Amazon Pay
インテグレーションガイド

Document Version 2.1.0 JP

Amazon Pay
Integration Guide

目次

概要 8
イントロダクション 8
どのように Amazon Pay は動作するのでしょうか 8
モバイルデバイス上の Amazon Pay の購入処理 9
トランザクションワークフロー 10
インテグレーションステップ 12
Step 1:登録 12
Step 2:SDK の設定 12
処理内容 13
Step 3:ボタンウィジェットの追加 14
処理内容 14
コードサンプル 17
Step 4:アドレス帳とお支払い方法ウィジェットの追加 18
処理内容 18
サンプルコード 20
アドレス帳ウィジェット 21
お支払い方法ウィジェット 22
Step 5:注文詳細のセットと確認 23
処理内容 23
サンプルコード 24
Step 6:オーソリ（Authorize）のリクエスト 27
非同期と同期のオーソリ(Authorize)API 呼び出し 28
処理内容 28
サンプルコード 30
Step 7:売上請求（Capture）のリクエスト 34
処理内容 35
サンプルコード 35
Step 8:Order Reference を closed に変更 37
処理内容 38
Step 9:返金のリクエスト 39

処理内容 39
インテグレーションステップのオプション 42
ログイン時での注文情報へのアクセス 42
リダイレクト認証を有効にする。 43
処理内容 43
クロスサイトリクエストフォージェリ 44
購入者が同意しない場合 44
プロファイル情報の取得 45
アドレス帳とお支払い方法参照ウィジェットの表示 49
アドレス帳ウィジェット無しのインテグレーション 50
トランザクションフローの管理 51
お支払い方法ウィジェットの再レンダリング 52
処理内容 52
売上請求（Capture）とオーソリ（Authorize）での即時請求 53
Amazon Pay でのゲスト購入 53
オーソリ 30 日後の売上請求（Capture） 54
決済とトランザクションレポート 54
追加トランザクション 56
オーソリ失敗のハンドリング 56
同期モード 56
非同期モード 58
Soft Decline ハンドリングの代表的なフロー 59
複雑な支払シナリオのハンドリング 59
支払を分割 59
延期された支払 60
即時請求 61
注文完了後の支払変更のハンドリング 62
最初の注文金額以上の請求 62
配送先住所の変更 62
注文のキャンセル 63
部分的な注文の実行 63
SANDBOX 環境でのインテグレーションテスト 65

Amazon Pay SANDBOX のテストアカウント設定 65
サンプルの住所と支払方法 66
サンプルの配送先住所 67
サンプルの支払方法 67
SANDBOX と本番環境の相違点 67
SANDBOX シミュレーション 68
本番環境へインテグレーション切り替え 74
コードリファレンス情報 75
エラーハンドリング 75
エラーハンドリングの概要 75
Amazon Pay API 呼び出しからのエラーハンドリング 75
Amazon Pay ウィジェットのエラーハンドリング 76
JavaScript 80
処理開始前にデータをエンコーディング 80
Web ページに callback と widgets.js コードを追加 81
非同期 JavaScript 81
インスタント支払通知（IPN） 82
インスタント支払通知メッセージのハンドリング 82
カスタムデータフィールド 83
サンプル通知 84
IPN メッセージの応答 90
IPN ハンドリングのベストプラクティス 91
API インフォメーション 91
API の概要 91
アクセストークン 91
提供可能な購入者メールコンテンツ 92
ウィジェット 101
ボタンウィジェット 101
Amazon Pay ウィジェット 109
TLS/SSL 証明書 113
TLS/SSL イントロダクション 113
Amazon が要求する TLS/SSL 証明書 115

付録 117
インテグレーション概要 117
Amazon Pay を購入で利用 117
支払処理 120
スタイルガイドライン 121
サイトに Amazon アカウントでお支払いボタンを追加 121
その他のサインインインテグレーションの提供 122
顧客アカウントを Amazon とリンク 123
サイトに Amazon アカウントでお支払いボタンを追加 123
商品詳細ページに Amazon Pay を追加 124
ゲスト購入後に顧客にログインを尋ねる 125
用語集 127

変更履歴

バージョン	変更日	変更内容
2.0.0 JP	2016/09/18	新インテグレーションガイドとして 2.0.0 を新規作成
2.0.1 JP	2016/09/29	誤字修正、および、英語表記の統一
2.0.2 JP	2016/10/13	Widgets.js URL の誤字修正
2.0.3 JP	2016/10/20	誤字修正
2.0.4 JP	2016/12/09	P14 の PHP サンプルプログラムを修正
2.0.5 JP	2016/12/16	P105 の profile:postal_code の表記ミスを修正し、postal_code に修正
2.0.6 JP	2017/01/19	P22 のサンプルコードのコメントタグの閉じる部分の表記ミスを修正 P110 のサンプルコードのコメントタグの閉じる部分の表記漏れを修正
2.0.7 JP	2017/01/25	P95 のメール送信条件の変更に対応
2.1.0 JP	2017/02/22	「Amazon ログイン&ペイメント」、および、「Amazon ペイメント」の名称変更により「Amazon Pay」に修正

概要

当資料は Amazon Pay US 環境の情報を翻訳したものです。仕様上は日本環境も同じですが、画像、および、サンプルコードの一部は US 環境のままで記載しております。
適宜 Amazon Pay 日本環境に読み替えて頂く必要があります。

2017 年 2 月 22 日から「Amazon Pay」にサービス名称を変更しました。当資料に記載されている旧名称については読み替えて下さい。

イントロダクション

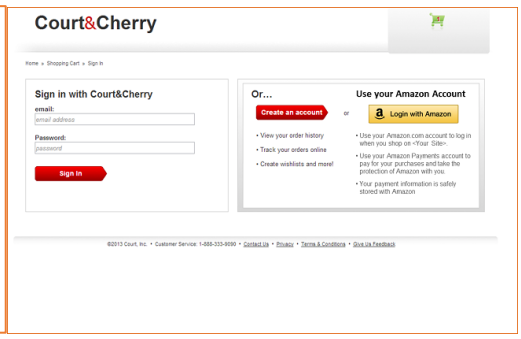
Amazon Pay は、Amazon のお客様が、セキュアで安心して便利にお買い物ができるログインと支払方法を販売事業者のサイトに提供します。購入者エクスペリエンスでは、購入処理を簡単に安全に行えるようにデザインされており、販売事業者様の支払処理は簡単に安全になっております。

注意：Amazon Pay サービスは Web ブラウザ上でのみ利用することでデザイン、および、開発されています。Amazon Pay のサービスはネイティブアプリケーション（iOS、Android、RIM、Windows オペレーティングシステムを含みますがこれに限りません）で利用することができません。

どのように Amazon Pay は動作するのでしょうか

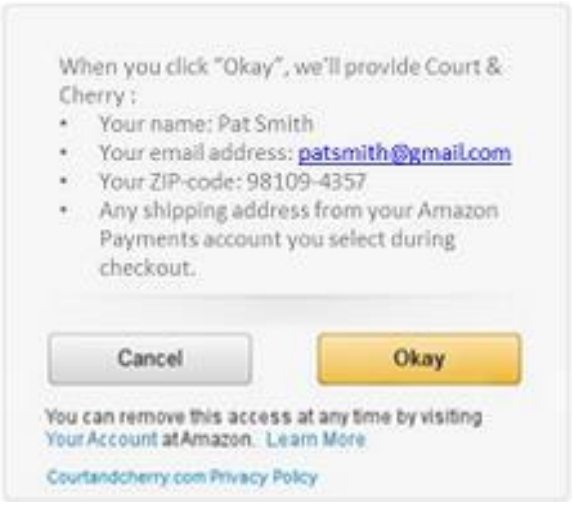
Amazon Pay は既に存在する Web サイトへ直接埋め込まれます。すべての購入者は Amazon Pay 用に埋め込まれたウィジェットを利用します。よって、購入者はサイトから離れることはありません。

購入者は Web サイト上の Amazon ログイン、または、Amazon Pay ボタンを参照します。

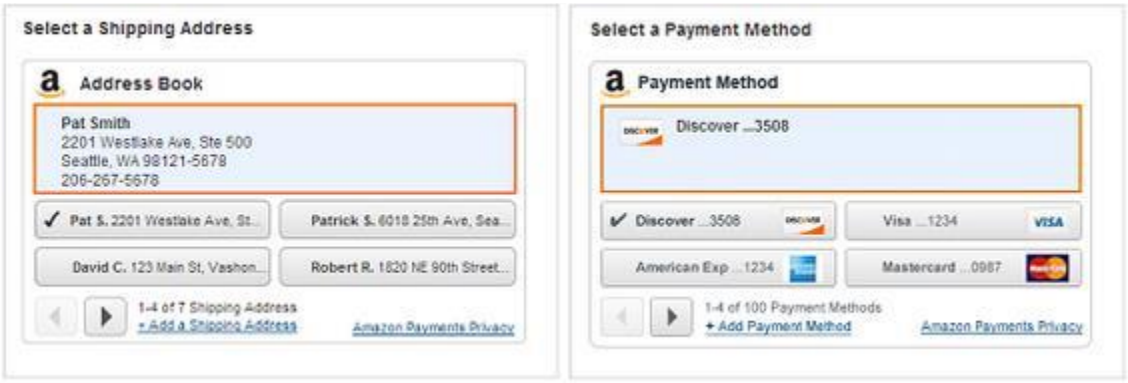


Amazon による購入者認証

購入者が初めて Amazon アカウントでログインする場合は、個人情報を共有することについて同意しなければなりません。それから、Web サイト上でお客さま情報やアカウントを生成するために個人情報を利用できます。



