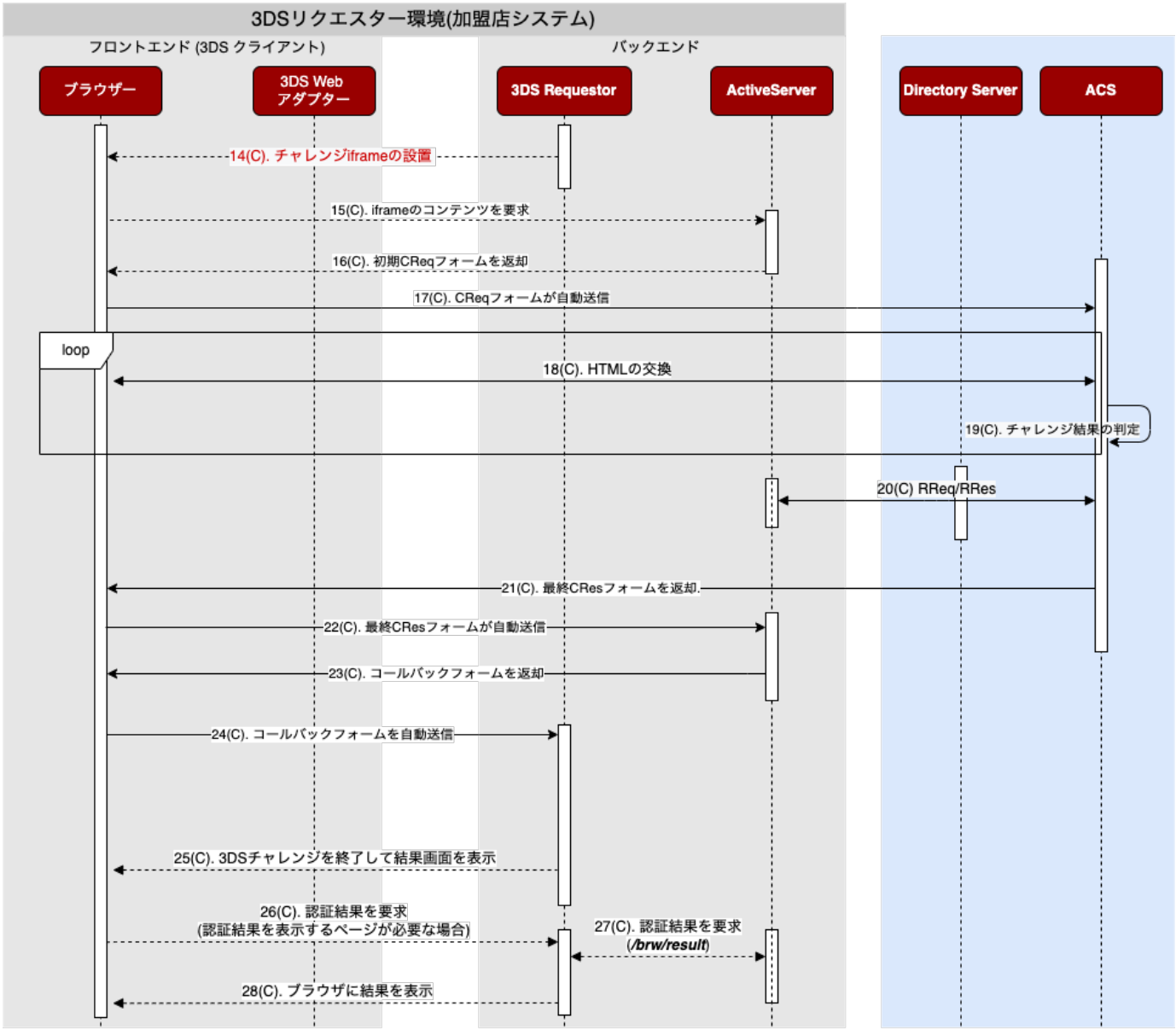
16(F). **認証結果を要求(`result`)**

- **3DSリクエスター**は `result` を呼び出し、**ActiveServer** から結果を受信するように要求します。

17(F). **ブラウザに結果を表示**

- 取得した認証結果を使用して、UI上に結果を表示するか、Authentication ValueとECIを使用してオーソリ処理に移行することができます。

チャレンジフローの場合



14(C). `iframe` の設置（チャレンジの場合）。 ←3DSリクエスターの処理

- 認証結果の `transStatus` が"C"である場合は `src` 属性を `challengeUrl` に設定したiframeを作成する必要があります。

