3ds1 integration

> **3DS 1.0.2は、GPaymentsSaaS製品でのみサポートしています。**
>
> 3DS 1.0.2 での開発は SaaS でのみサポートされており、 ではご利用いただけません。

ActiveServer は現在、SaaS向けの3DS1認証をサポートしています. 3DS1認証を加盟店または決済代行会社のeコマースサイトと統合するには、 eコマースサイトのチェックアウトプロセスで、 **ActiveServer** バックエンドへのAPI呼び出しと次のページフローを含む **3DS1チャレンジプロセス** およびACSによって初期化されたチャレンジプロセスを持つページフローを実装する必要があります。

ActiveServerでの3DS1認証プロセスまとめ

3DS1認証を実行する方法は下記の通りです:

1. **3DSリクエスター** (または加盟店のチェックアウトプロセス) は3DS1認証が必要か定義する。
2. **3DSリクエスター** (または加盟店のチェックアウトプロセス) は `ThreeDS1AuthReq` メッセージを利用して **ActiveServer**の3DS1認証APIを呼び出す。
3. **3DSリクエスター** は現在のチェックアウトページから、2の応答 `ThreeDS1AuthResp` によって返されるチャレンジページにリダイレクトします。
4. チャレンジページはACSのカード会員認証プロセスを実行し、2のメッセージ「ThreeDS1AuthReq」で **3DSリクエスター** によって提供された結果通知URLに対して、POSTフォームを介して認証結果を返します。
5. **3DSリクエスター** の認証結果は、結果通知URLを介して入手できます。**3DSリクエスター** は、それに応じて結果を処理します。

> **認証API - 認証方法**
>
> 3DS1 Auth API呼び出しの認証では、既存の3DS2プロセスと同じmerchant/masterAuth証明書が使用されることに注意してください。証明書の使用法の詳細についてはAPIドキュメントまとめを参照してください。

ステップ 1: 3DS1 認証方法の定義

3DS1認証プロセスを実行する前に、ユーザーは3DS1認証が必要かどうかを確認することをお勧めします。 **3DSリクエスター**は、カード所有者または販売者の情報に基づいて3DS1認証プロセスを開始する静的プロトコルルーティングプロセスを実装するか、 **ActiveServer** のEnrol APIを使用して、PANが3DS2に登録されているかどうかを確認します。3DS認証が必要な場合にEnrol API呼び出しが `00` （3DS2に登録されていない際のコード）を返す場合、**3DSリクエスター** は3DS1認証を実施できます。

> **3DS 1.0.2 認証の定義**
>
> トランザクションを3DS1または3DS2のどちらで認証するかは、**3DSリクエスター**が決定します。 Enrol APIの使用は必須ではなく、3DS1認証が必要かどうかを判断するための任意のオプションとして提供されています。

ステップ 2: 3DS1認証APIの呼び出し

3DS1統合は、2か所に分かれます。Auth API呼び出しを初期化し、結果通知ページをホストするバックエンドと、Auth API呼び出しの応答をチェックし、ACSチャレンジページをリダイレクト/ホストするフロントエンドJavascriptメソッドの2つです。

ActiveServer の3DS1統合は、3DS2統合と同じ言語とフレームワークをサポートします。3DS1統合を進める前に、 統合ガイドを参考に、提供されたデモ版リクエスターのコード、選択した言語/フレームワークを使用してローカルテスト環境をセットアップしてください。

3DS1認証APIを呼び出す前に、**ActiveServer**との相互認証TLS通信をセットアップするために適切なクライアント証明書が必要です。詳細については、バックエンド実装v2を参照してください。

バックエンドの `ThreeDS1AuthReq` の呼び出し部のコードスニペットは下記をご参照ください。:

