Sample Code Guide
Amazon Pay Checkout v2

Version 0.1

2019/09/01

Document 有効期限 2019/12/31

Contents

はじめに

Amazon Pay Checkout v2 は 2019 年 9 月 1 日現在 Pilot バージョンであり、今後予定される機能を全て使用することができません。また、正式バージョンリリース時に一部の仕様が変更されることがあります。その旨ご理解の上お取り扱いいただけますようお願いいたします。

この資料の位置づけ

この資料は Amazon Pay Checkout v2 をお試しいただきつつ、動作をご理解いただくためのサンプルコードの解説書です。別途ご用意しておりますサンプルコードと合わせてご確認ください。なお、サンプルコードは PHP の SDK を使用することを想定しており、PHP が利用できる環境をお持ちであることを前提条件といたします。PHP の SDK については下記を参照してください。

PHP SDK

https://github.com/amzn/amazon-pay-sdk-v2-php

環境準備

サンプルコードの内容

下表はサンプルコード内のフォルダとファイルの構成です。PHP が動作するテスト用の WEB サーバー等に配置してください。以降の説明にて括弧書きで行番号とファイル名を記載することがあります。（例. (03) amazon-pay-v2.phar）
その際は下表と対比してご確認ください。

フォルダとファイル構成

```
01>./
02> ├AmazonPayV2
03> │  └amazon-pay-v2.phar
04> ├css
05> │  ├common.css
06> │  ├uikit-rtl.css
07> │  ├uikit-rtl.min.css
08> │  ├uikit.css
09> │  └uikit.min.css
10> ├js
11> │  ├common.js
12> │  ├uikit-icons.js
13> │  ├uikit-icons.min.js
14> │  ├uikit.js
15> │  └uikit.min.js
16> ├keys
17> │  └private.pem
18> ├php
19> │  ├ap_sample_cancelCharge.php
```

```
20>  ├─ap_sample_captureCharge.php
21>  ├─ap_sample_closeChargePermission.php
22>  ├─ap_sample_config.php
23>  ├─ap_sample_createCharge.php
24>  ├─ap_sample_createCheckoutSession.php
25>  ├─ap_sample_createRefund.php
26>  ├─ap_sample_getCharge.php
27>  ├─ap_sample_getChargePermission.php
28>  ├─ap_sample_getCheckoutSession.php
29>  ├─ap_sample_getRefund.php
30>  ├─ap_sample_updateChargePermission.php
31>  ├─ap_sample_updateCheckoutSession.php
32>  ├─createCheckoutSession.php
33>  ├─result_return.php
34>  └─updateCheckoutSession.php
35> ├─vendor
36> ├─cart.html
37> ├─order_confirmation.html
38> ├─order_error.html
39> ├─review_order_page.html
40> ├─tester_cancelCharge.html
41> ├─tester_captureCharge.html
42> ├─tester_closeChargePermission.html
43> ├─tester_createCharge.html
44> ├─tester_createCheckoutSession.html
45> ├─tester_createRefund.html
46> ├─tester_getCharge.html
47> ├─tester_getChargePermission.html
48> ├─tester_getCheckoutSession.html
49> ├─tester_getRefund.html
50> ├─tester_resultPage.html
51> ├─tester_signin.html
52> ├─tester_updateChargePermission.html
53> └─tester_updateCheckoutSession.html
```

フォルダとファイルの概要

行数	フォルダ名/ファイル名	区分	概要
1	./	フォルダ	実際にサンプルファイルを配置する場所と読み替えてください。当ドキュメント内では http://localhost/sample に配置する想定といたします。
2	AmazonPayV2	フォルダ	PHP の SDK を配置するフォルダ
3	amazon-pay-v2.phar	ファイル	Amazon Pay v2 の PHP 用 SDK
4	css	フォルダ	CSS を配置するフォルダ

5	common.css	ファイル	（任意）各ページのパーツ構成に使用
6	uikit-rtl.css	ファイル	（任意）各ページのパーツ構成に使用
7	uikit-rtl.min.css	ファイル	（任意）各ページのパーツ構成に使用
8	uikit.css	ファイル	（任意）各ページのパーツ構成に使用
9	uikit.min.css	ファイル	（任意）各ページのパーツ構成に使用
10	js	フォルダ	JavaScript を配置するフォルダ
11	common.js	ファイル	（任意）共通で使う Function を記載。
12	uikit-icons.js	ファイル	（任意）各ページのパーツ構成に使用
13	uikit-icons.min.js	ファイル	（任意）各ページのパーツ構成に使用
14	uikit.js	ファイル	（任意）各ページのパーツ構成に使用
15	uikit.min.js	ファイル	（任意）各ページのパーツ構成に使用
16	keys	フォルダ	秘密キーを格納するフォルダ。実際には外部からアクセスできない場所にするか、そのようなアクセス権限に設定する必要があります。
17	private.pem	ファイル	Amazon Pay の API にアクセスするための秘密キー。貴社で

			厳重に保存し、Amazon の担当者にも公開しないでください。
18	php	フォルダ	PHP のファイルを格納するフォルダ
19	ap_sample_cancelCharge.php	ファイル	テストドライバ用：cancelCharge を呼ぶための PHP クラス
20	ap_sample_captureCharge.php	ファイル	テストドライバ用：captureCharge を呼ぶための PHP クラス
21	ap_sample_closeChargePermission.php	ファイル	テストドライバ用：closeChargePermission を呼ぶための PHP クラス
22	ap_sample_config.php	ファイル	API を呼ぶために必要な ID や環境の指定を記載したファイル
23	ap_sample_createCharge.php	ファイル	テストドライバ用：createCharge を呼ぶための PHP クラス
24	ap_sample_createCheckoutSession.php	ファイル	テストドライバ用：createCheckoutSession を呼ぶための PHP クラス
25	ap_sample_createRefund.php	ファイル	テストドライバ用：createRefund を呼ぶための PHP クラス
26	ap_sample_getCharge.php	ファイル	テストドライバ用：getCharge を呼ぶための PHP クラス