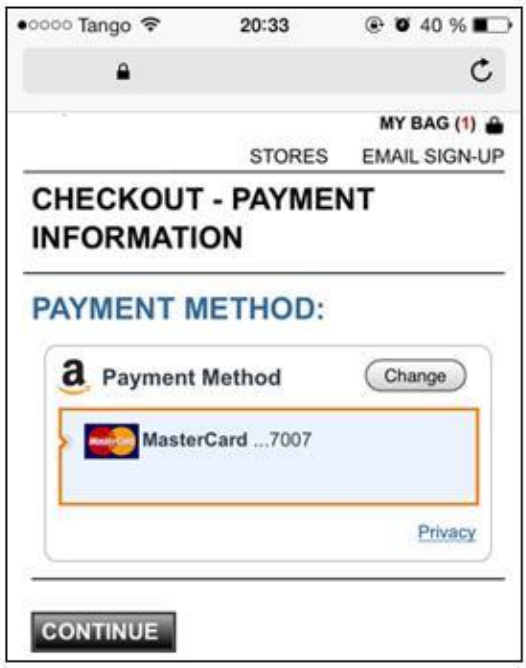
購入者は配送先住所と支払方法を選択します。



購入内容を確認し、購入者にサンクス（完了）ページを表示します。

モバイルデバイス上の Amazon Pay の購入処理

Amazon Pay のウィジェットは、完全にレスポンシブであり、モバイルデバイス用に最適化されております。



トランザクションワークフロー

次は物品販売における標準的な注文ワークフローです。

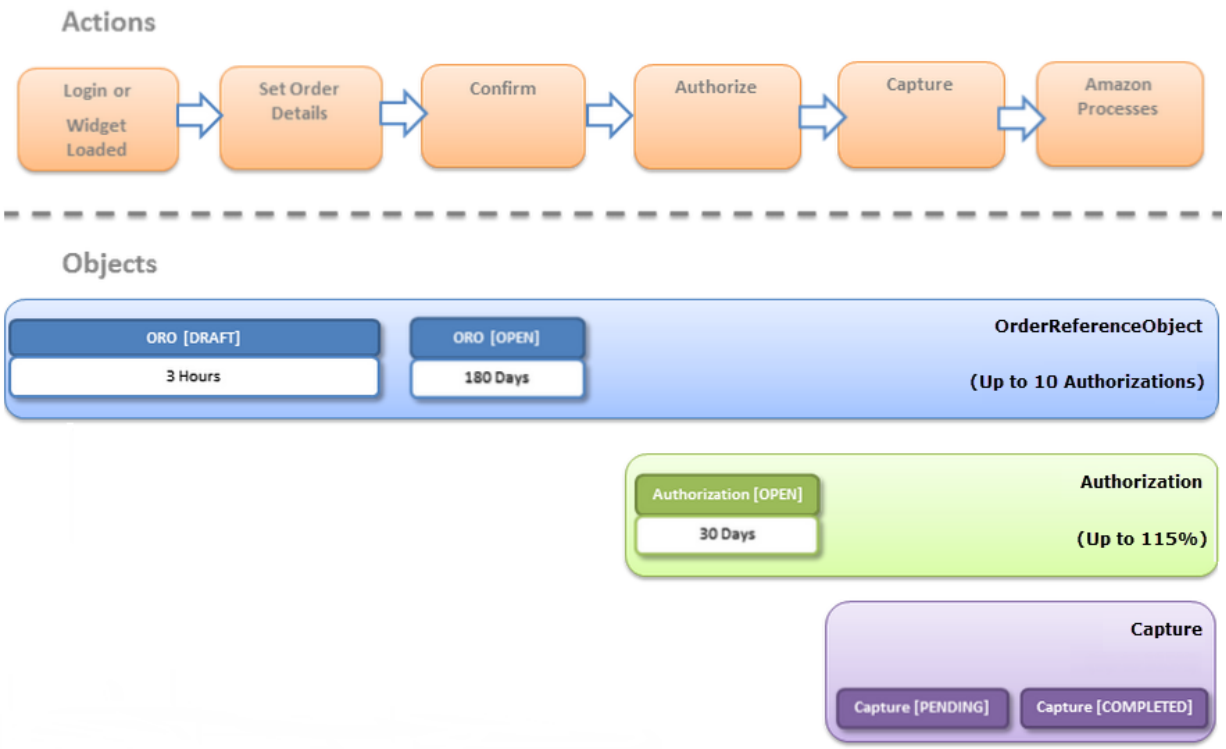
1. 購入者の **Order Reference** を生成するためのアドレス帳ウィジェットを表示します。
2. 購入者はアドレス帳ウィジェットから住所を、お支払い方法ウィジェットから支払方法を選択します。この時点で注文合計金額を計算します。
3. 購入者が注文を完了し、システムは注文詳細をセットし、**Order Reference** を承認（**Confirm**）します。
4. システムは選択された支払方法に対して注文金額のオーソリ（**Authorize**）のリクエストを発行します。
5. オーソリ（**Authorization**）が成功した場合は、注文は出荷できます。
6. 販売事業者は注文を出荷します。
7. システムはオーソリ（**Authorize**）された金額から売上請求（**Capture**）のリクエストを発行し、それから成功した売上請求（**Capture**）の後で **Order Reference** を終了します。

トランザクション中では、Amazon Pay オブジェクトは処理をサポートするために生成されます。

Amazon Pay オブジェクト	説明
Order Reference	購入者の支払を処理するために必要な主要な属性レコードです。購入者が支払いたい方法、どこへ配送したいのかを参照することができます。また、合計金額、注文 ID などの情報も含みます。

Authorization（オーソリ）	選択された支払方法の有効な金額を表し、売上請求（**Capture**）リクエストを利用して集金するために留保しています。
Capture（売上請求）	購入者の支払方法から Amazon Pay の販売事業者アカウントへ移動したことを表します。
Refund（返金）	返金する場合に利用され、直前の売上請求（**Capture**）された金額を購入者アカウントに戻したことを表します。

それぞれのオブジェクトは、トランザクションフロー中の注文についての状態を表すステータスを持っています。次の図はフロー内の様々なアクションと関連するオブジェクトを表します。

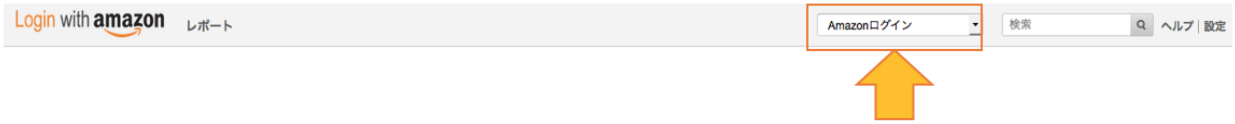


インテグレーションステップ

Step 1:登録

インテグレーションを開始するために次のステップを実行する必要があります。

1. 販売事業者は Amazon Pay 専用のセラーセントラルのアカウントを作成する必要があります。
   Amazon Pay アカウントは、Amazon に販売事業者の支払情報、および、レポートするためのアプリケーション情報を提供し、サインインボタンとウィジェットリクエストのための販売事業者 ID（セラーID）を提供します。
2. 販売事業者アプリケーションと Amazon ログインを関連付けします。
   セラーセントラルにログインし、画面右上のドロップダウンリストより **Amazon ログイン**を選択して新しいアプリケーションを登録します。それから「**保存する**」ボタンをクリックします。



   上記の通りに Amazon ログインを選択します。一度登録すると、Amazon ログインのクライアント ID とクライアントシークレット ID を取得できます。
3. Amazon Pay SANDBOX のテストアカウントをセットします。
   本番環境に移行する前にインテグレーションテストを行うために SANDBOX を利用します。詳しくは Amazon Pay SANDBOX テストアカウント設定を参照してください。
4. TLS/SSL 認証を取得します。
   サーバは信頼される有効な TLS/SSL 証明書を取得している必要があります。localhost 環境で動作させる場合は、TLS/SSL 証明書は必要ありません。詳しくは TLS/SSL 証明書を参照してください。
5. 認証情報を集めます。
   セラーセントラルの Amazon Pay 設定ページ（インテグレーション→MWS Aceess Key）上から次の認証情報を見つけることができます。

   出品者 ID
   MWS アカウント情報のアクセスキーID とシークレットアクセスキー
   Amazon ログインアカウント情報のクライアント ID

Step 2:SDK の設定

コンフィグファイルと Amazon MWS エンドポイントを設定するように、このセクションは SDK セットアップ要件をカバーします。

処理内容

Amazon Pay を Web サイトに設定するためには次のステップを完了します。

1. セラーセントラルの Amazon ログインより、JavaScript の種類とリダイレクト URL を設定します。
2. SANDBOX MWS エンドポイントとその他の URL をセットします。

推奨のベストプラクティスは、全ての API 呼び出しは本番環境に移管される前にテストする目的で SANDBOX にて実行されるべきです。

以下の表は、SANDBOX を本番環境のエンドポイントの表です。

Amazon MWS エンドポイント	
SANDBOX	https://mws.amazonservices.jp/OffAmazonPayments_Sandbox/2013-01-01/
本番環境	https://mws.amazonservices.jp/OffAmazonPayments/2013-01-01/
Widgets.js URL	
SANDBOX	https://static-fe.payments-amazon.com/OffAmazonPayments/jp/sandbox/lpa/js/Widgets.js
本番環境	https://static-fe.payments-amazon.com/OffAmazonPayments/jp/lpa/js/Widgets.js
Profile API エンドポイント	
SANDBOX	https://api-sandbox.amazon.co.jp/user/profile
本番環境	https://api.amazon.co.jp/user/profile

PHP の参考例は次の通りです。

PHP の必須条件は以下の通りです。

- PHP 5.3 またはそれ以上
- Curl 7.18 またはそれ以上

セットアップの詳しい情報は、Amazon Pay PHP SDK（https://github.com/amzn/login-and-pay-with-amazon-sdk-php）を参照してください。

これは Client オブジェクトのインスタンスを生成する時に設定をセットする方法のサンプルです。SANDBOX パラメータは指定しなければデフォルトで **false** になるので注意してください。

```
$config = array('merchant_id'    => 'YOUR_MERCHANT_ID',
```

```
                'access_key'    => 'YOUR_ACCESS_KEY',
                'secret_key'    => 'YOUR_SECRET_KEY',
                'client_id'    => 'YOUR_LOGIN_WITH_AMAZON_CLIENT_ID',
                'region'    => 'REGION',
                'sandbox'    => true );

$client = new Client($config);

// Also you can set the sandbox variable in the config() array of the Client
class by

$client->setSandbox(true);
```

Step 3:ボタンウィジェットの追加

Amazon Pay は **Amazon ログイン**と **Amazon Pay** サービスで構成されており、技術的に関連しており同じプラットフォームで動作します。

- Amazon ログインは、購入者が Web サイトを参照した時に Amazon ログイン認証を利用することを推奨します。
- Amazon Pay は、購入者が Amazon のユーザーアカウントに登録されている配送先住所と支払方法を選択することを許可する購入オプションです。

購入者が Amazon ログインに成功した場合は、**access token** が返されます。この **access token** は、クライアント、購入者、および、個人情報の種類を要求するための識別子です。顧客情報を取得するために **access token** は必要になります。

処理内容

注意：処理する前にデータをエンコードしなければなりません。Amazon から取得される HTML、JavaScript、URL 文字列のようないくつかの要素を出力する前にデータをエンコードすることを推奨します。Web サイト上で、悪意のあるスクリプト、その他の取り込まれた実行可能なファイルを実行できないように出力エンコードを確実にします。追加の詳しい情報は、JavaScript の「処理開始前にデータをエンコーディング」を参照してください。

次のステップは購入者認証のためのボタンウィジェットを追加する方法を説明します。注文処理でこのステップを実行します。特に、Step1 を完了する前に widgets.js スクリプト（Step 2）を追加してはいけません。コードサンプルは、このトピックの最後で説明します。

1. ボタンを表示させたい全てのページの<head>セクションに onAmazonLoginReady と onAmazonPaymentsReady のコールバックを追加します。この onAmazonPaymentsReady コールバックはボタンを表示します。

```
<head>
  <script type='text/javascript'>
    window.onAmazonLoginReady = function() {
      amazon.Login.setClientId('CLIENT-ID');
    };
    window.onAmazonPaymentsReady = function() {
                showButton();
    };
  </script>
</head>
```

詳しい情報は、JavaScript の「Web ページに Callback と widgets.js コードの追加」を参照してください。

2. コールバック関数に widget.js ファイルを追加します。

```
<head>
  <script type='text/javascript'>
    window.onAmazonLoginReady = function() {
      amazon.Login.setClientId('CLIENT-ID');
    };
    window.onAmazonPaymentsReady = function() {
                showButton();
    };
  </script>

  <script async="async" src='https://static-na.payments-
amazon.com/OffAmazonPayments/us/sandbox/js/Widgets.js'>
  </script>
</head>
```

このサンプルのスクリプトタグは、バックグラウンドで Amazon Pay を速やかに JavaScript コードをロードするために許可された非同期属性を含みます。これは Amazon Pay で唯一サポートしている非同期方法です。

3. Web ページの body にボタンコードを追加します。ボタンを表示したい全てのページの <body>に JavaScript を追加します。

```
<div id="AmazonPayButton">
 </div>
  ...
 <script type="text/javascript">
```

```
    function showButton(){
      var authRequest;
      OffAmazonPayments.Button("AmazonPayButton", "SELLER-ID", {
        type:  "TYPE",
        color: "COLOR",
        size:  "SIZE",

        authorization: function() {
        loginOptions = {scope: "SCOPES",
          popup: "POPUP-PARAMETER"};
        authRequest = amazon.Login.authorize (loginOptions,
          "REDIRECT-URL");
        }
  });
</script>
```