Java PHP C# Go

```
//MainController3DS1.java
  @ResponseBody
  @PostMapping(value = "/3ds1/auth")
  public ThreeDS1AuthResp auth(@RequestBody ThreeDS1AuthReq req) {
    logger.info("3ds1 auth request received: {}", req);
    return threeDS1Service.handleAuthRequest(req);
  }

//ThreeDS1Service.java
  ThreeDS1AuthResp handleAuthRequest(ThreeDS1AuthReq request) {

    //generate the transaction id, this is optional.
    request.setThreeDSRequestorTransID(UUID.randomUUID().toString());

    logger.info("sending 3ds1 auth request to ActiveServer: {}", authUrl);

    HttpEntity<ThreeDS1AuthReq> httpRequest;
    if (config.isGroupAuth()) {
      HttpHeaders headers = new HttpHeaders();
      headers.add(AuthServiceV2.AS_MERCHANT_TOKEN_HEADER,
config.getMerchantToken());
      httpRequest = new HttpEntity<>(request, headers);
    } else {

      httpRequest = new HttpEntity<>(request);
    }

    ResponseEntity<ThreeDS1AuthResp> response = restTemplate
        .postForEntity(authUrl, httpRequest, ThreeDS1AuthResp.class);

    if (response.getStatusCode() == HttpStatus.OK) {

      ThreeDS1AuthResp body = response.getBody();
      logger.info("server returns ok, content: {}", body);

      return body;

    } else {
      ThreeDS1AuthResp body = response.getBody();
      logger.error("server returns error code: {}, content: {}",
response.getStatusCode(),
          body);
      return body;
    }

  }
```

上記コードの通り、3DS1認証API呼び出しは、フロントエンドから**ActiveServer** 認証APIへ送信されたJSONリクエストを転送しています。相互認証されたHTTPクライアントのロードおよびマスター認証証明書の処理を行うコードは、このガイドには記載はありませんが3DS2コードと同じです。

> **3DSリクエスター API 認証**
>
> フロントエンドページフローで独自のAPI認証を処理するかは**3DSリクエスター**の実装次第であり、当ガイドではスコープ外です。

結果通知URLの準備

`ThreeDS1AuthReq` を **ActiveServer** に送信する場合、**3DSリクエスター**（または加盟店サイト）は、イシュアーのACSから認証結果を受信するための通知ページを用意する必要があります。カード所有者がチャレンジフォームを送信すると、ACSは結果（CAVV、ECIなど）をこの通知ページに送ります。

それに応じて**3DSリクエスター**は、バックエンドで結果を処理できます。 このガイドでは、受信した認証結果を結果ページに表示する部分のみご紹介します。

Java PHP C# Go

```
//MainController3DS1.java
  @PostMapping("/3ds1/result")
  public String resultPage(Model model, @RequestBody MultiValueMap<String,
String> body) {
    logger.info("received result: {}", body);

    model.addAttribute("cavv", body.getFirst("cavv"));
    model.addAttribute("cavvAlgo", body.getFirst("cavvAlgo"));
    model.addAttribute("eci", body.getFirst("eci"));
    model.addAttribute("threeDSRequestorTransID",
body.getFirst("threeDSRequestorTransID"));
    return "3ds1/result";
  }
```

デモコードの通り、結果通知URLは `/ 3ds1 / result` となるため、`ThreeDS1AuthReq` のフィールド `callbackUrl` には `https://<3DSリクエスターのベースURL>/3ds1/result` を設定する必要があります。

以下のコードスニペットは、デモページで `callbackUrl` がどのように設定されているかをご紹介します。

Java PHP C# Go

```
//MainController3DS1.java
  @GetMapping("/3ds1")
  public String paymentPage(Model model) {
    model.addAttribute("authUrl", config.getAsAuthUrl());
    model.addAttribute("callbackUrl", config.getBaseUrl() + "/3ds1/result");

    logger.info("3ds1 auth page called");
    return "3ds1/auth";
  }
```

ステップ3: 認証APIのレスポンスとACSチャレンジページの処理

ステップ2を経て、**ActiveServer**はDirectoryServerとACSを使用して内部で認証要求を処理します。正常終了すれば**ActiveServer**から `ThreeDS1AuthResp` 応答が返されます。 応答の詳細については、ActiveServer認証APIレファレンス をご確認ください。