27	ap_sample_getChargePermission.php	ファイル	テストドライバ用：getChargePermission を呼ぶための PHP クラス
28	ap_sample_getCheckoutSession.php	ファイル	テストドライバ用：getCheckoutSession を呼ぶための PHP クラス
29	ap_sample_getRefund.php	ファイル	テストドライバ用：getRefund を呼ぶための PHP クラス
30	ap_sample_updateChargePermission.php	ファイル	テストドライバ用：updateChargePermission を呼ぶための PHP クラス
31	ap_sample_updateCheckoutSession.php	ファイル	テストドライバ用：updateCheckoutSession を呼ぶための PHP クラス
32	createCheckoutSession.php	ファイル	サンプルサイト用：Checkout Session 生成用クラス
33	result_return.php	ファイル	サンプルサイト用：処理結果を判定し遷移先を仕分けるためのクラス
34	updateCheckoutSession.php	ファイル	サンプルサイト用：金額等を更新して amazonPayRedirectUrl を返すためのクラス
35	vendor	フォルダ	サードバーティー製の PHP ファイルを格納するフォルダ。詳細は省略します。
36	cart.html	ファイル	サンプルサイト用：カートページを想定したページ

37	order_confirmation.html	ファイル	サンプルサイト用：購入確定後のサンクスページを想定したページ
38	order_error.html	ファイル	サンプルサイト用：購入失敗時に遷移するエラー表示用ページ
39	review_order_page.html	ファイル	サンプルサイト用：購入確定前の確認ページを想定したページ
40	tester_cancelCharge.html	ファイル	テストドライバ用：cancelCharge API を呼ぶためのドライバページ
41	tester_captureCharge.html	ファイル	テストドライバ用：captureCharge API を呼ぶためのドライバページ
42	tester_closeChargePermission.html	ファイル	テストドライバ用：closeChargePermission API を呼ぶためのドライバページ
43	tester_createCharge.html	ファイル	テストドライバ用：createCharge API を呼ぶためのドライバページ
44	tester_createCheckoutSession.html	ファイル	テストドライバ用：createCheckoutSession API を呼ぶためのドライバページ
45	tester_createRefund.html	ファイル	テストドライバ用：createRefund API を呼ぶためのドライバページ
46	tester_getCharge.html	ファイル	テストドライバ用：getCharge API を呼ぶためのドライバページ

47	tester_getChargePermission.html	ファイル	テストドライバ用：getChargePermission API を呼ぶためのドライバページ
48	tester_getCheckoutSession.html	ファイル	テストドライバ用：getCheckoutSession API を呼ぶためのドライバページ
49	tester_getRefund.html	ファイル	テストドライバ用：getRefund API を呼ぶためのドライバページ
50	tester_resultPage.html	ファイル	テストドライバ用：resultPage API を呼ぶためのドライバページ
51	tester_signin.html	ファイル	テストドライバ用：signin API を呼ぶためのドライバページ
52	tester_updateChargePermission.html	ファイル	テストドライバ用：updateChargePermission API を呼ぶためのドライバページ
53	tester_updateCheckoutSession.html	ファイル	テストドライバ用：updateCheckoutSession API を呼ぶためのドライバページ

初期設定

private.pem

(17) private.pem を貴社の秘密キーと差し替えてください。この秘密キーは PublicKeyId の取得時に使った対となるキーです。これはサンドボックス環境のテストであっても厳重に管理し、漏洩しているリスクがある秘密キーは絶対に使わないでください。詳しくは Integration Guide の「PublicKeyId の取得とキー交換」をご覧ください。

ap_sample_config.php

(22) ap_sample_config.php を開き、貴社の環境に合わせて各値を修正します。

```
<?php
  $amazonpay_config = array(
```

```
    'public_key_id' => ' XXXXXXXXXXXXXXXXXXXXXXX',
    'private_key'   => '../keys/private.pem',
    'sandbox'      => true,
    'region'       => 'jp'
);
```

#	パラメタ名	概要
1	public_key_id	public_key_id は Amazon の担当営業より提供差し上げた ID を設定します。詳しくは Integration Guide Amazon Pay Checkout v2 の「4. PublicKeyId の取得とキー交換」をご確認ください。
2	private_key	上記、キー交換時に作成した秘密キー（17）private.pem へのパスを記載します。
3	sandbox	サンドボックスモードの指定です。true の場合はサンドボックスモードになります。false の場合は本番環境モードになります。
4	region	“jp”, “us”, “eu”を指定します。Amazon JAPAN にてアカウントをお持ちの場合は”jp”と指定してください。

createCheckoutSession.php

(32) createCheckoutSession.php を開き、checkoutReviewReturnUrl の「http://localhost/sample」の部分と、storeId の「xxxx」部分を貴社の環境に合わせて修正してください。

checkoutReviewReturnUrl

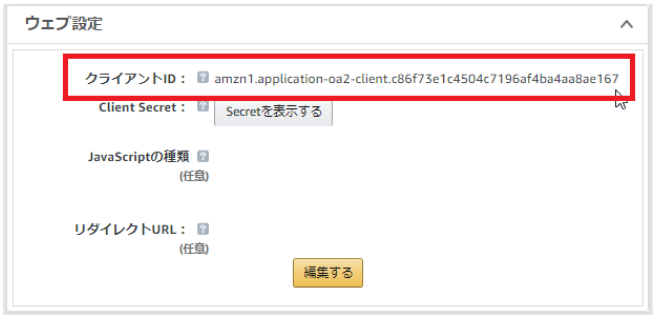
```
'checkoutReviewReturnUrl' => 'http://localhost/sample/review_order_page.html'
```

storeId

```
'storeId' => 'amzn1.application-oa2-client.xxxxxxxxxxxxxxxxxxxxxxxxxxxxxxxx'
```

storeId はセラーセントラルの「Amazon ログイン」のアプリケーション登録で設定するクライアント ID を使います。

※JavaScript の種類とリダイレクト URL は使用しません。



updateCheckoutSession.php

(34) updateCheckoutSession.php を開き、checkoutReviewReturnUrl の「http://localhost/sample」の部分を貴社の環境に合わせて修正してください。