15(C). チャレンジiframeのコンテンツを取得する

- ブラウザは `iframe` に `src` 属性を設定することで、ActiveServerから `iframe` のコンテンツを要求します。

16(C). **初期のCReqフォームを返却** * `iframe` には、ブラウザ内でフォームがレンダリングされたときに自動的に送信される初期CReqフォームが返却されます。

17(C). **CReqフォームが自動送信** * ActiveServerは、初期CReqフォームを自動的に送信し、初期CReqをAResで返された `acsURL` に送信して、ACSとのチャレンジプロセスを開始します。

18(C). **HTMLの交換**

- ACSは `iframe` 内にチャレンジ画面を埋め込み、カード会員は認証チャレンジを実行します。

19(C). **チャレンジ結果の判定**

- ACSは、実行されたチャレンジが成功したかいなかを判定します。

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

- ACSは、ディレクトリ・サーバーを介して**ActiveServer**に、認証結果を含む結果リクエスト（RReq）を送信します。**ActiveServer**　は、結果応答（RRes）を使用して受信確認通知を送信します。

21(C). **最終CResフォームを返却**

- ACSは、ブラウザ上で非表示の最終CResフォームをレンダリングします。

22(C). **最終CResフォームが自動送信**

- 最終的なCResフォームは、ブラウザ上でレンダリングされるとすぐに送信されます。ActiveServerはEMVCo仕様の要件に従って最終的なCResを検証し、ブラウザーの `iframe` にリクエスターに通知するためのコールバックフォームを返します。

23(C). **コールバックフォームを返却**

- ActiveServerからコールバックフォームが返ってきます。これは隠しフォームを含むHTMLで、ブラウザ上でレンダリングされると同時に自動的に送信されます。

24(C).　**コールバックフォームが自動送信**

- コールバックフォームは自動的に送信され、3DSリクエスターのバックエンドにイベント `AuthResultReady` を送信し、3DSリクエスター側に最終的な認証結果がActiveServerから要求可能であることを通知します。

25(C). **3DSチャレンジを終了して結果画面を表示** ←3DSリクエスターの処理

- チャレンジが終了したため、**3DSリクエスター**はページを `/auth/result?transId` にリダイレクトします。

26(C). **認証結果を要求(認証結果を表示するページが必要な場合)** ←3DSリクエスターの処理

- ブラウザーが `transId` で**3DSリクエスター**に通知し、取引結果がリクエストに使用できる状態になります。

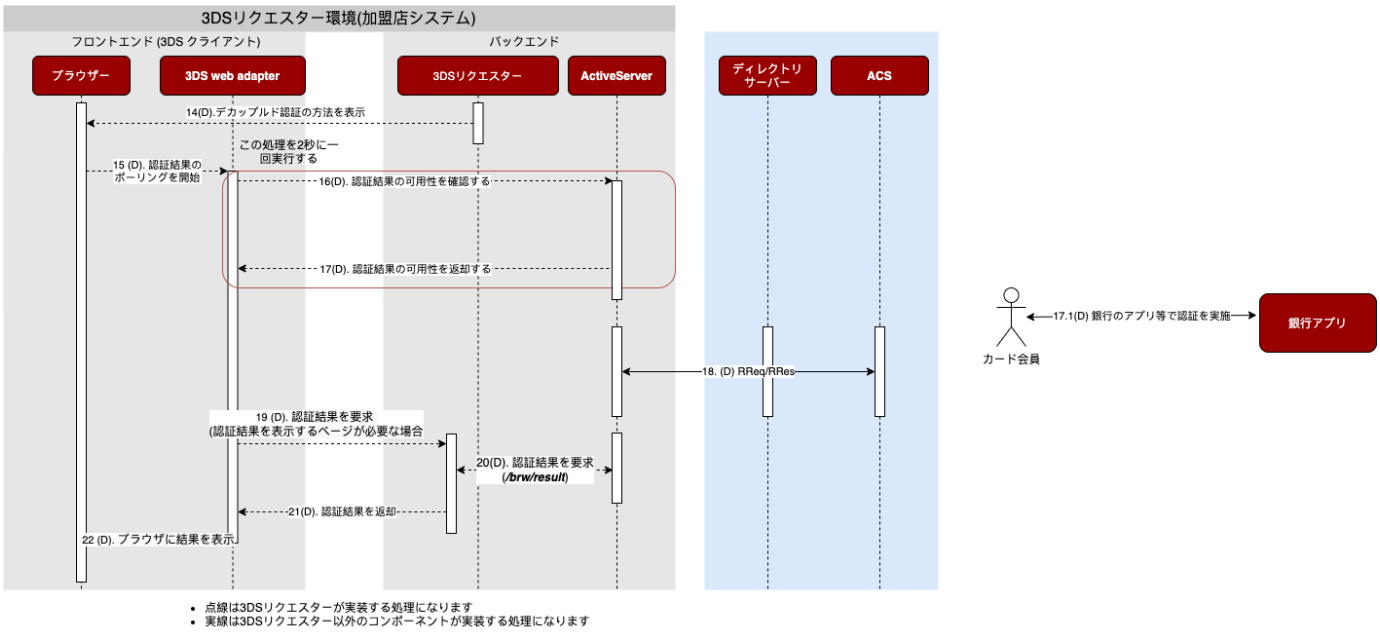
27(C). **認証結果を要求(`result`)** ←3DSリクエスターの処理