レスポンスの `errorCode` がnullでない場合はエラーとみなされます。フロントエンド（またはバックエンド、実際の実装によってエラーの処理方法は異なる場合があります）はそれをエラーとして処理し、認証プロセスを再試行するようカード所有者に促す必要があります。

レスポンスの `errorCode` がnullの場合、フィールド `challengeUrl` が返されます。

以下のJavascriptコードは、現在のページをリダイレクトしてチャレンジURLを読み込む方法と、 `ThreeDS1AuthReq` がエラーを返したときにエラーメッセージを表示する簡易的な例を示しています。

Javascript

```
//3ds1/auth.html
<script src="/js/v2/3ds-web-adapter.js"></script>
<script>

  function handleResponse(response) {
    //use the challenge url returned from the response
    if (response.errorCode) {
      console.error("error response", _onError);
      alert("auth request returns error: \n\n" + JSON.stringify(response))
    } else {
      console.log("response:", response)
      window.location.href = response.challengeUrl; //show the challenge url on
current page
    }
  }

  function handleError(response) {
    console.error("error", response);
    alert("auth request returns error: \n\n" + JSON.stringify(response))
  }

  $("#btnAuth").click(function () {

    var authData = objectifyForm($("#authForm").serializeArray());
    console.log(authData);
    doPost("/3ds1/auth", authData, handleResponse, handleError);

  })

  function objectifyForm(formArray) {
    //serialize data function
    var returnArray = {};
    for (var i = 0; i < formArray.length; i++) {
      returnArray[formArray[i]['name']] = formArray[i]['value'];
    }
    return returnArray;
  }

</script>
```

上記のJavaScriptスニペットの通り、チャレンジURLは `window.location.href` を設定することによって提示され、エラーは単純な `alert()` 関数によって表示されます。

ステップ4&5: 結果通知の処理

ステップ2で述べたように、 `ThreeDS1AuthReq` 呼び出しの結果通知ページを用意する必要があります。ACSがチャレンジフローを終了すると、認証結果を含むPOSTフォームが結果通知ページに送られます。この時に**3DSリクエスター**のバックエンドは結果を処理する必要があります。

Java PHP C# Go

```
//MainController3DS1.java
  @PostMapping("/3ds1/result")
  public String resultPage(Model model, @RequestBody MultiValueMap<String,
String> body) {
    logger.info("received result: {}", body);

    model.addAttribute("errorCode", body.getFirst("errorCode"));
    model.addAttribute("errorMessage", body.getFirst("errorMessage"));
    model.addAttribute("txStatus", body.getFirst("txStatus"));
    model.addAttribute("cavv", body.getFirst("cavv"));
    model.addAttribute("cavvAlgo", body.getFirst("cavvAlgo"));
    model.addAttribute("eci", body.getFirst("eci"));
    model.addAttribute("threeDSRequestorTransID",
body.getFirst("threeDSRequestorTransID"));
    return "3ds1/result";
  }
```