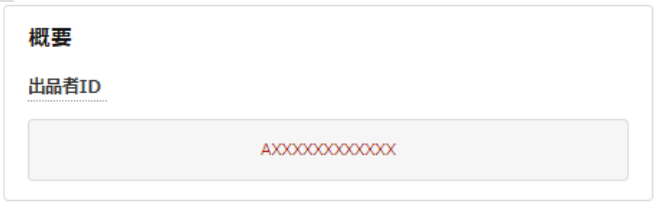
```
'checkoutResultReturnUrl' => 'http://localhost/sample/php/result_return.php'
```

cart.html

(36) cart.html を開き、merchantId の「AXXXXXXXXXXXX」を貴社の Seller ID と置き換えてください。

```
            amazon.Pay.renderButton('#AmazonPayButton', {
                merchantId: 'AXXXXXXXXXXXX',
                createCheckoutSession: function() {
                    return checkoutSessionId;
                },
```

Seller ID はセラーセントラルのメニューバーにある「インテグレーション＞MWS Access Key」にある「出品者 ID」と同義です。



テストアカウントの用意

動作を確認するためにサンドボックス環境のテスト用購入者アカウントを用意します。詳しくは Integration Guide Amazon Pay Checkout v2 の「Amazon Pay のサンドボックス環境におけるテスト用購入者アカウントの作成」をご覧ください。

疎通確認

上記の初期設定が整い PHP が稼働する状況になりましたら、http://localhost/sample/cart.html をブラウザなどで開いてください。Amazon Pay のボタンが表示されたら、それをクリックしサンドボックス環境のテスト用購入者アカウントで

サインインし支払方法などを変更せずに処理を進めてください。review_order_page.html ページに遷移した後に「購入」ボタンを押しますと、Amazon Pay のプロセスページにリダイレクトした後に order_confirmation.html ページが表示されます。うまく行かない場合は、初期設定の修正忘れや誤りがないか点検してください。合わせてブラウザのコンソールにどのようなエラーが出力されているか確認してください。

コンソールのエラーの例（秘密キーが誤っているケース）

ブラウザのコンソールに次のようなメッセージが表示されている場合、秘密キーに何らかの問題がある恐れがあります。(17) private.pem が正しいことを確認してください。

- response:
  - message: "Unable to verify signature, signing String [AMZN-PAY-RSASSA-PSS↵f86f3ac970426b25a325074b41d2a66a50c8f655bd89ec43b0e902c6fc082fbe] , signature [dhMe6rlXkQiGePtHHULN6210IBHc1fXwAW9Kdh3OvN2biUfdmz6ldiUmKQpd8iXth4aATVO6ueHLYny+WxPhKEFITi2YWWAk9rkGR5TQ4y86dsCPutH078VeN3HsuXv/AxWZb52u6ejetOenOmRriLz6eKM5s8Vgzd4w2ImACFg4nFI+nvSIBMU/Ckz4oodVVpdmLNaZ5lrT8ws1Qn2cx0w1e6HsZEW8oJGCRDaA8SNaEH1Rgcf21svpXCv6oKwl8FMaJAgd+YAjQqdaq8+66Dux9jG2k24L6bvvwwj82Tz+vcYfi3CpmBL5fgWbc/zflQk2rbFI1VaaIaX/Dx9veQ==]"
  - reasonCode: "InvalidRequestSignature"

サインインから決済までの基本的な流れ

基本的な流れ

A. フロントエンド
    1. Checkout Session の生成
    2. Button Render
    3. Checkout Session の更新（金額等のセット）と注文確認画面の表示
    4. 購入確定処理
B. バックエンド
    1. 売上請求処理

フロントエンド

1. Checkout Session の生成

Amazon Pay の購入者が決済を進める際に、その購入者の情報等を収める入れものとして Checkout Session を作る必要があります。これは 1 回の購入につき 1 つの Checkout Session が必要です。Checkout Session には購入者の情報の他にも決済するための金額や店舗名などの各種メタデータ、サインイン後に遷移するパス、決済完了後に遷移するパスなどを設定します。Checkout Session を作るためには API の CreateCheckoutSession を呼ぶ必要があります。詳しくは後述のテストドライバか API Reference Amazon Pay Checkout v2 の「Checkout Session」をご覧ください。

2. Button Render

上記の Checkout Session の ID 等を埋め込んで、Amazon Pay のサインインボタンをカートページなどに表示します。この時点でサインインした Amazon アカウントと Checkout Session が紐づきます。購入者がサインインとその後の選択を進めると、上記で設定したサインイン後に遷移するパスにリダイレクトされます。

3. Checkout Session の更新（金額等のセット）と注文確認画面の表示

送付先や配送オプションが確定して合計金額が確定したら、API の updateCheckoutSession を呼んで、その金額を Checkout Session にセットします。金額等の決済に必要な情報がすべてそろうと Checkout Session にある amazonPayRedirectUrl というカラムに決済用の一時的な URL が入ります。この URL を購入者が購入確定するタイミングで、購入者に開いてもらうようにすることでオーソリ処理を開始することができます。