このコードサンプルで表示しているパラメータは、Amazon から提示された値、および、システムの設定値に置き換えてください。詳しい情報はウィジェットの「ボタンウィジェット」を参照してください。

4. ボタンコードのエラーハンドリングを追加

```
. . .
onError: function(error) {
  // your error handling code.
  // alert("The following error occurred: "
  //        + error.getErrorCode()
  //        + ' - ' + error.getErrorMessage());
}
. . .
```

onError ハンドラーについての詳しい情報は、エラーハンドリングの「Amazon Pay ウィジェットのエラーハンドリング」を参照してください。

5. ログアウトオプションの追加

```
<script type="text/javascript">
  document.getElementById('Logout').onclick = function() {
    amazon.Login.logout();
  };
</script>
```

通常はリンク形式でセットアップされるログアウトオプションは、Web サイトからキャッシュされたトークンを削除し、お名前などのお客さま個人情報を取り除きます。それから、Web サイトは再度ログインボタンを表示できます。キャッシュされたトークンを削除

するために amazon.Login.logout()メソッドを呼び出し、Amazon によって生成されたセッションをクリアします。その後の amazon.Login.authorize を呼び出しでは、デフォルトでログイン画面を表示します。

コードサンプル

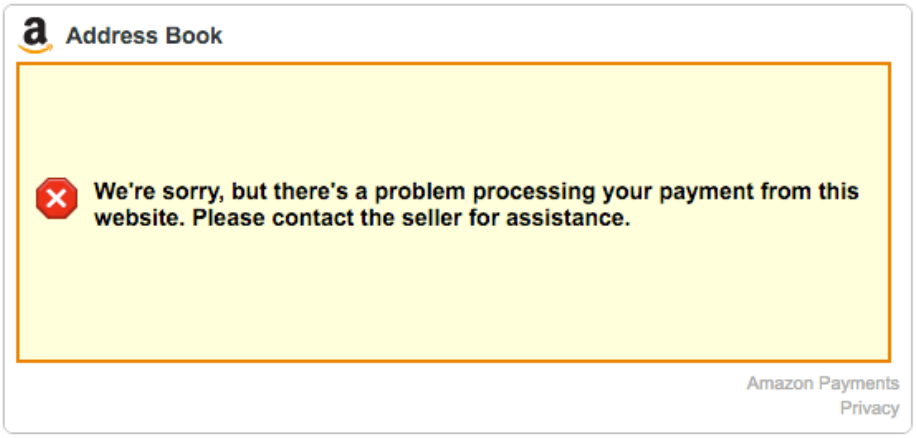
```
<head>
  <script type='text/javascript'>
    window.onAmazonLoginReady = function() {
      amazon.Login.setClientId('CLIENT-ID');
    };
    window.onAmazonPaymentsReady = function() {
                showButton();
    };
  </script>
    <script async="async" src='https://static-na.payments-
amazon.com/OffAmazonPayments/us/sandbox/js/Widgets.js'>
  </script>
</head>

<body>
. . .
 <div id="AmazonPayButton">
 </div>
  ...
 <script type="text/javascript">
    function showButton(){
      var authRequest;
      OffAmazonPayments.Button("AmazonPayButton", "SELLER-ID", {
        type:  "TYPE",
        color: "COLOR",
        size:  "SIZE",

        authorization: function() {
        loginOptions = {scope: "SCOPES",
          popup: "POPUP-PARAMETER"};
        authRequest = amazon.Login.authorize (loginOptions,
          "REDIRECT-URL");
        },

        onError: function(error) {
          // your error handling code.
          // alert("The following error occurred: "
          //        + error.getErrorCode()
          //        + ' - ' + error.getErrorMessage());
        }
     });
    };
   </script>
   . . .
   <script type="text/javascript">
     document.getElementById('Logout').onclick = function() {
       amazon.Login.logout();
     };
   </script>
```

```
</body>
```

Step 4:アドレス帳とお支払い方法ウィジェットの追加

購入者の認証に成功した後は、サイト上に Amazon Pay の**アドレス帳**と**お支払い方法**ウィジェットをレンダリングできます。**アドレス帳**と**お支払い方法**ウィジェットは、購入者の Amazon アカウントからそれぞれデフォルトの配送先住所と支払方法を表示します。

アドレス帳とお支払い方法ウィジェットが表示された場合は、購入者はセッションが切れるまでに 15 分間あります。一度セッションが切れると、ウィジェットの中でメッセージが表示され、購入者は新しい住所または支払方法を選択することを試みます。



処理内容

ウィジェットの埋め込み

1. 表示したい場所がある HTML ページにコードを追加することで Web サイトにウィジェットの埋め込みができます。
   Amazon Pay 機能を利用する多くのページでは、Widgets.js をロードする JavaScript ブロックにクライアント ID をセットしなければなりません。次のサンプルコードで示しているように同じ手順でコードが表示されるか確認します。これらは 2 つのスクリプトブロック間で他のコードで示すことができます。

```
<script>
  window.onAmazonLoginReady = function() {
    amazon.Login.setClientId('YOUR-CLIENT-ID');
  };
```

```
</script>

<script async='async' type='text/javascript'
src='https://static-na.payments-amazon.com/OffAmazonPayments/us/
       sandbox/js/Widgets.js'
</script>
```

上記のサンプルには非同期属性のスクリプトタグが含まれます。バックグラウンドで Amazon Pay を素早く Web ページにロードさせる JavaScript コードです。これは Amazon Pay が唯一サポートする非同期方法です。

2. モバイルに最適化された Web サイトのために、Web ページの head に<meta>タグを追加します。
この<meta>タグは、ユーザーが手動でページズームすること無しに、モバイルデバイスや電子書籍で参照可能な画面サイズに応じてウィジェットを調整します。

```
<meta name="viewport"
content="width-device-width,initial-scale=1.0, maximum-scale=1.0"/>
```

3. 幅と高さのパラメータを指定してウィジェットがレンダリングされます。（詳しく情報は、ウィジェットの「Amazon Pay ウィジェット」を参照してください。）

トランザクションフロー

アドレス帳と**お支払い方法**ウィジェットを表示した場合は、購入者はセッションが切れるまでに選択しなければなりません。セッションが切れた場合は、購入者はログイン画面が表示され、再度ログインしなければなりません。

トランザクションは次の通りに発生します。

1. 購入者の配送先住所を取得します。配送料や消費税を注文の合計金額に含めた計算を行うために購入者の配送先住所が必要になります。onAddressSelect 通知を受け取った後で、GetOrderReferenceDetails API 呼び出しにて住所を受け取ります。
AccessToken フィールドは GetOrderReferenceDetails 呼び出しを生成するために必要です。この Access Token は、レスポンスに完全な配送先住所を取得するためにどの scope パラメータをセットするのか影響します。

2. 購入者の支払方法を取得します。購入者が初期表示された住所の選択を変更した場合は、再度支払方法を選択しなければなりません。これはそれぞれの住所選択での onPaymentSelect 通知を確認しなければなりません。

3. 購入フローで次に進めることを可能にします。onPaymentSelect コールバックが発生した後では、次の購入フローを可能にできます。例えば、注文確認画面を用意している場合は、「購入する」ボタンや「次に進める」ボタンを有効にできます。

サンプルコード

Python サンプルコード（US）

GetOrderReferenceDetails API の呼び出し

```
from pay_with_amazon.client import PayWithAmazonClient

client = PayWithAmazonClient(
      mws_access_key='YOUR_ACCESS_KEY',
      mws_secret_key='YOUR_SECRET_KEY',
      merchant_id='YOUR_MERCHANT_ID',
      region='na',
      currency_code='USD')

response = client.get_order_reference_details(
      amazon_order_reference_id='AMAZON_ORDER_REFERENCE_ID',
      address_consent_token='ADDRESS_CONSENT_TOKEN')
```

Ruby サンプルコード（US）

GetOrderReferenceDetails API の呼び出し

```
require 'pay_with_amazon'

merchant_id = 'YOUR_MERCHANT_ID'
access_key = 'YOUR_ACCESS_KEY'
secret_key = 'YOUR_SECRET_KEY'

client = PayWithAmazon::Client.new(
      merchant_id,
      access_key,
      secret_key,
      sandbox: true,
      currency_code: :usd,
      region: :na
)

amazon_order_reference_id = 'AMAZON_ORDER_REFERENCE_ID'
address_consent_token = 'ADDRESS_CONSENT_TOKEN'
```

```
client.get_order_reference_details(
      amazon_order_reference_id,
      address_consent_token: address_consent_token,
      mws_auth_token: 'amzn.mws.4ea38b7b-f563-7709-4bae-87aeaEXAMPLE'
)
```

その他のサンプルコード（US）

GetOrderReferenceDetails API の呼び出し

```
https://mws.amazonservices.com/OffAmazonPayments_Sandbox/2013-01-01
     ?AWSAccessKeyId=AKIAJKYFSJU7PEXAMPLE
     &Action=GetOrderReferenceDetails
     &AddressConsentToken=IQEBLzAtAhUAjagYW4Jrgw84pCaaIDjrKoEhZXsEXAMPLE
     &AmazonOrderReferenceId=S23-1234567-1234567
     &SellerId=YOUR_SELLER_ID_HERE
     &SignatureMethod=HmacSHA256
     &SignatureVersion=2
     &Timestamp=2012-10-03T19%3A01%3A11Z
     &Version=2013-01-01
     &Signature=CLZOdtJGjAo81IxaLoE7af6HqK0EXAMPLE
```

アドレス帳ウィジェット

注意：同じページで両方のウィジェットを表示する場合は、お支払い方法ウィジェットはアドレス帳ウィジェットのローディングが完了してからロードされることを確認してください。

```
<div id="addressBookWidgetDiv">
</div>

<script>
  new OffAmazonPayments.Widgets.AddressBook({
    sellerId: 'YOUR_SELLER_ID_HERE',
    onOrderReferenceCreate: function(orderReference) {
      // Here is where you can grab the Order Reference ID.
      orderReference.getAmazonOrderReferenceId();
    },
    onAddressSelect: function(orderReference) {
      // Replace the following code with the action that you want
      // to perform after the address is selected. The
      // amazonOrderReferenceId can be used to retrieve the address
      // details by calling the GetOrderReferenceDetails operation.

      // If rendering the AddressBook and Wallet widgets
```

```
      // on the same page, you do not have to provide any additional
      // logic to load the Wallet widget after the AddressBook widget.

      // The Wallet widget will re-render itself on all subsequent
      // onAddressSelect events, without any action from you.
      // It is not recommended that you explicitly refresh it.
    },
    design: {
      designMode: 'responsive'
    },
    onReady: function(orderReference) {
      // Enter code here you want to be executed
      // when the address widget has been rendered.
    },

    onError: function(error) {
      // Your error handling code.
      // During development you can use the following
      // code to view error messages:
      // console.log(error.getErrorCode() + ': ' + error.getErrorMessage());
      // See "Handling Errors" for more information.
    }
  }).bind("addressBookWidgetDiv");
</script>
```

お支払い方法ウィジェット

```
<!-- Place this code in your HTML where you would like the
     Wallet widget to appear. -->
<div id="walletWidgetDiv">
</div>

<script>
  new OffAmazonPayments.Widgets.Wallet({
    sellerId: 'YOUR_SELLER_ID_HERE',
    onPaymentSelect: function(orderReference) {
      // Replace this code with the action that you want to perform
      // after the payment method is selected.

      // Ideally this would enable the next action for the buyer
      // including either a "Continue" or "Place Order" button.
    },
    design: {
      designMode: 'responsive'
    },

    onError: function(error) {
      // Your error handling code.
      // During development you can use the following
      // code to view error messages:
      // console.log(error.getErrorCode() + ': ' + error.getErrorMessage());
      // See "Handling Errors" for more information.
    }
  }).bind("walletWidgetDiv");
```

```
</script>
```

