Back end v1

In this section, we will show the implementation details for the back-end side of the merchant application using our sample code based on the **Auth API version 1**.

> **Important**
>
> GPayments strongly recommends all new implementations of **ActiveServer** to integrate with version 2 of the Auth API from the beginning, as version 1 will be deprecated in a future release. Refer to the API migration from v1 to v2 guide for a summary of the changes and migration process from **v1** to **v2**. To check the new **API v2** implementation, refer here.

For the back-end, we need to implement a **3DS Requestor**. The **3DS Requestor** receives information from the **3ds-web-adapter** and sends the requests to **ActiveServer**. It also receives the authentication results from **ActiveServer** and forwards the results to the **3ds-web-adapter**.

The demo **3DS Requestor** code provides the backend implementation with the following server side languages:

- **Java**: The java version is implemented based on the Springboot framework. For details of Springboot, check out https://spring.io/projects/spring-boot
- **C#**: The C# version is implemented based on ASP.net.
- **PHP**: The PHP version is implemented based on PHP 7.2 with cURL (Client URL Library).
- **Go**: The Go version is implemented based on Go 1.12 with Go Module support. All dependencies are listed in `go.mod` file.

> **Why we need to implement a backend?**
>
> As defined by EMVCo's 3DSecure 2.0 specification, when the 3DS Server and the 3DS Requestor environment is separated, the communication between these two components must be mutual authenticated:
>
> > [Req 300] If the 3DS Requestor and 3DS Server are separate components, ensure that data transferred between the components is protected at a level that satisfies Payment System security requirements with mutual authentication of both servers.

Copyright ©2019 GPayments Pty Ltd. All rights reserved.

Before starting the authentication process with **ActiveServer**, the **3DS Requestor** needs to establish a mutual TLS connection with **ActiveServer**. Make sure you have the **client certificate** and configure it with the **3DS Requestor**. If not, follow the Get client certificate and Configure 3DS Requestor details instructions found on the Introduction page.

> **Tip**
>
> The implementation of TLS configuration for the HTTP Client can be found as follows:
>
> - Java: The TLS configuration and client certificate loading can be found in class `RestClientConfig.java`.
> - C#: The TLS configuration and client certificate loading can be found in class `RestClientHelper.cs`.
> - PHP: The TLS configuration and client certificate loading can be found in file `RestClientConfig.php`.
> - Go: The TLS configuration and client certificate loading can be found in file `https.go`.

Next, we will describe the details of the back-end implementation based on the authentication processes and authentication sequence.

Process 1: Initialise the Authentication

To initialise authentication, the **3DS Requestor** needs to:

- Receive the `Initialise authentication` request from the **3DS-web-adapter** (Step. 2).
- Forward the request to **ActiveServer** (Step. 3).
- Receive the response data from **ActiveServer** (Step. 4).
- Return the response data to the **3DS-web-adapter** (Step. 5).

Copyright ©2019 GPayments Pty Ltd. All rights reserved.

Java C# PHP Go

```java
//AuthControllerV1.java
/**
 * Receives the initialise authentication request from the 3DS-web-adapter
(Step 2)
 * Send data to ActiveServer to Initialise Authentication
 */
  @PostMapping("/v1/auth/init/{messageCategory}")
  public Message initAuth(@RequestParam(value = "trans-type", required = false)
String transType,
      @RequestBody Message request,
      @PathVariable(value = "messageCategory") String messageCategory) {

    //Generate requestor trans ID
    String transId = UUID.randomUUID().toString();
    request.put("threeDSRequestorTransID", transId);
    //Fill the event call back url with requestor url + /3ds-notify
    String callBackUrl = config.getBaseUrl() + "/3ds-notify";
    request.put("eventCallbackUrl", callBackUrl);

    //ActiveServer url for Initialise Authentication
    String initAuthUrl = config.getAsAuthUrl() + "/api/v1/auth/brw/init/" +
messageCategory;

    //Add parameter trans-type=prod in the initAuthUrl to use prod DS,
otherwise use TestLabs DS
    //For example, in this demo, the initAuthUrl for transactions with prod DS
is https://api.as.testlab.3dsecure.cloud:7443/api/v1/auth/brw/init?trans-
type=prod
    //For more details, refer to: https://docs.activeserver.cloud
    if ("prod".equals(transType)) {
      initAuthUrl = initAuthUrl + "?trans-type=prod";
    }

    logger.info("initAuthRequest on url: {}, body: \n{}", initAuthUrl, request);

    //Send data to ActiveServer to Initialise authentication (Step 3)
    //Get the response data from ActiveServer (Step 4)
    Message response =
        sendRequest(initAuthUrl, request, HttpMethod.POST);
    logger.info("initAuthResponseBRW: \n{}", response);

    //Return data to 3ds-web-adapter (Step 5)
    return response;
}
```

Copyright ©2019 GPayments Pty Ltd. All rights reserved.