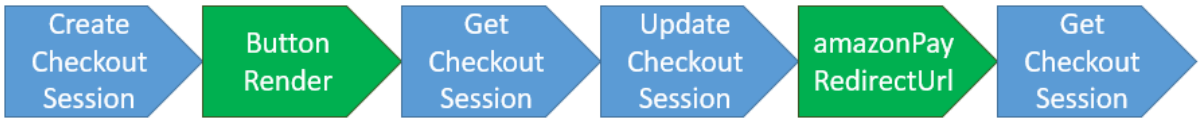
購入確定処理の前には注文商品や送付先、注文金額などが正しいことを購入者が十分に確認できるようにするために、API の getCheckoutSession を呼んで Checkout Session の設定に誤解がないように、送付先や決済金額を画面に表示します。

updateCheckoutSession と getCheckoutSession について、詳しくは後述のテストドライバか API Reference Amazon Pay Checkout v2 の「Checkout Session」をご覧ください。

4. 購入確定処理

購入者が amazonPayRedirectUrl に遷移した後、Amazon Pay 内部ではオーソリ処理を進めます。その後、決済完了後に遷移するパスにリダイレクトされます。このリダイレクト時に Get Parameter として Checkout Session ID が受け渡されますので、再び API の getCheckoutSession を呼んでオーソリの成否に応じて画面を出し分けます。

オーソリが成功した場合は受注成立として、このときに Checkout Session より取得できる、chargePermissionId と chargeId を DB などに保存します。オーソリ処理の後は、送付先などの情報が Checkout Session から取得できなくなり、Charge Permission に移ります。



バックエンド

1. 売上請求処理

商品等の発送後、売上請求をします。購入確定処理時に確保した chargeId を使って API の Capture Charge を呼びます。

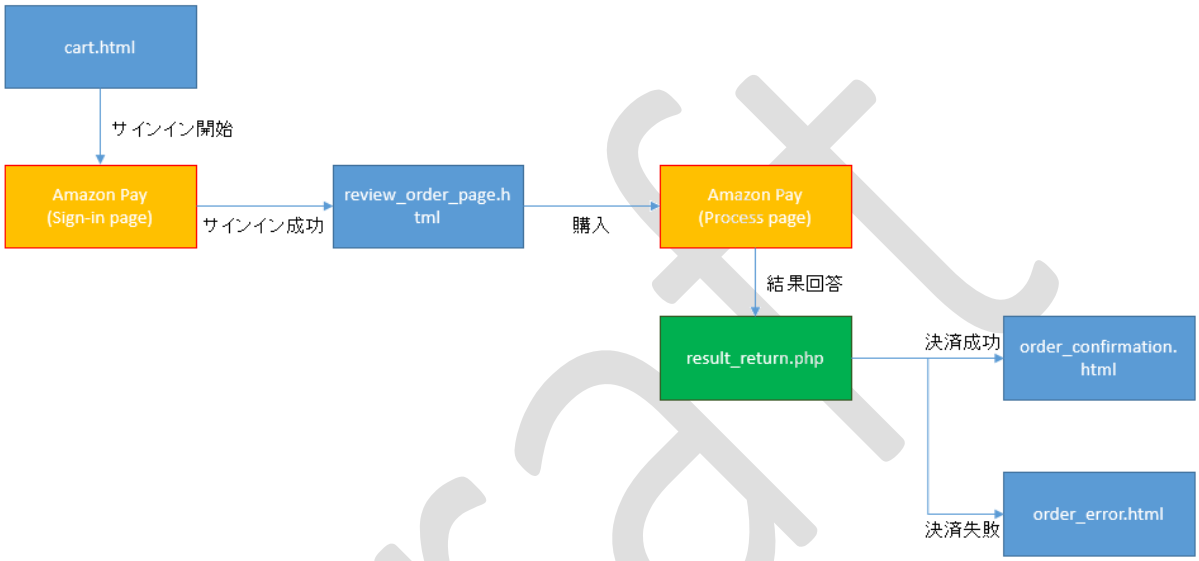
※オーソリ金額を下回る金額請求は可能ですが、一度売上請求が成功すると Charge Permission はクローズになりますのでご注意ください。

フロントエンドの仕組み

概要

疎通確認したサンプルサイトの画面フローは以下のとおりです。カートページ（cart.html）と注文確認ページ（review_order_page.html）があり、決済の成否を判定ロジック（result_return.php）が処理し、結果に応じて成功用の画面（order_confirmation.html）と失敗用の画面（order_error.html）を表示する構成です。

画面フロー



各ページの要点

カートページ

- ボタンとなる HTML コンテナ要素(DIV)を追加
- checkout.js の読み込み
- amazon.Pay.renderButton の追加

ボタンとなる HTML コンテナ要素（DIV）を追加

Amazon Pay のサインインボタンを表示する場所に下記のようなタグを追記します。ボタンサイズの調整については Integration Guide Amazon Pay Checkout v2 の「3. ボタンの表示」をご覧ください。

```
<div style="margin: 0 auto;" id="AmazonPayButton"></div>
```

checkout.js の読み込み

Amazon Pay の JavaScript を読み込むために checkout.js を定義しています。必ず下記の記述方法で実装する必要があり、JavaScript をローカルにダウンロードして使うことはできませんのでご注意ください。

```
<script src="https://static-fe.payments-amazon.com/checkout.js"></script>
```

amazon.Pay.renderButton の追加

上記の HTML コンテナ要素（DIV）にボタンを表示し、それがクリックされた際にサインインを開始できるようにしています。merchantId には貴社の Seller ID を指定します。各パラメタの詳細は Integration Guide Amazon Pay Checkout v2 の「3. ボタンの表示」の「関数のパラメタ」をご覧ください。