Step 5:注文詳細のセットと確認

支払要求を行う前に、**アドレス帳**と**お支払い方法**ウィジェットのステップ内で入手した **OrderReferenceId** を利用して注文のセットと確認を行います。

処理内容

1. SetOrderReferenceDetails API 呼び出しを行います。リクエストするには、**OrderReferenceAttributes** の次の属性をセットします。

   必須：
   - Amazon Order Reference ID
   - Order Total（注文金額と通貨コード）

   Amazon Order Reference ID はユニークな値であることを確認します。詳しい情報は「Amazon Pay API 呼び出しからのエラーハンドリング」を参照してください。

   強く勧めるオプション：
   - Store Name
   - Seller Order ID
   - Seller Note

   Store Name と Seller Information のセットは、購入者メールとアカウント状態に影響して詳細表示され、情報は IPN メッセージ内、決済とトランザクションレポートで返されます。

2. ConfirmOrderReference API の呼び出しを行います。SetOrderReferenceDetails API 呼び出しが成功した場合は、ConfirmOrderReference API 呼び出しを行うことで、OrderReference オブジェクトを確認します。このステップは購入者がサイト上で注文したことを Amazon に通知します。ConfirmOrderReference 呼び出しが成功した場合は、Order Reference オブジェクトは **Open** 状態にセットされ、Amazon は確認のために購入者に対して Amazon Pay を利用した購入支払がされることを「ご利用内容の確認」メールとして送信します。レスポンスが制約で返された場合は再処理を試みてください。例えば、**PaymentPlanNotSet** 制約のケースでは、購入者に異なる支払方法の選択するように案内します。これ以外のメッセージを画面上に表示できるか、購入者を適切なページに遷移することもできます。

3. GetOrderReferenceDetails 呼び出しを行います。Order Reference の承認が成功した後は、最新の住所であるか確実にするために、名前や配送先住所のような残りの購入者情報を取得するための GetOrderReferenceDetails API を呼び出すことができます。

注意：API 呼び出しのエラーハンドリングを導入する必要があり、API レスポンスの結果についてテストする必要があります。詳しい情報は「エラーハンドリング」を参照してください。

サンプルコード

Python サンプルコード（US）

SetOrderReferenceDetails API の呼び出し

```
from pay_with_amazon.client import PayWithAmazonClient
      client = PayWithAmazonClient(
      mws_access_key='YOUR_ACCESS_KEY',
      mws_secret_key='YOUR_SECRET_KEY',
      merchant_id='YOUR_MERCHANT_ID',
      region='na',
      currency_code='USD')

response = client.set_order_reference_details(
      amazon_order_reference_id='AMAZON_ORDER_REFERENCE_ID',
      order_total='1.00',
      seller_note='My seller note.',
      seller_order_id='MY_UNIQUE_ORDER_ID',
      store_name='My store name.',
      custom_information='My custom information.')
```

ConfirmOrderReference API の呼び出し

```
from pay_with_amazon.client import PayWithAmazonClient

      client = PayWithAmazonClient(
      mws_access_key='YOUR_ACCESS_KEY',
      mws_secret_key='YOUR_SECRET_KEY',
      merchant_id='YOUR_MERCHANT_ID',
      region='na',
      currency_code='USD')

response = client.confirm_order_reference(
      amazon_order_reference_id='AMAZON_ORDER_REFERENCE_ID')
```

GetOrderReferenceDetails API の呼び出し

```
Making a call to the GetOrderReferenceDetails API
```

```
from pay_with_amazon.client import PayWithAmazonClient

      client = PayWithAmazonClient(
      mws_access_key='YOUR_ACCESS_KEY',
      mws_secret_key='YOUR_SECRET_KEY',
      merchant_id='YOUR_MERCHANT_ID',
      region='na',
      currency_code='USD')

response = client.get_order_reference_details(
      amazon_order_reference_id='AMAZON_ORDER_REFERENCE_ID',
      address_consent_token='ADDRESS_CONSENT_TOKEN')
```

Ruby サンプルコード（US）

SetOrderReferenceDetails API の呼び出し

```
# SetOrderReferenceDetails

require 'pay_with_amazon'

merchant_id = 'YOUR_MERCHANT_ID'
access_key = 'YOUR_ACCESS_KEY'
secret_key = 'YOUR_SECRET_KEY'

client = PayWithAmazon::Client.new(
      merchant_id,
      access_key,
      secret_key,
      sandbox: true,
      currency_code: :usd,
      region: :na
)

amazon_order_reference_id = 'AMAZON_ORDER_REFERENCE_ID'
amount = 106

client.set_order_reference_details(
      amazon_order_reference_id,
      amount,
      seller_note: 'Lorem ipsum dolor',
      seller_order_id: '5678-23',
      store_name: 'CourtAndCherry.com',
      mws_auth_token: 'amzn.mws.4ea38b7b-f563-7709-4bae-87aeaEXAMPLE'
)
```

ConfirmOrderReference API の呼び出し

```
Making a call to the ConfirmOrderReference API

client.confirm_order_reference(
      amazon_order_reference_id,
      mws_auth_token: 'amzn.mws.4ea38b7b-f563-7709-4bae-87aeaEXAMPLE'
)
```

GetOrderReferenceDetails API の呼び出し

```
require 'pay_with_amazon'

merchant_id = 'YOUR_MERCHANT_ID'
access_key = 'YOUR_ACCESS_KEY'
secret_key = 'YOUR_SECRET_KEY'

client = PayWithAmazon::Client.new(
      merchant_id,
      access_key,
      secret_key,
      sandbox: true,
      currency_code: :usd,
      region: :na
)

amazon_order_reference_id = 'AMAZON_ORDER_REFERENCE_ID'
address_consent_token = 'ADDRESS_CONSENT_TOKEN'

client.get_order_reference_details(
      amazon_order_reference_id,
      address_consent_token: address_consent_token,
      mws_auth_token: 'amzn.mws.4ea38b7b-f563-7709-4bae-87aeaEXAMPLE'
)
```

その他のサンプルコード（US）

SetOrderReferenceDetails API の呼び出し

```
https://mws.amazonservices.com/OffAmazonPayments_Sandbox/2013-01-01
?AWSAccessKeyId=0GS7553JW74RRM612K02EXAMPLE
&Action=SetOrderReferenceDetails
&AmazonOrderReferenceId=S23-1234567-1234567
&OrderReferenceAttributes.OrderTotal.Amount=106
&OrderReferenceAttributes.OrderTotal.CurrencyCode=USD
&OrderReferenceAttributes.SellerNote=For your order of Casad
```

```
 '2Jours Bonheur' Satchel in Sandstorm
&OrderReferenceAttributes.SellerOrderAttributes.SellerOrderId=5678-23
&OrderReferenceAttributes.SellerOrderAttributes.StoreName=CourtAndCherry.com
&SellerId=YOUR_SELLER_ID_HERE
&SignatureMethod=HmacSHA256
&SignatureVersion=2
&Timestamp=2013-11-16T19:01:11Z
&Version=2013-01-01
&Signature=2RPzkOgQmDybUjk0dA54maCEXAMPLE
```

ConfirmOrderReference API の呼び出し

```
https://mws.amazonservices.com/OffAmazonPayments_Sandbox/2013-01-01
?AWSAccessKeyId=AKIAJKYFSJU7PEXAMPLE
&Action=ConfirmOrderReference
&AmazonOrderReferenceId=S23-1234567-1234567
&SellerId=YOUR_SELLER_ID_HERE
&SignatureMethod=HmacSHA256
&SignatureVersion=2
&Timestamp=2013-11-16T19:01:11Z
&Version=2013-01-01
&Signature=CLZOdtJGjAo81IxaLoE7af6HqK0EXAMPLE
```

GetOrderReferenceDetails API の呼び出し

```
https://mws.amazonservices.com/OffAmazonPayments_Sandbox/2013-01-01
?AWSAccessKeyId=AKIAJKYFSJU7PEXAMPLE
&Action=GetOrderReferenceDetails
&AddressConsentToken=IQEBLzAtAhUAjagYW4Jrgw84pCaaIDjrKoEhZXsEXAMPLE
&AmazonOrderReferenceId=S23-1234567-1234567
&SellerId=YOUR_SELLER_ID_HERE
&SignatureMethod=HmacSHA256
&SignatureVersion=2
&Timestamp=2012-10-03T19:01:11Z
&Version=2013-01-01
&Signature=CLZOdtJGjAo81IxaLoE7af6HqK0EXAMPLE
```

Step 6:オーソリ（Authorize）のリクエスト

オーソリ（Authorize）API の呼び出しは、購入者が購入処理中に選択し、Order Reference オブジェクトに保存された支払方法に対して指定した購入金額を確保します。

オーソリに成功した場合は、**AuthorizationStatus** が **Open** であるオーソリオブジェクトが生成されます。これは次のステップの購入金額の請求が許可されます。オーソリオブジェクトは 30 日間 **Open** 状態を維持し、1 つの **Open** 状態の Order Reference に対して 10 個のオーソリを作成できます。

注意：クレジットカードの有効性を確認するために 1 円の Authorize API を要求するのはお薦めしません。

非同期と同期のオーソリ(Authorize)API 呼び出し

運用ルールの要件次第でオーソリ（Authorize）API 呼び出しのモードを選択します。

- **非同期オーソリ**
  商品を配送した段階で請求したい場合は非同期モードを利用します。24 時間以上注文を保持する場合はこのモードを利用します。なぜならば、Amazon に注文確定した後で、すぐに購入者へ注文完了ページを表示する場合では、最終処理ステータスはリアルタイムで有効ではないからです。オーソリが **Declined** ステータスである場合は、トランザクションが失敗したことを購入者に通知する必要があり、Amazon Pay Web サイトで支払方法の更新を要求し、その他の正しい支払い方法を選択してもらうか、失敗した理由コードに基いて注文をキャンセルします。非同期モードは一般的に低いオーソリ失敗率になり、Amazon Pay がトランザクションを調査する時間を多く提供します。

- **同期オーソリ**
  購入者がサイトにいる間に支払のオーソリを行いたい場合は、同期モードを利用します。例えば、デジタルダウンロードの提供や速やかに配送を確認できる場合は同期モードを利用します。同期モードを選択した場合は、非同期モードに比べて、**Pending** 状態のオーソリを **Declined** 状態に遷移させるような高いオーソリ失敗率になるかもしれません。**理由コード**が **TransactionTimedOut** によって、これらのオーソリ失敗を確認することができます。

注意：オーソリで **InvalidPaymentMethod** の理由コードが返された場合は、Hard Decline と Soft Decline の異なる 2 つの処理方法を利用できる SoftDecline パラメータのオプションがあります。Soft Decline のケースでは、追加のオーソリを要求できます。Soft Decline についての追加の情報は「オーソリ失敗のハンドリング」を参照してください。