Note

We set the `eventCallbackUrl` to `{baseUrl}/3ds-notify`. This allows **ActiveServer** to make a notification when the browser information collection (Step. 7) is done. The `baseUrl` is set here.

The `initAuth` request will be sent to `{ActiveServer auth url}/api/v1/auth/brw/init/{messageCategory}`. To check the data structure of `initAuth` request, refer to the API document.

Important: By default, the URL above will send the authentication request to GPayments TestLabs for testing purposes. When moving into production, to send the API request to the card scheme directory server, the *trans-type query parameter* must be appended to this API URL. See API description for further information on usage.

HTTP header for Master Auth API client certificate

If you are using a Master Auth API client certificate to authenticate a **Business Admin** user on behalf of a merchant, the back-end needs to add a HTTP Header in the HTTP Request with a field of `AS-Merchant-Token`, which should be set to the merchantToken from the merchants profile.

Copyright ©2019 GPayments Pty Ltd. All rights reserved.

Java C# PHP Go

```
//Note: the sendRequest() function will send the request to ActiveServer.
//If this is groupAuth, the request should include a HTTP Header with the field
of AS-Merchant-Token.
private Message sendRequest(String url, Message request, HttpMethod method) {

    HttpEntity<Message> req;
    HttpHeaders headers = null;

    if (config.isGroupAuth()) {
      //the certificate is for groupAuth, work out the header.
      headers = new HttpHeaders();
      headers.add("AS-Merchant-Token", config.getMerchantToken());
    }

    switch (method) {
      case POST:
        req = new HttpEntity<>(request, headers);
        return restTemplate.postForObject(url, req, Message.class);
      case GET:
        if (headers == null) {
          return restTemplate.getForObject(url, Message.class);
        } else {
          req = new HttpEntity<>(headers);
          return restTemplate.exchange(url, HttpMethod.GET, req,
Message.class).getBody();
        }
      default:
        return null;
    }
  }
}
```

Process 2: Execute Authentication

To execute authentication, the **3DS Requestor** needs to:

- Handle the `/3ds-notify` message from **ActiveServer** called through the iframe after Step. 7.
- Handle the `Execute authentication` request from the **3DS-web-adapter** (Step. 9 and Step. 10).

Copyright ©2019 GPayments Pty Ltd. All rights reserved.

- Receive the authentication result and return it to the **3DS-web-adapter** (Step. 12 and Step. 13).

When the browser information collection (Step. 7) is done, **ActiveServer** makes a notification to the `eventCallbackUrl` which we set to `http://localhost:8082/3ds-notify`. The **3DS Requestor** handles this notification and passes the required parameters to the `notify-3ds-events.html` page.

Java C# PHP Go

```
//MainController.java
@PostMapping("/3ds-notify")
public String notifyResult(
    @RequestParam("requestorTransId") String transId,
    @RequestParam("event") String callbackType,
    @RequestParam(name = "param", required = false) String param,
    Model model) {

  String callbackName;
  if ("3DSMethodFinished".equals(callbackType)) {

    callbackName = "_on3DSMethodFinished";

  } else if ("3DSMethodSkipped".equals(callbackType)) {

    callbackName = "_on3DSMethodSkipped";

  } else if ("AuthResultReady".equals(callbackType)) {

    callbackName = "_onAuthResult";

  } else if ("InitAuthTimedOut".equals(callbackType)) {

   callbackName = "_onInitAuthTimedOut";

  } else {
    throw new IllegalArgumentException("invalid callback type");
  }

  model.addAttribute("transId", transId);
  model.addAttribute("callbackName", callbackName);
  model.addAttribute("callbackParam", param);

  return "notify_3ds_events";
}
```

Copyright ©2019 GPayments Pty Ltd. All rights reserved.