■Cart.html 抜粋（ボタンの生成）

```
            amazon.Pay.renderButton('#AmazonPayButton', {
                merchantId: 'AXXXXXXXXXXXX',
                createCheckoutSession: ‘bb6b830b-f41d-4cf6-b03c-4c5c09feb626’,
                ledgerCurrency: 'JPY',
                checkoutLanguage: 'ja_JP',
                productType: 'PayAndShip',
                placement: 'Cart'
            });
```

サンプルでは createCheckoutSession を AJAX で(32) createChekoutSession.php を使って取得しています。下記は createChekoutSession.php コードです。サインイン後のページを指定する checkoutReviewReturnUrl を「review_order_page.html」としています。決済処理後のページを指定する checkoutResultReturnUrl はコメントアウトしています。これは後述の update の際に指定していますが、このようにこの時点では設定しないこともできます。x-amz-pay-Idempotency-Key（冪等性キー）には便宜上、乱数を入れていますが、エラー時のリトライ用に保持することが望ましいです。例えば、途中まで処理が進んで回線エラーなどの不可避のエラーが発生したことで処理結果が分からなくなった場合、x-amz-pay-Idempotency-Key（冪等性キー）に同じ値を入れることで処理を重複させることなく続けることができます。

■createChekoutSession.php

```
<?php
    require_once '../vendor/autoload.php';
    require_once '../AmazonPayV2/amazon-pay-v2.phar';
    require_once 'ap_sample_config.php';

    $payload = array(
        'webCheckoutDetails' => array(
            'checkoutReviewReturnUrl' =>
'http://localhost/sample/review_order_page.html' //,
            // 'checkoutResultReturnUrl' =>
'http://localhost/sample/order_confirmation.html'
        ),
        'storeId' => 'amzn1.application-oa2-client.xxxx'
    );

    $headers = array('x-amz-pay-Idempotency-Key' => uniqid());
```

```
    try {
        $client = new AmazonPayV2\Client($amazonpay_config);
        $result = $client->createCheckoutSession($payload, $headers);
        $response = '{"status":' . $result['status'] . ',"response":' .
$result['response'] . '}';
        echo($response);

    } catch (\Exception $e) {
        // handle the exception
        echo $e . "\n";
    }
```

このサンプルでは AJAX で Checkout Session ID を取得していますが、ページを描画する際にあらかじめ取得しておいたものを使うこともできます。ただし、Checkout Session は有効期限が 24 時間なのでご注意ください。

Amazon Pay サインインページと送付先・支払方法選択ページ

カートページのボタンを押すと、Amazon Pay のサインインページが表示されます。過去のサインイン状況によってはサインインページがスキップされることがあります。Checkout Session をサンドボックスモードで生成していた場合（(22) ap_sample_config.php の sandbox パラメタを true にしていた場合）、このページのヘッダーにサンドボックス環境であることを示すアイコンが表示されます。

注文確認ページ

- 購入者の選択状態を取得して表示する
- 送付先を再修正できるようにする
- Checkout Session を更新して amazonPayRedirectUrl にリダイレクトする

購入者の選択状態を取得して表示する

サインインと送付先・支払方法の選択が済み、checkoutReviewReturnUrl にリダイレクトされると、Get Parameter に amazonCheckoutSessionId が入ってきます。これを使ってページのロード時に(28) ap_sample_getCheckoutSession.php を呼んで、その結果（氏名や送付先住所等）を画面に転記しています。購入者が誤解しないように、送付先情報や合計金額を明記する必要があります。

■ review_order_page.html 抜粋（ロード時のスクリプト）

```
    var checkout_session_id = getUrlParameter('amazonCheckoutSessionId');
```

```
    $.ajax({
        type: 'POST',
        url: './php/ap_sample_getCheckoutSession.php',
        data: {
            checkout_session_id: checkout_session_id
        },
    })
    .then(
        function(result) { //success
            var json = $.parseJSON(result).response;
            console.log(result);

            $("#buyerName").html(json.buyer.name);
            $("#buyerEmail").html(json.buyer.email);
            $("#addressName").html(json.shippingAddress.name);
            $("#addressLine1").html(json.shippingAddress.addressLine1);
            $("#addressLine2").html(json.shippingAddress.addressLine2);
            $("#addressLine3").html(json.shippingAddress.addressLine3);

$("#addressStageOfRegion").html(json.shippingAddress.stateOrRegion);
            $("#addressZipcode").html(json.shippingAddress.postalCode);
        },
        function() { //failure
            console.log("error");
        }
    );
```

送付先を再修正できるようにする

送付先を変更したい場合に備えて、再度送付先を変更する導線を作ります。その導線を呼ぶボタンとして HTML コンテナ要素（DIV）を用意し、そのボタンに対して次のようなコードを作り、ページのロード時などに関連付けを行います。ただし、この機能はカートページに配置した Amazon Pay のボタンと同様のものを配置することでも同じことができます。ページの構成によっては当変更ボタンが必要なく、Amazon Pay のページに戻させることなどでも足りることがあります。

■ 変更ボタンの HTML コンテナ要素（DIV）

```
<button id="updateCheckoutDetails" type="button">変更</button>
```

■ 変更ボタンと変更処理の関連付け

```
    amazon.Pay.bindChangeAction('#updateCheckoutDetails', {
        amazonCheckoutSessionId: checkout_session_id,
        changeAction: 'changeAddress'
    });
```

Checkout Session を更新して amazonPayRedirectUrl にリダイレクトする

購入者が送付先や金額を確定させて、最終確定のボタン（サンプル上では「購入」ボタン）を押下した際に、Checkout Session を更新して、更新結果として得られる amazonPayRedirectUrl に window.location.href でリダイレクトしています。