処理内容

次のステップでオーソリをリクエストします。

1. オーソリ（Authorize） API を呼び出します。オーソリリクエストに次の値をセットします。

要素	値

AuthorizationReferenceId	販売事業者が生成するオーソリのユニークな ID です。これは Amazon によって生成される **AmazonAuthorizationId** とは異なります。
SellerAuthorizationNote	購入者へのメールに表示されるトランザクションの説明です。**CaptureNow** に **true** をセットしている場合でのみ表示されます。
SoftDescriptor	**CaptureNow** に **true** をセットしている場合に、この値は購入者の支払明細書に表示されますます。
TransactionTimeout	非同期オーソリ API 呼び出し： **TransactionTimeout** には最小値 5 から最大値 1440（デフォルト値）分までの値を 5 の倍数でセットします。 オーソリがこのタイムリミット内で処理できなかった場合に理由コードを **TransactionTimeOut** として失敗します。 非同期フローを利用する場合は、**AuthorizationStatus** レスポンス要素は常に **Pending** がセットされます。Amazon によって処理されると、オーソリリクエストの最終ステータスを IPN 経由で受け取ります。（例えば、**Open** や **Declined** です。） 同期オーソリ API 呼び出し： **TransactionTimeout** には 0（ゼロ）分をセットします。 **AuthorizationStatus** は、通常は 13 秒以内に **Open** か **Declined** ステータスを返します。 **CaptureNow** に **true** をセットした場合は、オーソリ（Authorize）API 呼び出しでの **SellerAuthorizationNote** の情報は、購入者への支払確認メールと決済レポートに表示されます。

注意：AuthorizationReferenceId、SellerAuthorizationNote、SoftDescriptor の値は、購入者の金額確認メールとアカウントステータス、購入者の利用明細に表示され、決済とトランザクションレポートにも表示されます。

2. オーソリの詳細を取得します。**AmazonAuthorizationId** を利用して GetAuthorizationDetails 処理を呼び出すことでオーソリオブジェクトの詳細を要求することができ、それをオーソリのレスポンスとして受け取ります。

注意：API 呼び出しでのエラーハンドリングを導入し、API レスポンスの結果をテストしなければなりません。詳しい情報はエラーハンドリングを参照してください。

サンプルコード

Python のサンプルコード（US）

Authorize API 呼び出し

```
from pay_with_amazon.client import PayWithAmazonClient

client = PayWithAmazonClient(
      mws_access_key='YOUR_ACCESS_KEY',
      mws_secret_key='YOUR_SECRET_KEY',
      merchant_id='YOUR_MERCHANT_ID',
      region='na',
      currency_code='USD')

response = client.authorize(
      amazon_order_reference_id='AMAZON_ORDER_REFERENCE_ID',
      authorization_reference_id='MY_UNIQUE_AUTHORIZATION_ID',
      amount='1.00',
      seller_authorization_note='Authorization note.'
      capture_now=False)
```

Authorize 非同期 API 呼び出し

```
from pay_with_amazon.client import PayWithAmazonClient

client = PayWithAmazonClient(
      mws_access_key='YOUR_ACCESS_KEY',
      mws_secret_key='YOUR_SECRET_KEY',
      merchant_id='YOUR_MERCHANT_ID',
      region='na',
      currency_code='USD')
```

```
response = client.authorize(
      amazon_order_reference_id='AMAZON_ORDER_REFERENCE_ID',
      authorization_reference_id='MY_UNIQUE_AUTHORIZATION_ID',
      amount='1.00',
      seller_authorization_note='Authorization note.'
      transaction_timeout=60,
      capture_now=False)
```

Authorize 同期 API 呼び出し

```
from pay_with_amazon.client import PayWithAmazonClient

client = PayWithAmazonClient(
      mws_access_key='YOUR_ACCESS_KEY',
      mws_secret_key='YOUR_SECRET_KEY',
      merchant_id='YOUR_MERCHANT_ID',
      region='na',
      currency_code='USD')

response = client.authorize(
      amazon_order_reference_id='AMAZON_ORDER_REFERENCE_ID',
      authorization_reference_id='MY_UNIQUE_AUTHORIZATION_ID',
      amount='1.00',
      seller_authorization_note='Authorization note.'
      transaction_timeout=0,
      capture_now=False)
```

Ruby のサンプルコード（US）

Authorize API 呼び出し

```
# Authorize

require 'pay_with_amazon'

merchant_id = 'YOUR_MERCHANT_ID'
access_key = 'YOUR_ACCESS_KEY'
secret_key = 'YOUR_SECRET_KEY'

client = PayWithAmazon::Client.new(
      merchant_id,
      access_key,
      secret_key,
      sandbox: true,
      currency_code: :usd,
```

```
      region: :na
)

amazon_order_reference_id = 'AMAZON_ORDER_REFERENCE_ID'
authorization_reference_id = 'test_authorize_1'
amount = 106

client.authorize(
      amazon_order_reference_id,
      authorization_reference_id,
      amount,
      seller_authorization_note: 'Lorem ipsum dolor',
      mws_auth_token: 'amzn.mws.4ea38b7b-f563-7709-4bae-87aeaEXAMPLE'
)
```

Authorize 非同期 API 呼び出し

```
# Authorize

require 'pay_with_amazon'

merchant_id = 'YOUR_MERCHANT_ID'
access_key = 'YOUR_ACCESS_KEY'
secret_key = 'YOUR_SECRET_KEY'

client = PayWithAmazon::Client.new(
      merchant_id,
      access_key,
      secret_key,
      sandbox: true,
      currency_code: :usd,
      region: :na
)

amazon_order_reference_id = 'AMAZON_ORDER_REFERENCE_ID'
authorization_reference_id = 'test_authorize_1'
amount = 106

client.authorize(
      amazon_order_reference_id,
      authorization_reference_id,
      amount,
      seller_authorization_note: 'Lorem ipsum dolor',
      transaction_timeout: 60,
      mws_auth_token: 'amzn.mws.4ea38b7b-f563-7709-4bae-87aeaEXAMPLE'
)
```

Authorize 同期 API 呼び出し

```
# Authorize

require 'pay_with_amazon'

merchant_id = 'YOUR_MERCHANT_ID'
access_key = 'YOUR_ACCESS_KEY'
secret_key = 'YOUR_SECRET_KEY'

client = PayWithAmazon::Client.new(
      merchant_id,
      access_key,
      secret_key,
      sandbox: true,
      currency_code: :usd,
      region: :na
)

amazon_order_reference_id = 'AMAZON_ORDER_REFERENCE_ID'
authorization_reference_id = 'test_authorize_1'
amount = 106

client.authorize(
      amazon_order_reference_id,
      authorization_reference_id,
      amount,
      seller_authorization_note: 'Lorem ipsum dolor',
      transaction_timeout: 0,
      mws_auth_token: 'amzn.mws.4ea38b7b-f563-7709-4bae-87aeaEXAMPLE'
)
```

その他のサンプルコード（US）

非同期モードの Authorize API 呼び出し

```
https://mws.amazonservices.com/OffAmazonPayments_Sandbox/2013-01-01
    ?AWSAccessKeyId=AKIAFBM3LG5JEEXAMPLE
    &Action=Authorize
    &AmazonOrderReferenceId=S23-1234567-1234567
    &AuthorizationAmount.Amount=94.50
    &AuthorizationAmount.CurrencyCode=USD
    &AuthorizationReferenceId=test_authorize_1
    &SellerAuthorizationNote=Authorization for Blue Shoes
```

```
    &SellerId=YOUR_SELLER_ID_HERE
    &SignatureMethod=HmacSHA256
    &SignatureVersion=2
    &Timestamp=2013-11-16T19:01:11Z
    &TransactionTimeout=60
    &Version=2013-01-01
    &Signature=WlQ708aqyHXMkoUBk69Hjxj8qdh3aDcqpY71hVgEXAMPLE
```

GetAuthorizationDetails API 呼び出し

```
POST /OffAmazonPayments/2013-01-01 HTTP/1.1
      Content-Type: x-www-form-urlencoded
      Host: mws.amazonservices.com
      User-Agent: <your user="" agent="" header="">

      AWSAccessKeyId=AKIAFBM3LG5JEEXAMPLE
      &Action=GetAuthorizationDetails
      &AmazonAuthorizationId=P01-1234567-1234567-0000001
      &SellerId=YOUR_SELLER_ID_HERE
      &SignatureMethod=HmacSHA256
      &SignatureVersion=2
      &Timestamp=2012-11-05T19:01:11Z
      &Version=2013-01-01
      &Signature=WlQ708aqyHXMkoUBk69Hjxj8qdh3aDcqpY71hVgEXAMPLE
    </your>
```

Step 7:売上請求（Capture）のリクエスト

売上請求はオーソリ中の確保された資金を販売事業者の口座に資金移動することになります。

資金を回収するためには、本番環境モードではオーソリに成功してから 30 日以内に売上請求（Capture）API を呼び出します。テスト環境モードでは 2 日以内です。30 日を過ぎた場合は、Amazon によってオーソリオブジェクトは **Closed** 状態にします。

Amazon Pay のポリシーでは、注文された商品を出荷してから売上請求しなければなりません。よって、商品を出荷する前に売上請求してはなりません。

購入者から回収できた場合は、売上請求オブジェクトステータスは **Completed** になります。

注意：既に返金を行っている場合は、売上請求として有効ではありません。

処理内容

次のステップで売上請求をリクエストします。

1. 売上請求（Capture） API を呼び出します。リクエスト内の **Seller Capture Note** をセットして、購入者へのメールにこの情報を含めるます。売上請求はリアルタイムで受け付けますが、Amazon 内では非同期で処理されますので、すぐに処理されませんので注意してください。売上請求リクエストが処理された場合は売上請求オブジェクトは **Completed** か **Declined** 状態になります。
2. 売上請求リクエストでの **CaptureStatus** の確認するために売上請求のインスタント支払通知（IPN）を確認します。リクエストされた直後は **Pending** 状態になっており、Amazon が処理を完了するまではこの状態のままです。処理が完了するまで 1 時間以上掛かる場合があります。Amazon にて処理が出来なかった場合は、Capture メッセージとして **Declined** を受け取ります。
3. GetCaptureDetails API を呼び出します。**AmazonCaptureId** を利用して GetCaptureDetails API を呼び出しレスポンスを受け取ることで、売上請求オブジェクトの完全な詳細情報を要求することができます。**AmazonCaptureId** は完了した売上請求に対して返金する場合に必要になります。
4. **エラー**または **Declined** のレスポンスをチェックします。API 呼び出しから発生したエラーのハンドリングが必要であり、**Declined** 状態の売上請求の管理も必要です。売上請求が **Declined** の場合は、レスポンス内に次の理由コードがあります。

   AmazonRejected
   Amazon の決定によって売上請求が **Declined** になります。Amazon が売上請求を拒否した場合は、オーソリオブジェクトと Order Reference オブジェクトは **Closed** になるかもしれません。オーソリオブジェクトが **Closed** の場合は、それに関連する売上請求リクエストも Declined になります。Order Reference オブジェクトがまだ **Open** 状態であれば、新しいオーソリとそれからの売上請求をリクエストします。

   ProcessingFailure
   Amazon が内部処理エラーの原因でトランザクションを処理できませんでした。１、２分後に再度リクエストしてください。Order Reference オブジェクトがまだ **Open** の場合は、Amazon は新しいオーソリ（Authorize） API 呼び出し、次に新しい売上請求（Capture） API を行うことを勧めます。

   エラーハンドリングについての詳しい情報は、Amazon Pay API 呼び出しのエラーハンドリングを参照してください。

サンプルコード

Python のサンプルコード（US）

Capture API 呼び出し

```
from pay_with_amazon.client import PayWithAmazonClient

client = PayWithAmazonClient(
      mws_access_key='YOUR_ACCESS_KEY',
      mws_secret_key='YOUR_SECRET_KEY',
      merchant_id='YOUR_MERCHANT_ID',
      region='na',
      currency_code='USD')

# authorization ID returned from 'Authorize' call.
response = client.capture(
      amazon_authorization_id='MY_ATHORIZATION_ID',
      capture_reference_id='MY_UNIQUE_CAPTURE_ID',
      capture_amount='1.00',
      seller_capture_note='Capture note.')
```