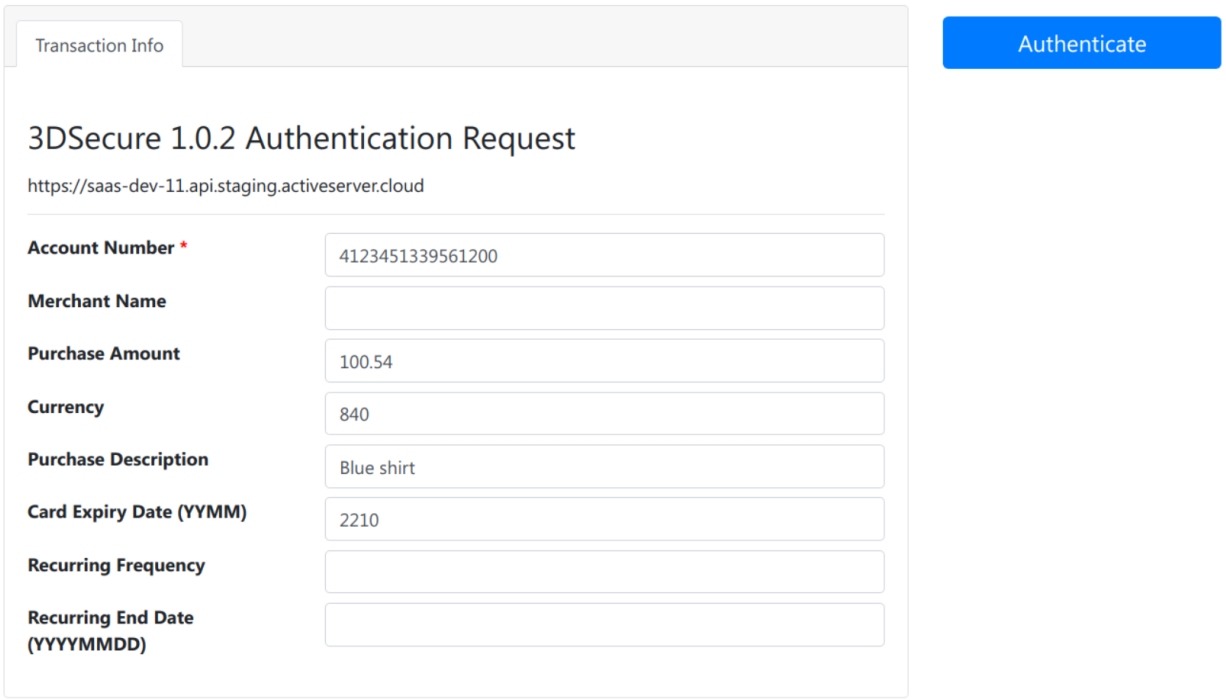
- **cavv**: Cardholder Authentication Verification Value（カード会員認証用検証値）の略です。一部の地域のVISAでは3-Dセキュア認証の証明として必要になります。
- **cavvAlgo**: CAVVアルゴリズムのことです。一部の地域のVISAでは3-Dセキュア認証の証明として必要になります。
- **eci**: ActiveMerchantにて定義されます。支払い用ゲートウェイインターフェースを通じてアクワイアラに送信する必要があります。
- **threeDSRequestorTransID**: 3DS1のトランザクションIDです。EnrolReqでは“XID”として使われます。
- **txStatus**: 3DS1の仕様ではPARes_TX_Statusとして定義されている内容です。
- **errorCode**: 返却されたエラーコードです。エラーがない場合は“0”が返却されます。
- **errorMessage**: errorCodeが“0”以外の場合にエラーメッセージが設定されます。

3DSリクエスターのデモによる3DS1.0統合のウォークスルー

これで、**ActiveServer**を使用して独自の3DS1リクエスターを実行できます。参考として、以下にデモ3DSリクエスターのGPayments 3DS1TestLabを介した3DS1認証プロセスのスクリーンショットを添付します。