■ review_order_page.html の抜粋（購入ボタン押下時のイベント）

```
    $(document).on('click','#placeorder', function() {
        $.ajax({
            type: 'POST',
            url: './php/updateCheckoutSession.php',
            data: {
                checkout_session_id: checkout_session_id
            },
        })
        .then(
            function(result) { //success
                var json = $.parseJSON(result).response;
                console.log(result);

                if(json.webCheckoutDetails.amazonPayRedirectUrl != null) {
                    window.location.href =
json.webCheckoutDetails.amazonPayRedirectUrl;
                }
            },
            function() { //failure
                console.log("error");
            }
        );
    });
```

更新は API の updateCheckoutSession を呼ぶことで行っています。金額は画面から引き継がず、フロントからの改ざんを防ぐために PHP のプログラム内で行っています。当サンプルでは便宜上の都合で金額を固定値としていますが、実際にはカートの内容もプログラム内で処理します。また、ここでは Button Render の際にスキップした checkoutResultReturnUrl の指定も行っています。(33) result_return.php に仕向けて、そこでは getCheckoutSession でオーソリの結果（statusDetails の state）を取得して、成功の場合は(37) order_confirmation.html に、失敗の場合は(38) order_error.html にリダイレクトさせています。

■ updateCheckoutSession.php

```
<?php
    require_once '../vendor/autoload.php';
    require_once '../AmazonPayV2/amazon-pay-v2.phar';
    require_once 'ap_sample_config.php';
```

```
    $checkout_session_id = $_POST['checkout_session_id'];

    $payload = array(
        'webCheckoutDetails' => array(
            'checkoutResultReturnUrl' =>
'http://localhost/sample/php/result_return.php'
        ),
        'paymentDetails' => array(
            'paymentIntent' => 'Authorize',
            'canHandlePendingAuthorization' => false,
            'chargeAmount' => array(
                'amount' => '100',
                'currencyCode' => 'JPY'
            ),
        ),
        'merchantMetadata' => array(
            'merchantReferenceId' => '2020-00000001',
            'merchantStoreName' => 'Store Name',
            'noteToBuyer' => 'Thank you for your order!'
            // "customInformation" => "customInformation"
        )
    );

    try {
        $client = new AmazonPayV2\Client($amazonpay_config);
        $result = $client->updateCheckoutSession($checkout_session_id,
$payload);
        $response = '{"status":' . $result['status'] . ',"response":' .
$result['response'] . '}';
        echo($response);

    } catch (\Exception $e) {
        // handle the exception
        echo $e . "\n";
    }
```

■ result_return.php

```
<?php
    require_once '../vendor/autoload.php';
    require_once '../AmazonPayV2/amazon-pay-v2.phar';
    require_once 'ap_sample_config.php';

    $checkout_session_id = $_GET['amazonCheckoutSessionId'];

    try {
        $client = new AmazonPayV2\Client($amazonpay_config);
        $result = $client->getCheckoutSession($checkout_session_id);
```

```
        $json_result = json_decode($result['response']);

        $state = $json_result->statusDetails->state;

        if ('Completed' == $state) {
            header('location: ../order_confirmation.html?chargeId=' .
$json_result->chargeId);
            exit();
        } else {
            header('location: ../order_error.html');
            exit();
        }

    } catch (\Exception $e) {
        // handle the exception
        echo $e . "\n";
    }
```

上記の getCheckoutSession で得られる chargePermissionId と chargeId を受注データと合わせて DB 等に保存しておく必要があります。上記サンプルでは便宜上リダイレクト時の Get Parameter に ChargeId を付けていますが、実際のサイトに実装する際は不要です。

getCheckoutSession でオーソリの結果（statusDetails の state）をハンドリングする部分では、状態と理由コードで細かく制御することも可能です。詳しくは API Reference Amazon Pay Checkout v2「Checkout Session」の「状態と理由コード」をご覧ください。

バックエンドの仕組み

概要

バックエンドは通常 OMS（order management system）に組み込みますが、当サンプルではテストドライバで動きをみられるようにしています。決済確定後の基本的な流れは、chargeId をキーにして API の CaptureCharge を呼ぶことで行います。

売上請求処理

動きを確認するために http://localhost/ sample/tester_captureCharge.html を開いてください。下図の chargeId 項目に保持した Charge ID を入力して「captureCharge 実行」ボタンを押しますと、API の CaptureCharge を呼んで、その結果を以降の画面に表示します。この処理の結果で、画面最下部の statusDetail.state が Captured と表示されたら売上請求が成功です。

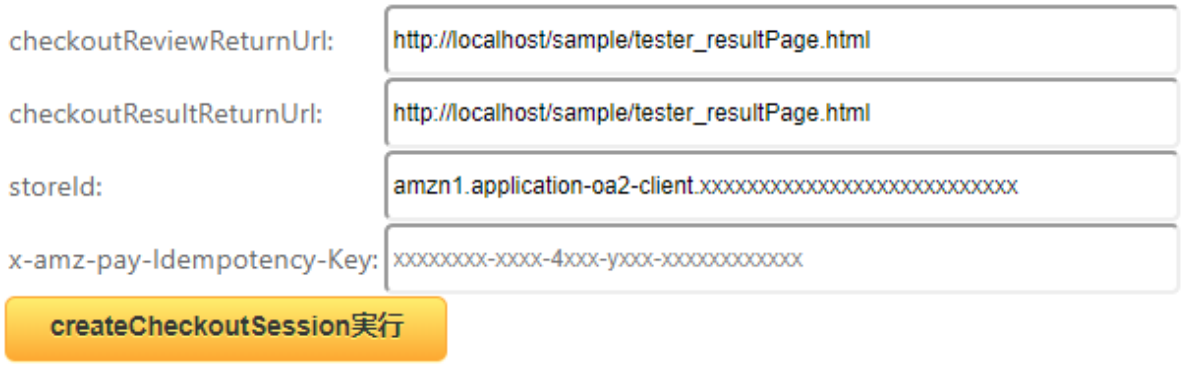
■ tester_captureCharge.html 画面の入力項目

テストドライバについて

前項の(41) tester_captureCharge.html もテストドライバの一つですが、当サンプルではすべての API に対になるドライバページを用意しています。(40)～(53)
それぞれの API の動きをドライバページで見ることができます。