Ruby のサンプルコード（US）

Capture API 呼び出し

```
# Capture

require 'pay_with_amazon'

merchant_id = 'YOUR_MERCHANT_ID'
access_key = 'YOUR_ACCESS_KEY'
secret_key = 'YOUR_SECRET_KEY'

client = PayWithAmazon::Client.new(
      merchant_id,
      access_key,
      secret_key,
      sandbox: true,
      currency_code: :usd,
      region: :na
)

amazon_authorization_id = 'S23-1234567-1234567-0000001'
capture_reference_id = 'test_capture_1'
amount = 106

client.capture(
      amazon_authorization_id,
      capture_reference_id,
```

```
      amount,
      seller_capture_note: 'Lorem ipsum dolor',
      mws_auth_token: 'amzn.mws.4ea38b7b-f563-7709-4bae-87aeaEXAMPLE'
)
```

その他のサンプルコード（US）

Capture API 呼び出し

```
https://mws.amazonservices.com/OffAmazonPayments_Sandbox/2013-01-01
?AWSAccessKeyId=AKIAFBM3LG5JEEXAMPLE
&Action=Capture
&AmazonAuthorizationId=S23-1234567-1234567-0000001
&CaptureAmount.Amount=199.00
&CaptureAmount.CurrencyCode=USD
&CaptureReferenceId=test_capture_1
&SellerCaptureNote=For%20your%20order%20of%20Casad%20%272Jours%20Bonheur
%27%20Satchel%20in%20Sandstorm
&SellerId=YOUR_SELLER_ID_HERE
&SignatureMethod=HmacSHA256
&SignatureVersion=2
&Timestamp=2013-11-19T19%3A01%3A11Z
&Version=2013-01-01
&Signature=WlQ708aqyHXMkoUBk69Hjxj8qdh3aDcqpY71hVgEXAMPLE
```

Step 8:Order Reference を closed に変更

注文商品の出荷が完了した場合に Order Reference を **Open** または **Suspended** 状態から **Closed** に変更し、これ以上のオーソリをリクエストできないようにします。追加のオーソリを行う必要がない場合は **Closed** にすることをお勧めしますが、部分出荷などで分割請求した場合は残存する請求ができなくなりますので、すべてが完了するまで、もしくは、これ以上の金額追加がない場合のみ Order Reference を **Closed** にしてください。

Amazon は注文支払が完了したことを Order Reference が **Closed** されたとして購入者に示します。別の方法では、購入者は Amazon Pay の Web サイト上でトランザクションを確認できます。Order Reference は **Closed** にしていなければ 180 日間は **Open** 状態で表示され、180 日以降は Amazon が Order Reference オブジェクトを **Expired** の理由コードを付けて **Closed** にします。

Order Reference オブジェクトが **Closed** になった場合は以下の通りになります。

- Order Reference 上の Open のオーソリ（**Authorization**）に対して売上請求（**Capture**）ができます。
- Order Reference オブジェクト上に新しいオーソリ（**Authorization**）は作成できません。
- Order Reference オブジェクトに対して返金（**Refund**）することができます。

処理内容

1. ClosedOrderReference API の呼び出しを行います。

Python のサンプルコード（US）

```
from pay_with_amazon.client import PayWithAmazonClient

client = PayWithAmazonClient(
        mws_access_key='YOUR_ACCESS_KEY',
        mws_secret_key='YOUR_SECRET_KEY',
        merchant_id='YOUR_MERCHANT_ID',
        region='na',
        currency_code='USD')

response = client.cancel_order_reference(
    amazon_order_reference_id='AMAZON_ORDER_REFERENCE_ID',
    cancelation_reason='My cancel reason.')
```

Ruby のサンプルコード（US）

```
# CloseOrderReference

require 'pay_with_amazon'

merchant_id = 'YOUR_MERCHANT_ID'
access_key = 'YOUR_ACCESS_KEY'
secret_key = 'YOUR_SECRET_KEY'

client = PayWithAmazon::Client.new(
  merchant_id,
  access_key,
  secret_key,
  sandbox: true,
  currency_code: :usd,
  region: :na
)

amazon_order_reference_id = 'AMAZON_ORDER_REFERENCE_ID'

client.close_order_reference(
  amazon_order_reference_id,
  mws_auth_token: 'amzn.mws.4ea38b7b-f563-7709-4bae-87aeaEXAMPLE'
)
```

Step 9:返金のリクエスト

既に成功している売上請求金額に対して、全部または一部の返金を行うことができます。

注意：返金が呼び出しされた場合は、もはや資金はオーソリや売上請求できません。返金処理が成功した場合は、返金（**Refund**）オブジェクトは **Completed** 状態になり、購入者に返金通知メールが送信されます。

Amazon によって返金が失敗した場合は、返金オブジェクトは **Declined** 状態になります。詳しい情報は以下の Step4「エラーまたは失敗のレスポンスを確認」を参照してください。

返金するための期限はありません。しかしながら、購入から長い時間が経った返金要求は無効になったり、失敗する場合があります。

注意：めったに返金することがない場合は、ワークフロー内で返金（Refund） API を呼び出す代わりにセラーセントラル経由で返金を手動で呼び出すことができます。

処理内容

次のステップの通りに返金をリクエストします。

1. 返金（Refund） API の呼び出しを行います。次の値を返金リクエスト内にセットすることで、購入者への返金通知メールとアカウントステータスに表示され、決済とトランザクションレポートにも表示されます。

    - Refund Reference ID（返金用 Reference ID）
    - Refund amount（返金金額）
    - Seller Refund Note（販売事業者用の返金メモ）

    Reference ID はユニークであることを確認します。詳しい情報は Amazon Pay API 呼び出しのエラーハンドリングを参照してください。
2. Amazon から返される返金の IPN メッセージを確認します。返金リクエストはリアルタイムで処理されず、最初の **RefundStatus** は常に **Pending** です。処理時間は変動し、数時間かかります。Amazon は処理ステータスを IPN メッセージ経由で通知します。
3. 返金詳細を取得します。返金の IPN メッセージから返された **AmazonRefundId** を利用して GetRefundDetails API を呼び出して、返金オブジェクトの詳細を問い合わせできます。

4. エラーか失敗のレスポンスがないか確認します。API 呼び出しで発生するかもしれないエラーについてハンドリングが必要であり、**Declined** 状態にの返金について管理が必要です。
   返金が Declined の場合は、レスポンス内に次の理由コードを参照できます。

   AmazonRejected
   売上請求に対して既に最大金額の返金を行った場合は、Amazon が返金を失敗させます。それでも購入者に返金を行いたい場合は、その他の方法として、ショップ用のギフトカードやクーポンの提供を行うことになります。

   ProcessingFailure
   Amazon は次の理由によりトランザクションを処理できませんでした。
   - 内部処理エラーによりトランザクションを処理できませんでした。
   - Amazon マーケットプレイス保証、または、チャージバックが既に発行されている。

   ProcessingFailure の理由コードを受け取った場合は、セラーセントラルの**パフォーマンス**をクリックして Amazon マーケットプレイス保証かチャージバックが発生していないか確認し、売上請求オブジェクトが **Completed** 状態か確認した後で、１，２分後にリトライしてください。リトライが成功しなかった場合は、購入者に問い合わせし、他の方法で返金を行ってください。

   詳しい情報は Amazon Pay API 呼び出しのエラーハンドリングを参照してください。

Python のサンプルコード（US）

```
from pay_with_amazon.client import PayWithAmazonClient

client = PayWithAmazonClient(
        mws_access_key='YOUR_ACCESS_KEY',
        mws_secret_key='YOUR_SECRET_KEY',
        merchant_id='YOUR_MERCHANT_ID',
        region='na',
        currency_code='USD')

response = client.refund(
    amazon_capture_id='AMAZON_CAPTURE_ID',
    refund_reference_id='REFUND_REFERENCE_ID',
    refund_amount='1.00',
    seller_refund_note='Test order.')
```

Ruby のサンプルコード（US）

```
# Refund

require 'pay_with_amazon'
```

```
merchant_id = 'YOUR_MERCHANT_ID'
access_key = 'YOUR_ACCESS_KEY'
secret_key = 'YOUR_SECRET_KEY'

client = PayWithAmazon::Client.new(
  merchant_id,
  access_key,
  secret_key,
  sandbox: true,
  currency_code: :usd,
  region: :na
)

amazon_capture_id = 'S23-1234567-1234567-0000002'
refund_reference_id = 'test_refund_1'
amount = 106

client.refund(
  amazon_capture_id,
  refund_reference_id,
  amount,
  seller_refund_note: 'Lorem ipsum dolor',
  mws_auth_token: 'amzn.mws.4ea38b7b-f563-7709-4bae-87aeaEXAMPLE'
)
```

その他のサンプルコード（US）

```
https://mws.amazonservices.com/OffAmazonPayments_Sandbox/2013-01-01
?AWSAccessKeyId=AKIAFBM3LG5JEEXAMPLE
&Action=Refund
&AmazonCaptureId=S23-1234567-1234567-0000002
&RefundAmount.Amount=199.00
&RefundAmount.CurrencyCode=USD
&RefundReferenceId=test_refund_1
&SellerRefundNote=For%20your%20returned%20item%2C%20Casad%20%272Jours
%20Bonheur%27%20Satchel%20in%20Sandstorm
&SellerId=YOUR_SELLER_ID_HERE
&SignatureMethod=HmacSHA256
&SignatureVersion=2
&Timestamp=2013-11-27T19%3A01%3A11Z
&Version=2013-01-01
&Signature=WlQ708aqyHXMkoUBk69Hjxj8qdh3aDcqpY71hVgEXAMPLE
```

インテグレーションステップのオプション

ログイン時での注文情報へのアクセス

購入者のログイン時に注文を生成している場合は、Order Reference オブジェクトを生成する Amazon Pay ボタンに onSignIn 関数を追加して注文詳細にアクセスできます。

Order Reference オブジェクトが生成されている場合は、注文情報の詳細が含まれている GetOrderReferenceDetails API を呼び出すことができます。

次のコードサンプルは、購入者がログインし生成された Order Reference オブジェクトの onSignIn 関数を表しています。

```
<div id="AmazonPayButton"/>

<script type="text/javascript">
  var authRequest;
  var addressConsentToken;
  OffAmazonPayments.Button("AmazonPayButton", "YOUR_SELLER_ID", {
    type:  "ENTER_TYPE_PARAMETER",
    color: "ENTER_COLOR_PARAMETER",
    size:  "ENTER_SIZE_PARAMETER",
    language: "ENTER_LANGUAGE_PARAMETER",

    authorization: function() {
      loginOptions = {scope: "ENTER_SCOPES",
                      popup: "ENTER_POPUP_PARAMETER"};
      authRequest = amazon.Login.authorize (loginOptions,
        function(response) {
          addressConsentToken = response.access_token;
        });
    },
    onSignIn: function (orderReference) {
      var referenceId = orderReference.getAmazonOrderReferenceId();

      if (!referenceId) {
        errorHandler(new Error('referenceId missing'));
      } else {
        window.location = "YOUR_REDIRECT_URL" + '?referenceId=' +
          orderReference.getAmazonOrderReferenceId() +
          "&access_token=" + addressConsentToken;
      }
    },
    onError:errorHandler || function(error) {
      // your error handling code
    }
  });
</script>
```

リダイレクト認証を有効にする。

購入者がモバイルデバイスを利用しているなどのいくつかのケースでは、ポップアップウィンドウを表示するよりも、同じウィンドウ内で Amazon 認証ページへ購入者をリダイレクトするのがベストプラクティスです。