- ステップ16(F)と同様に、**3DSリクエスター**は `result` からの結果受信を要求します。

28(C). **ブラウザに結果を表示** ←3DSリクエスターの処理

- ブラウザーで結果画面が開き、認証結果が表示されます。

デカップルド・フローの場合



14(D). **デカップルド認証の方法を表示します** ←3DSリクエスターの処理

- 認証結果の `transStatus` が"D"で `acsDecConInd` が"Y"の場合。この場合、ACSはカード会員との間でデカップリング認証を行うことに同意したことになります。どのようにデカップルド認証が行われるかは3DSプロトコルの範囲対象外です。
- ACSは、カード会員が従うべき情報を含む `cardholderInfo` テキストを返します。(「取引を続行するには、銀行アプリを開いてください。」など。)カード会員に銀行アプリを開く

よう促し、生体認証などを実行します。3DSリクエスターは、このメッセージをUIに表示して、カード会員に認証を実行するために次に何をすべきかを表示する必要があります。

> **重要**
>
> `D` の `transStatus` と `Y` の `acsDecConInd` は、ステップ.9で `threeDSReqDecInd` が `Y` に設定されている場合にのみ返されます。つまり、3DSリクエスターによってデカップルド・フローが指定され、さらにACSによって同意されないとデカップルド・フローには移行しません。

15(D). **認証結果のポーリングを開始** ←3DSリクエスターの処理

- `resultMonUrl` は `Step.13` でも返却されます。これは、3DSリクエスターがActiveServerに認証結果を要求するタイミングを知ることができるURLです。このURLをブラウザから間隔を置いて呼び出し、ActiveServerから結果の可用性のステータスをポーリングできます。

16(D) **認証結果の可用性を確認する** ←3DSリクエスターの処理

- `resultMonUrl` は、ActiveServerからの認証結果の可用性を確認するために呼び出されます。
- デカップルド認証は3DSフローとは分離された環境で実行されるため、認証結果の可用性の確認は間隔を置いて実行される必要があります。

17(D) **認証結果の可用性を返却する** ←3DSリクエスターの処理 * 結果の可用性のステータスが返されます。認証結果がActiveServerから要求可能な場合、 `event` フィールドは `AuthResultReady` に設定され、そうでない場合は `AuthResultNotReady` に設定されます。

17.1(D) **銀行アプリなどで認証を実施**

- このステップは3DSフローの外部で実行され、この認証の実行方法は3DSの範囲外です(例：カード会員は登録済みの銀行アプリで認証できます。）。認証プロセスは、3DSリクエスターが指定した `threeDSReqDecMaxTime` を超えない限り、いつでも実行でき、この処理時間は最大10080分（168時間）まで設定可能です。

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

- ACSは、カード会員とのデカップルド認証を終えたあとまたは、 `threeDSReqDecMaxTime` を超えた後、認証結果を含む結果要求（RReq）をDirectoryServer経由で**ActiveServer**に送信します。 **ActiveServer**は、結果応答（RRes）を返却します。

19(D). **認証結果を要求(認証結果を表示するページが必要な場合)** ←3DSリクエスターの処理

- `ステップ. 17(D)` のポーリング結果はで `AuthResultReady` を返します。これは、ActiveServerがRReqを受信し、最終的な認証結果を要求できるようになったためです。

20(D). **認証結果を要求(** `result` **)** ←3DSリクエスターの処理

- ステップ16(F)、ステップ27(C)と同様に、**3DSリクエスター**は `result` からの結果受信を要求します。

21(D). **認証結果を返却** ←3DSリクエスターの処理

- 認証結果は**3DSWebアダプター**に返されます。

22(D). **ブラウザーに結果を表示** ←3DSリクエスターの処理

- 取得された認証結果、eciを使用して、UIに認証結果を表示するか、オーソリ処理に移行できます。

> **次のチャプター**

下のフッターの**次**を選択して**実装ガイド**にアクセスし、**ActiveServer**を使用した加盟店の決済処理機能を組み込んでください。