はじめに、http://localhost/ sample/ tester_createCheckoutSession.html を開いてください。

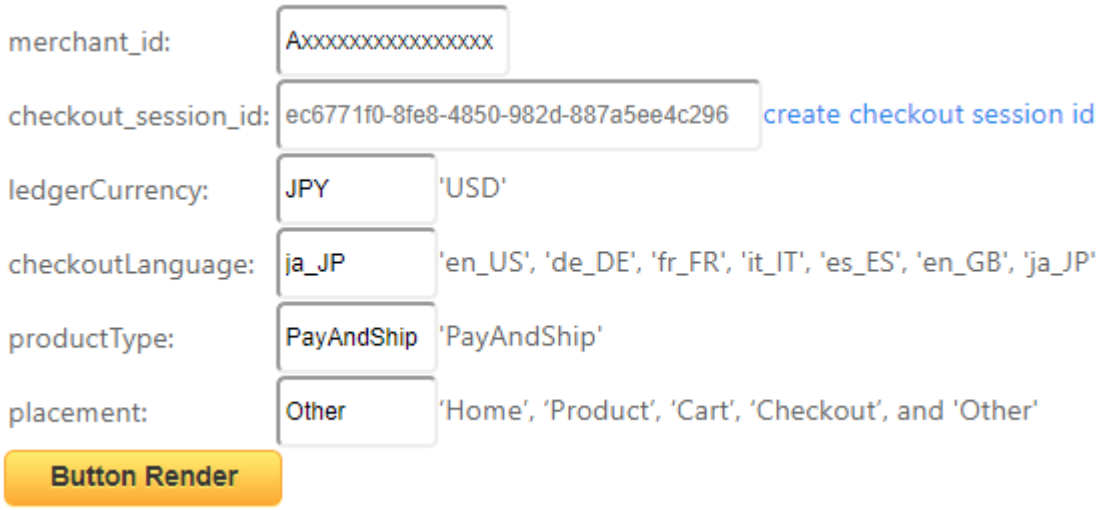
■ tester_ createCheckoutSession.html 画面の入力項目



storeId を貴社のクライアント ID で書き換え、checkoutReviewReturnUrl と checkoutResultReturnUrl はそのままで「createCheckoutSession 実行」ボタンを押下し、ボタン下の表の 6 段目の「checkoutSessionId」に表示される値をコピーします。

つぎに、http://localhost/sample/tester_ signin.html を開いてください。

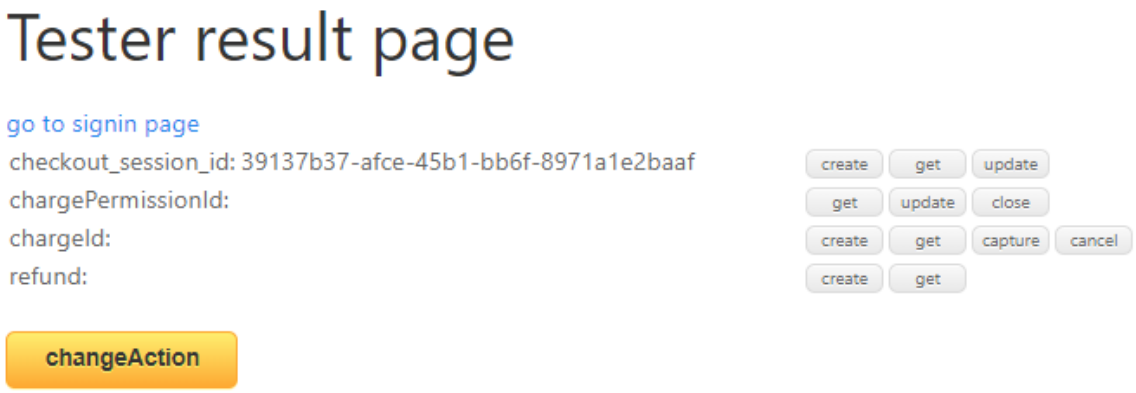
■ tester_ signin.html 画面の入力項目



merchant_id を貴社の ID に入れ替えて、先ほどコピーした CheckoutSessionId を checkout_session_id にペーストします。「Button Render」ボタンを押下すると、サインイ