処理内容

次のステップでは、Web サイトでリダイレクト認証を有効にする方法を説明します。

1. ボタンウィジェット内のポップアップパラメータを **false** にセットします。
2. Web ページ上で購入者をリダイレクトしたい場合は、ウィジェットで利用するために必要なアクセストークンを受け取るために、<head>セクションに JavaScript コードを追加しなければなりません。詳しい情報はアクセストークンを参照してください。
   次のサンプルコードは、アクセストークンを受け取る JavaScript コードを表します。

```
<script type='text/javascript'>
  function getURLParameter(name, source) {
    return decodeURIComponent((new RegExp('[?|&|#]' + name + '=' +
      '([^&]+?)(&|#|;|$)').exec(source) || [,""])[1].replace(/\+/g,
      '%20')) || null;
  }

  var accessToken = getURLParameter("access_token", location.hash);

  if (typeof accessToken === 'string' && accessToken.match(/^Atza/)) {
    document.cookie = "amazon_Login_accessToken=" + accessToken +
      ";secure";
  }

  window.onAmazonLoginReady = function () {
    amazon.Login.setClientId('amzn1.application-oa2-
    client.d607ddd4957c44019e73086bc7cSAMPLE');

    amazon.Login.setUseCookie(true);
  };

</script>

<script src='https://static-na.payments-
amazon.com/OffAmazonPayments/us/
  sandbox/js/Widgets.js?sellerId=YOUR_SELLER_ID'>
</script>
```

3. ウィジェットの Cookie を削除します。Amazon ログイン JavaScript SDK を利用している場合は、キャッシュされたトークンを削除するために amazon.Login.logout メソッドを呼び出すことができます。リダイレクト認証を利用している場合は、**アドレス帳**と**お支払い方法**ウィジェットで利用された Cookie も削除する必要があります。デフォルトのログイン画面を表示するために、amazon.Login.authorize の呼び出しを確実にします。次のサ

ンプルコードは、Amazon ログイン JavaScript SDK の amazon.Login.logout メソッドを利用して、キャッシュされたトークンを削除します。

```
<script type="text/javascript">
  document.getElementById('Logout').onclick = function() {
    amazon.Login.logout();
    document.cookie = "amazon_Login_accessToken=;
      expires=Thu, 01 Jan 1970 00:00:00 GMT";
    window.location = 'REDIRECT_URL';
  };
</script>
```

クロスサイトリクエストフォージェリ

クロスサイトリクエストフォージェリは、攻撃者がユーザーを認証させて悪意のあるリンクをクリックさせて詐称を行うことで起こります。ユーザーは既にサイト上で認証されているので、多くのコマンドが埋め込まれた悪意のあるリンクは自動的に実行されます。よって、ユーザーはログイン画面や悪意のある活動履歴を参照しません。Amazon ログインのケースでは、クロスサイトリクエストフォージェリはクライアント、または、認証サーバを偽装して利用することができます。

Amazon ログインのためにクライアントを登録した場合は、クライアント識別子（client_id）とクライアントシークレット（client_secret）が割り与えられます。クライアントは、アクセストークン要求内の client_id と client_secret パラメータを利用してその識別子を確認します。（これらの値は、攻撃者によって偽装されなければなりません）

Amazon ログインは、クロスサイトリクエストフォージェリを防ぐために State パラメータも利用します。クライアントは、認証要求を初期化する場合に、State パラメータの値をセットします。client_id と client_secret が同じでない場合は、State の値はそれぞれの認証要求で異なります。認証コードとアクセストークンを利用してクライアントが通信している場合は、認証サーバは同じ値を返します。最初の呼び出しから State パラメータの値が一致しない場合は、通信は無視するべきです。

購入者が同意しない場合

Web サイト上で購入者がログインし認証される最初の場面では、Amazon Pay プロファイル情報をサイトと共有するための同意を得なければなりません。リダイレクト認証中の場合は、購入者は同意画面でキャンセルボタンをクリックし同意しません。Amazon Pay は購入者をリダイレクト URL に遷移させ、アクセストークンを返す代わりにエラー情報を返します。

アクセスは URL フラグメント内で返され、エラー情報はクエリー文字列経由で返されることに注意してください。リダイレクト URL をレンダリングする前に、エラー情報の文字列が存在するか確認し、もしあれば、購入者をログイン処理が初期化されたページにリダイレクトします。

次のサンプルは、購入者が Amazon Pay プロファイル情報を共有するための同意を得た URL フラグメントを表します。

```
https://www.sample-
store.com/amazonReturn.html#access_token=Atza%7CIQEBLjAsAhQWkVKdr_uRdbW7QpkR
Y8LIxgQIUczBscZcklUALuEd651Nd1_ulkU8WiXq7ZmzcAlp9lZ6Vf_pwaAGL1eVBVKx2x74TmQJ
kdZiaMdVOT99M34V3HvWOtPgWOxUqIgcgBoJ4R1LenaOIg9ZMyU_YO0Ma4Xvsqg7R5fTozyZaVVv
5PwGNRhrQq32RFU7x4Jr6riKsR1AWFoSW3i1arpDCmbF_P3CgVf5X7Y3Dmdkci7JTestAccessTo
ken&token_type=bearer&expires_in=3600&scope=profile+payments%3Awidget
```

次のサンプルは、購入者のプロファイル情報の共有に失敗したクエリー文字列を含むエラー情報を表します。

```
https://www.sample-
store.com/amazonReturn.html?error_description=Access+not+permitted.&error=ac
cess_denied
```

プロファイル情報の取得

購入者のローカルアカウントを生成するために、アクセストークンを認証し、返されたプロファイル情報のハンドリングを行うことで、Amazon アカウントから購入者のプロファイル情報を取得することができます。

プログラミング言語ごとの処理は次のサンプルコードの詳細を確認してください。

Java のサンプルコード
このサンプルコードを利用するためには、Jackson と HttpComponents ライブラリをダウンロードしなければなりません。

```
import com.fasterxml.jackson.core.type.TypeReference;
import com.fasterxml.jackson.databind.ObjectMapper;
import org.apache.http.client.fluent.Content;
import org.apache.http.client.fluent.Request;
import java.net.URLEncoder;
import java.util.Map;
...
// verify that the access token belongs to us
Content c = Request.Get("https://api.sandbox.amazon.com/auth/o2/
  tokeninfo?access_token=" + URLEncoder.encode(access_token, "UTF-8"))
  .execute()
```

```
  .returnContent();
  Map <string><string> m = new ObjectMapper().readValue(
  c.toString(), new TypeReference<map <string><string>>(){});
if (!"YOUR-CLIENT-ID".equals(m.get("aud"))) {
  // the access token does not belong to us
  throw new RuntimeException("Invalid token");
}

// exchange the access token for user profile
c = Request.Get("https://api.sandbox.amazon.com/user/profile")
  .addHeader("Authorization", "bearer " + access_token)
  .execute()
  .returnContent();
m = new ObjectMapper().readValue(
  c.toString(), new TypeReference<map>(){});
System.out.println(
  String.format("%s %s %s", m.get("name"),
  m.get("email"), m.get("user_id")));
```

PHP のサンプルコード

サーバサイドアプリケーション内で、/handle_login.php のリクエストをハンドルし、アクセストークンと Profile REST API を利用してプロファイル情報を取得します。

```
$c = curl_init('https://api.sandbox.amazon.com/auth/o2/
  tokeninfo?access_token='. urlencode($_REQUEST['access_token']));
curl_setopt($c, CURLOPT_RETURNTRANSFER, true);
$r = curl_exec($c);
curl_close($c);
$d = json_decode($r);
if ($d->aud != 'YOUR-CLIENT-ID') {
  // the access token does not belong to us
  header('HTTP/1.1 404 Not Found');
  echo 'Page not found';
  exit;
}

// exchange the access token for user profile
$c = curl_init('https://api.sandbox.amazon.com/user/profile');

curl_setopt($c, CURLOPT_HTTPHEADER, array('Authorization: bearer '
  . $_REQUEST['access_token']));
curl_setopt($c, CURLOPT_RETURNTRANSFER, true);
$r = curl_exec($c);
curl_close($c);
$d = json_decode($r);
echo sprintf('%s %s %s', $d->name, $d->email, $d->user_id);
```

Python のサンプルコード

```
import pycurl
import urllib
import json
```

```
import StringIO

...

b = StringIO.StringIO()

# verify that the access token belongs to us
c = pycurl.Curl()
c.setopt(pycurl.URL,
"https://api.amazon.com/auth/o2/tokeninfo?access_token=" +
urllib.quote_plus(access_token))
c.setopt(pycurl.SSL_VERIFYPEER, 1)
c.setopt(pycurl.WRITEFUNCTION, b.write)

c.perform()
d = json.loads(b.getvalue())

if d['aud'] != 'YOUR-CLIENT-ID' :
    # the access token does not belong to us
    raise BaseException("Invalid Token")

# exchange the access token for user profile
b = StringIO.StringIO()

c = pycurl.Curl()
c.setopt(pycurl.URL, "https://api.amazon.com/user/profile")
c.setopt(pycurl.HTTPHEADER, ["Authorization: bearer " + access_token])
c.setopt(pycurl.SSL_VERIFYPEER, 1)
c.setopt(pycurl.WRITEFUNCTION, b.write)

c.perform()
d = json.loads(b.getvalue())

print "%s %s %s"%(d['name'], d['email'], d['user_id'])
```

Ruby のサンプルコード

```
require "rubygems"
require "net/https"
require "json"
require "uri"

...

# verify that the access token belongs to us
uri = URI.parse("https://api.amazon.com/auth/o2/tokeninfo?access_token=" +
URI.encode(access_token))
req = Net::HTTP::Get.new(uri.request_uri)
http = Net::HTTP.new(uri.host, uri.port)
http.use_ssl = true
http.verify_mode = OpenSSL::SSL::VERIFY_PEER

response = http.request(req)
decode = JSON.parse(response.body)
```

```
if decode['aud'] != 'YOUR-CLIENT-ID'
  # the access token does not belong to us
  raise "Invalid token"
end

# exchange the access token for user profile
uri = URI.parse("https://api.amazon.com/user/profile")
req = Net::HTTP::Get.new(uri.request_uri)
req['Authorization'] = "bearer " + access_token
http = Net::HTTP.new(uri.host, uri.port)
http.use_ssl = true
http.verify_mode = OpenSSL::SSL::VERIFY_PEER

response = http.request(req)
decode = JSON.parse(response.body)

puts sprintf "%s %s %s", decode['name'], decode['email'], decode['user_id']
```

その他のサンプルコード

JavaScript コードサンプル
retrieveProfile API 呼び出しを生成するためにアクセストークンとコールバック関数を利用します。詳しい情報は「Amazon ログイン JavaScript SDK リファレンス」を参照してください。

次のサンプルコードは、Amazon ログイン JavaScript SDK を利用して Amazon の顧客プロファイルをリクエストする方法を表します。

```
amazon.Login.retrieveProfile(response.access_token,function
(profileResponse) {
  // this callback is executed Asynchronously!!
  // check if the profile request was successful
  if(profileResponse.success) {
    // request was successful
    console.log("Buyer Name  = " + profileResponse.profile.Name);
    console.log("Buyer Email = " + profileResponse.profile.PrimaryEmail);
    console.log("Buyer Zip   = " + profileResponse.profile.PostalCode);
    console.log("Amzn Customer ID = " + profileResponse.profile.CustomerId);
    }
});
```

<body>タグの Amazon ログインボタン後にこのコードを配置します。www.example.com を Web サイトのドメインに変更してください。

一度、購入者がログインし、プロファイル情報の共有に同意している現在のウィジェットは与えられた URI にリダイレクトされ、認証レスポンスはクエリー文字列に追加されます。URI は HTTPS プロトコルを使わなければならず、本番環境では、現在のウィンドウは同じドメインである必要があります。

```
<script type="text/javascript">

  document.getElementById('LoginWithAmazon').onclick = function() {
    options = { scope : 'profile' };
    amazon.Login.authorize(options,
      'https://www.example.com/handle_login.php');
    return false;
  };

</script>
```