認証リクエストの送信

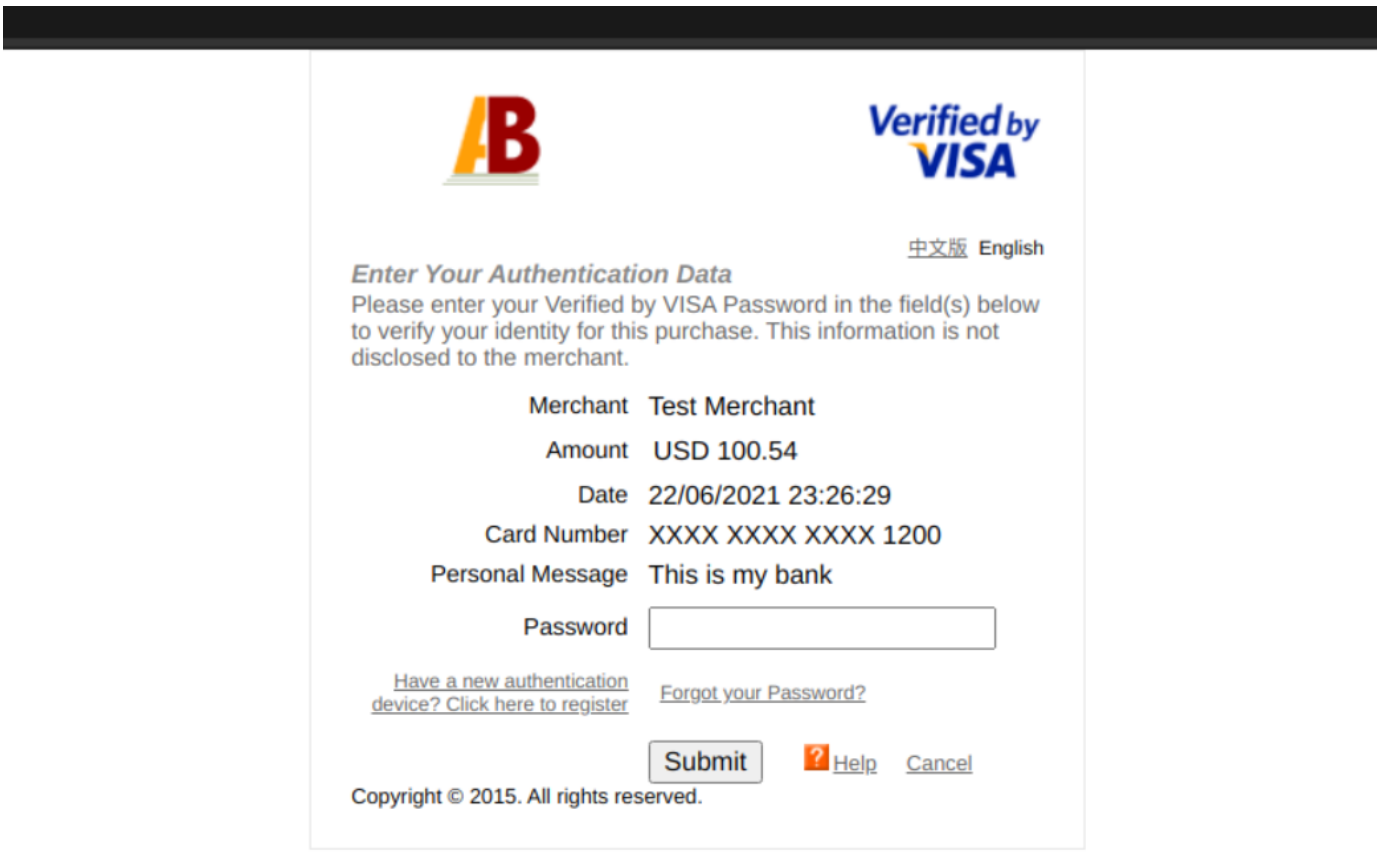
デモリクエスターでは、単純なフォームを使用して `ThreeDS1AuthReq` リクエストを入力し、それをJSONメッセージとして3DSリクエスターバックエンドに送信しています。その後バックエンドはリクエストを**ActiveServer**に転送します。 この時デモコードでは、認証リクエストフォームが最初にJSONデータとしてシリアル化されてから、バックエンドに送信されることに注意してください。