ンボタンが出現しますので、サインインしテストしたいシミュレーションカードや送付先住所を選択してリダイレクトさせます。

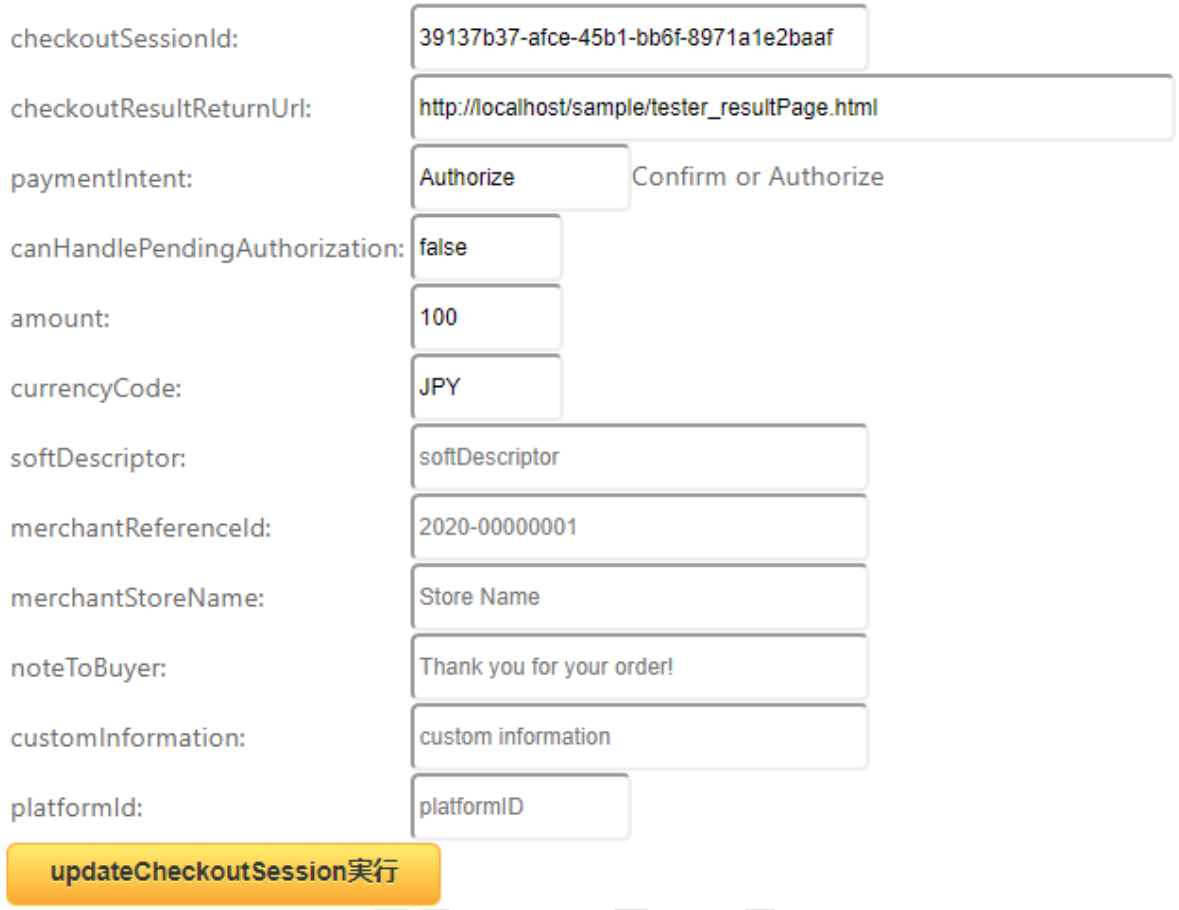
■ tester_ resultPage.html 画面の項目



この時点では Checkout Session に必須項目が揃っていないため、ChargePermissionId と ChargeId は取得できません。

次に、checkout_session_id の行にある「update」ボタンを押して tester_ updateCheckoutSession.html を開いてください。

■ tester_ updateCheckoutSession.html 画面の項目



金額等を設定して CheckoutSession を更新することができます。この更新で必要項目が揃うと画面下部の webCheckoutDetails.amazonPayRedirectUrl の行に次のような URL が表示されるようになります。

```
https://apay-
us.amazon.com/checkout/processing?amazonCheckoutSessionId=39137b37-afce-
45b1-bb6f-8971a1e2baaf
```

上記の URL をブラウザなどで開くと、updateCheckoutSession の設定に応じて、オーソリ等の処理が進みます。この処理の後は ChargePermissionId と ChargeId が取得できるようになりますので、tester_ resultPage.html から同じ要領で各種 API を操作してください。

ドライバページの URL は下記のとおりです。

ドライバページ URL

1. http://localhost/sample/tester_signin.html
2. http://localhost/sample/tester_cancelCharge.html
3. http://localhost/sample/tester_captureCharge.html
4. http://localhost/sample/tester_closeChargePermission.html
5. http://localhost/sample/tester_createCharge.html
6. http://localhost/sample/tester_createCheckoutSession.html

7. http://localhost/sample/tester_createRefund.html
8. http://localhost/sample/tester_getCharge.html
9. http://localhost/sample/tester_getChargePermission.html
10. http://localhost/sample/tester_getCheckoutSession.html
11. http://localhost/sample/tester_getRefund.html
12. http://localhost/sample/tester_resultPage.html
13. http://localhost/sample/tester_updateChargePermission.html
14. http://localhost/sample/tester_updateCheckoutSession.html

特殊処理

オーソリ期限切れ対応

オーソリを確保しても 30 日後には期限切れになります。この場合、改めてオーソリを取得する必要があります。このケースでは API の createChage を呼んでください。createChage は updateCheckoutSession で指定した金額以下で再オーソリをかけることができます。この時、Charge Permission は Chargeable になっている必要があります。Open 状態の chageId がある場合（NonChargeable）や chargePermission 自体が Closed の場合は処理できません。また、createChage で取得された新しい chargeId は必ず DB などで保存してください。この ID はこの機会で取得しない限り、Checkout Session や Charge Permission を API で GET しても確認できませんのでご注意ください。

オーソリキャンセル

オーソリのキャンセルは API の closeCharge を呼ぶことで行うことができます。この処理を行うと Charge Permission は Chargeable に戻ります。

返金処理

返金処理（Refund）は chargeId をキーにして API の createRefund を呼ぶことで行うことができます。また、createRefund で取得された新しい refundId は必ず DB などで保存してください。この ID はこの機会で取得しない限り、Checkout Session や Charge Permission を API で GET しても確認できませんのでご注意ください。