Note

This handler is called by **ActiveServer**, parameters `requestorTransId`, `event` and an optional `param` will be provided by **ActiveServer**. The `event` can be `3DSMethodFinished`, `3DSMethodSkipped`, `InitAuthTimedOut` or `AuthResultReady`. For descriptions of each event refer to our API document for the field `eventCallbackUrl`. The handler method then sets the proper attribute in the page context and returns to the `notify_3ds_events.html` page. The page then renders the content with Mustache templating engine.

To check the front-end implementation of `notify_3ds_events.html`, refer here.

Then when the 3DS client is finished browser information collecting, it will call the `auth` endpoint to start the authentication. The **3DS Requestor** handles the `Execute authentication` requests (Step. 9, Step. 10, Step. 12, and Step. 13).

Java C# PHP Go

```java
//AuthControllerV1.java
/**
 * Receives the Execute authentication request from the 3DS-web-adapter (Step 9)
 * Send data to ActiveServer to Execute Authentication
 */
@PostMapping("/v1/auth")
public Message auth(@RequestBody Message request) {

    //ActiveServer url for Execute Authentication
    String authUrl = config.getAsAuthUrl() + "/api/v1/auth/brw";
    logger.info("requesting BRW Auth API {}, body: \n{}", authUrl, request);

    //Send data to ActiveServer to Execute Authentication (Step 10)
    //Get the response data from ActiveServer (Step 12)
    Message response =
        sendRequest(authUrl, request, HttpMethod.POST);
    logger.info("authResponseBRW: \n{}", response);

    //Return data to 3ds-web-adapter (Step 13)
    return response;
}
```

Copyright ©2019 GPayments Pty Ltd. All rights reserved.

> **Note**
>
> The **3DS Requestor** returns a response to the front-end. The returned message contains a `transStatus` with value `Y` or `C`, which will trigger either frictionless flow or challenge flow. To check how the front-end handles the `transStatus`, refer here. To check the structure of the response data, refer to the API document.

Cancel challenge flow

When the `transStatus=C`, the 3DS client can choose to start the challenge or not. If the 3DS client chooses to cancel the challenge, it may call the **`challengeStatus`** endpoint to specify the Cancel Reason which the **3DS Requestor** will send to **ActiveServer**. To check how the **front-end** handles this request, refer here.

Java C# PHP Go

```
//AuthControllerV1.java
@PostMapping("/v1/auth/challenge/status")
public Message challengeStatus(@RequestBody Message request) {

    String challengeStatusUrl = config.getAsAuthUrl() + "/api/v1/auth/challenge/
status";
    logger.info("request challenge status API {}, body: \n{}",
challengeStatusUrl, request);

    Message response =
        sendRequest(challengeStatusUrl, request, HttpMethod.POST);
    logger.info("challengeStatus response: \n{}", response);

    return response;
}
```

Process 3: Get Authentication Result

To get the authentication result, the **3DS Requestor** needs to:

- Handle the `Request for authentication result` request from the **3DS-web-adapter** (Step. 15(F) or Step. 20(C)).

Copyright ©2019 GPayments Pty Ltd. All rights reserved.

- Send a request to **ActiveServer** to get the result and return the result to the front-end. (Step. 16(F), Step. 17(F) or Step. 21(C), Step. 22(C)).

Java C# PHP Go

```
//AuthControllerV1.java
/**
 * Receives the Request for authentication result request (Step 15(F) and Step
20(C))
 * Send data to ActiveServer to Retrieve Authentication Results
 */
@GetMapping("/v1/auth/result")
public Message result(@RequestParam("txid") String serverTransId) {

    //ActiveServer url for Retrieve Results
    String resultUrl = config.getAsAuthUrl() +
        "/api/v1/auth/brw/result?threeDSServerTransID=" +
        serverTransId;

    //Get authentication result from ActiveServer (Step 16(F) and Step 21(C))
    Message response =
        sendRequest(resultUrl, null, HttpMethod.GET);
    logger.info("authResponse: \n{}", response);
    //Show authentication results on result.html (Step 17(F) and Step 22(C))
    return response;
}
```

Success

This covers the backend implementation. After authentication is completed the checkout process can move on to performing authorisation using the Transaction Status, ECI and CAVV to complete the transaction.

Whats next?

Select **Next** to go through all the features of our **Sample Code** for the **GPayments 3DS Requestor**.

Copyright ©2019 GPayments Pty Ltd. All rights reserved.