チャレンジページの表示

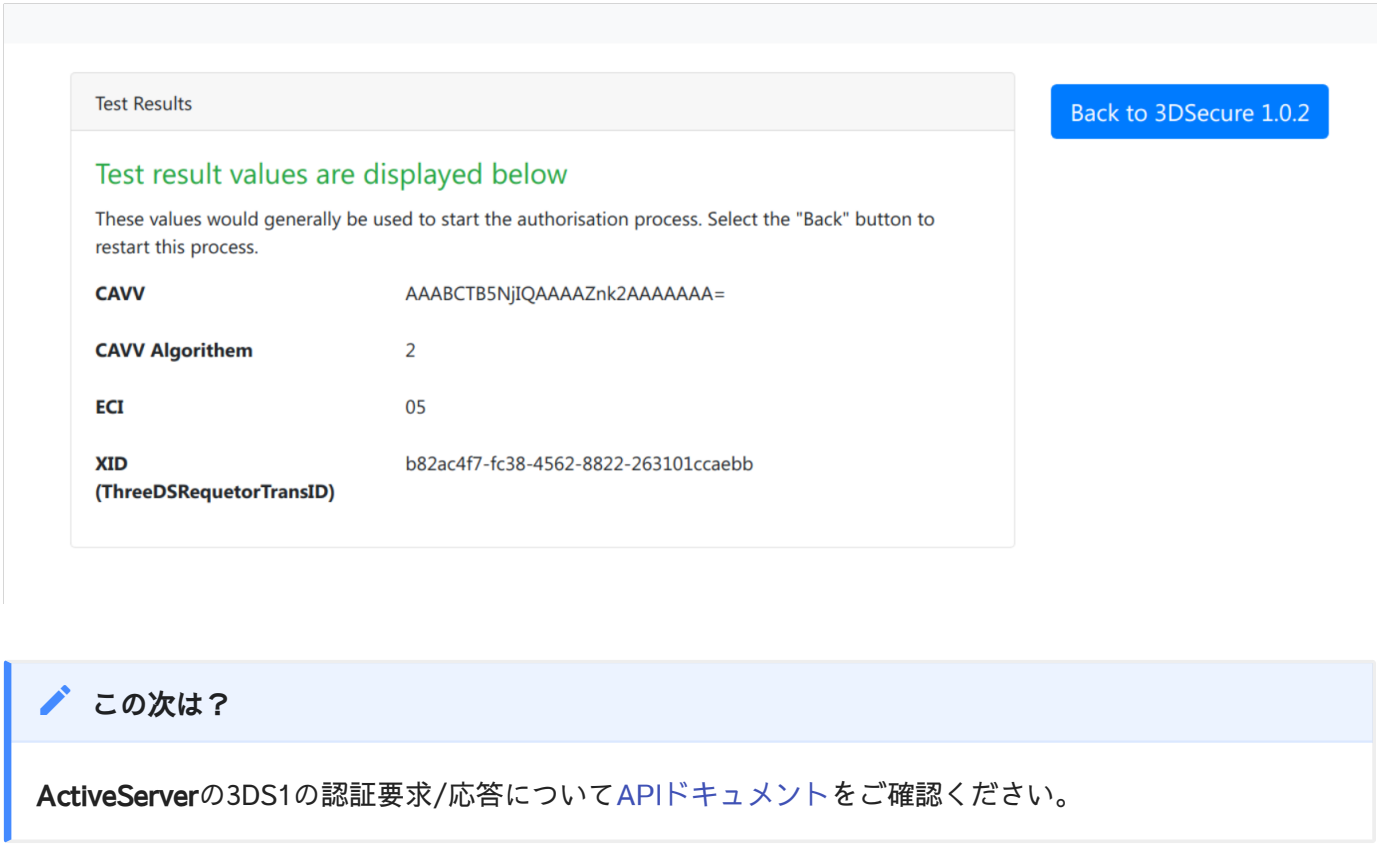
認証要求が**ActiveServer**に送信されると、3DS1プロトコルが処理され、ACSはカード所有者のブラウザに表示されるチャレンジページを返します。

以下のスクリーンショットはGPayments TestLabのACSチャレンジページです。



結果の表示

カード所有者がチャレンジページで情報を送信すると、ACSは入力を確認し、最初の `ThreeDS1AuthReq` で準備および提供されたフォームを `callbackUrl` に送信し認証結果を返します。



この次は？

ActiveServerの3DS1の認証要求/応答についてAPIドキュメントをご確認ください。