Web サイトにて、/handle_login.php で Amazon からのレスポンスを制御します。後から選択したいずれかにこの path を変更することができます。

amazon.Login.retrieceProfile 関数は 3 つのパラメータ（success、error、profile）を返します。success は呼び出しが成功されたどうか示します。エラーが発生した場合は、エラーパラメータはエラーメッセージを返します。エラーが無かった場合は、profile は購入者プロファイルを含みます。

アドレス帳とお支払い方法参照ウィジェットの表示

購入者が配送先住所と支払方法を Amazon Pay ウィジェットから選択した後で、購入確認画面で選択されたものを確認するための参照専用の**アドレス帳**と**お支払い方法**ウィジェットを表示することができます。

サイト上でウィジェットを表示するのと同じコードを利用して参照専用のウィジェットを表示します。コードに displayMode: “Read”パラメータを追加します。

```
<!—Place this code in your webpage -->

<div id="readOnlyAddressBookWidgetDiv">
</div>

<div id="readOnlyWalletWidgetDiv">
</div>

<script>
  new OffAmazonPayments.Widgets.AddressBook({
    sellerId: 'YOUR_SELLER_ID_HERE',
    amazonOrderReferenceId: amazonOrderReferenceId,
    // amazonOrderReferenceId obtained from Address widget
    displayMode: "Read",
    design: {
      designMode: 'responsive'
    },
    onError: function(error) {
      // your error handling code
```

```
    }
  }).bind("readOnlyAddressBookWidgetDiv");
</script>

<script>
  new OffAmazonPayments.Widgets.Wallet({
    sellerId: 'YOUR_SELLER_ID_HERE',
    amazonOrderReferenceId: amazonOrderReferenceId,
    // amazonOrderReferenceId obtained from Address widget
    displayMode: "Read",
    design: {
      designMode: 'responsive'
    },
    onError: function(error) {
     // your error handling code
    }
  }).bind("readOnlyWalletWidgetDiv");
</script>
```

選択可能な**アドレス帳**と**お支払い方法**ウィジェットと同じページ上に参照専用ウィジェットを表示する必要がある場合は、onAddressSelect か onPaymentSelect 通知を記載し、その時に参照専用のウィジェットを再表示します。さもなければ、参照専用ウィジェットは正しく表示しません。

アドレス帳ウィジェット無しのインテグレーション

デジタルコンテンツの販売、または、Web サイトで購入者の住所を入力させたいケースがあり、**アドレス帳**ウィジェットから購入者の住所情報を収集したくない場合があります。

注意：購入者の住所情報を収集するのに Amazon の**アドレス帳**ウィジェットを利用しない場合は、トランザクションは Amazon Pay 支払保護ポリシーに適合されません。詳しい情報は Amazon Pay 支払保護ポリシーを参照してください。

アドレス帳ウィジェットが表示されたときに Order Reference が生成されます。**アドレス帳**ウィジェット無しでインテグレーションした場合は、**お支払い方法**ウィジェット呼び出し内のパラメータに onOrderReferenceCreate コールバック関数を指定して、Order Reference を生成する必要があります。

ウィジェットが表示されたときに、onOrderReferenceCreate 関数が Order Reference オブジェクトを呼び出します。Order Reference オブジェクトの getAmazonOrderReferenceId() 関数を呼び出すことで、**orderReferenceId** を得ることができます。以降のトランザクション処理を行うために、この Order Reference の番号を保存しなければなりません。

次のサンプルは、**アドレス帳**ウィジェット無しで**お支払い方法**ウィジェットを表示する方法です。

```
<script>
  var orderReferenceId = null;

  new OffAmazonPayments.Widgets.Wallet({
    sellerId: 'YOUR_SELLER_ID_HERE',
    // Add the onOrderReferenceCreate function to
    // generate an Order Reference ID.
    onOrderReferenceCreate: function(orderReference) {
      // Use the following cod to get the generated Order Reference ID.
      orderReferenceId = orderReference.getAmazonOrderReferenceId();
    },
    design: {
      designMode: 'responsive'
    },
    onPaymentSelect: function(orderReference) {
      // Replace this code with the action that you want to perform
      // after the payment method is selected.

      // Ideally this would enable the next action for the buyer
      // such as a "Continue" or "Place Order" button.
    },
    onError: function(error) {
      // Your error handling code.
      // During development you can use the following
      // code to view error messages:
      // console.log(error.getErrorCode() + ': ' + error.getErrorMessage());
      // See "Handling Errors" for more information.
    }
  }).bind("walletWidgetDiv");
</script>
```

width と height を指定しなければ、このウィジェットは表示されません。詳しい情報は、ウィジェットの「Amazon Pay ウィジェット」を参照してください。

トランザクションフローの管理

購入者にとって、アドレス帳とお支払い方法ウィジェットはお互いに影響します。「購入する」または「続ける」ボタンなどを制御して購入フローを進めなくすることは正しいことです。

- 購入者が住所と支払を選択するまで。
- 購入者が配送先住所を変更してから再度支払方法を選択するまで。
- Order Reference オブジェクトに問題がなくなるまで。

お支払い方法ウィジェットの再レンダリング

多くの場合では、購入者がアドレス帳ウィジェットを選択した場合は自動的にお支払い方法ウィジェットはリフレッシュされます。しかしながら、Web サイトのパフォーマンスや購入者の問題によってお支払い方法ウィジェットの再レンダリングは影響されます。

購入処理中にいくつかの制約が返された場合に、お支払い方法ウィジェットの再レンダリングが本当に要求されることがあります。次はそれらの例です。

- PaymenetMethodNotAllowed – Amazon Pay は与えられたトランザクションのための全ての支払方法について許可しませんでした。SetOrderReferenceDetails か getOrderRefereneDetails 処理の呼び出した後で、レスポンス内の **PaymentMethodNotAllowed** 制約を確認できます。Order Reference の確認を試みることで、制約はエラーコード **ConstraintsExist** を返されます。
- InvalidPaymentMethod – 購入者の支払方法に問題がある場合は失敗します。よって、GetOrderReferenceDetails API 呼び出しを行った後で、**InvalidPaymentMethod** の理由コードになっている **Declined** 状態のレスポンスを確認します。

ベストプラクティスは、**お支払い方法**ウィジェットを再レンダリングし、購入者に異なる支払方法を選択してもらうべきです。

制約についての詳しい情報は、Amazon Pay API リファレンスガイドの OrderReference 制約を参照してください。

処理内容

アドレス帳ウィジェットから生成された Order Reference ID に結びつけてお支払い方法ウィジェットを再レンダリングするためには、次のサンプルの通りになります。

```
<!-- Place this code in your HTML where you would like the
     Wallet widget to appear. -- >
<div id="walletWidgetDiv"></div>

<script>
  new OffAmazonPayments.Widgets.Wallet({
    sellerId: 'YOUR_SELLER_ID_HERE',

    // Tie the order reference Id obtained
    // from the addressBook Widget
    amazonOrderReferenceId: ADDRESSBOOK_ORDER_REFERENCE_ID,

    onPaymentSelect: function(orderReference) {
      // Replace this code with the action you want
      // to perform after the payment method is selected.
    },
    design: {
```

```
      designMode: 'responsive'
    },

    onError: function(error) {
      // Your error handling code.
      // During development you can use the following
      // code to view error messages:
      // console.log(error.getErrorCode()
      //   + ': '
      //   + error.getErrorMessage());
      // See "Handling Errors" for more information.
    }
  }).bind("walletWidgetDiv");
</script>
```

最後に、お支払いウィジェットをレンダリングするために、ページをリロードするか、Javascript 関数を呼び出します。

売上請求（Capture）とオーソリ（Authorize）での即時請求

オーソリリクエスト内の CaptureNow 要素に **true** の値をセットした場合は、オーソリ（Authorize）API は呼び出しは Amazon に対して売上請求（Capture）呼び出しを実行します。これは次のステップである売上請求（Capture）API 呼び出しの必要がなく、注文の出荷を待つ必要がありませんが、物品を配送する場合は、Amazon Pay ポリシーで出荷が完了するまで請求しないことになっております。受注生産の商品を販売する以外では、出荷するまで売上請求（Capture）するべきではありません。

Amazon Pay でのゲスト購入

Amazon Pay（ゲスト購入）オプションを利用している場合は、ComfirmOrderReference 処理を呼び出して Order Reference オブジェクトを承認（Confirm）するまで部分的な配送先住所のみ表示されます。この部分的な住所を利用して配送料を計算することができます。

ConfirmOrderReference 処理を呼び出して Order Reference オブジェクトを確認（Confirm）した後は、GetOrderReferenceDetails 処理を呼び出して完全な配送先住所を取得できます。配送の住所を精査する必要があります。

Amazon Pay のゲスト購入オプションを利用している場合は、Order Reference オブジェクトが承認（Confirm）された後で次の顧客情報が有効になります。

購入者詳細	取得方法
購入者名	ConfirmOrderReference 処理の呼び出しに成功した後で、GetOrderReferenceDetails 処理を呼び出します。
購入者メールアドレス	

購入者電話番号（利用可能な場合のみ）	
配送先名	
配送先住所	**アドレス帳**ウィジェット内の住所が選択された後すぐに、GetOrderReferenceDetails 処理を呼び出すことで、部分的な住所情報（**都道府県、郵便番号、国コード**）を取得できます。ConfirmOrderReferene API の呼び出しに成功した後に、GetOrderReferenceDetails 処理を呼び出すことで完全な配送先住所（住所詳細情報）を取得できます。
配送先電話番号	ConfirmOrderReference 処理の呼び出しに成功した後で、GetOrderReferenceDetails 処理を呼び出します。

オーソリ 30 日後の売上請求（Capture）

オーソリされてから 30 日以内に売上請求しなかった場合は、Amazon によって、オーソリステータスは自動的に **Closed** に変更されます。もはやこのオーソリから費用を売上請求できません。

Order Reference オブジェクトが **Closed** ではない場合は、別のオーソリ（Authorize）呼び出しを実行することができます。しかしながら、Order Reference オブジェクトが Closed の場合は、購入者に問合せする必要があり、請求ミスがあることを説明し、新しい支払で再度注文して頂けるか尋ねる必要があります。もしくは、Amazon Pay 以外の直接支払いなど色々な手段を使うことができます。

Order Reference オブジェクトがまだ **Open** 状態の場合は、再オーソリは普通に利用できます。最初のオーソリとは異なる **AuthorizationReferenceId** を利用する必要があります。

売上請求（Capture）状態の詳しい情報は、「Amazon Pay API リファレンスガイドの売上請求（Capture）と理由コード」を参照してください。

決済とトランザクションレポート

Amazon Pay では、決済レポートとトランザクションレポートの 2 種類のレポートを提供しており、ビジネス処理をサポートしています。

決済レポート（Settlement Reports）

決済レポートは決済期間でのセラーアカウントの活動について詳細までブレークダウンした内容と費用の支払情報を提供しています。経理上の入金消し込みなどにこのレポートは役立ちます。

トランザクションレポート（Transaction Reports）
トランザクションレポートはレポート対象期間中に状態が生成や変更になった Amazon Pay のオブジェクトのリストを提供します。このレポートを利用して、システムでの Amazon Pay オブジェクトの状態を同期させたり、返金率の計算、オーソリ失敗などのビジネス分析に利用することができます。

決済とトランザクションレポートの詳しい情報は、「Amazon Pay 決済とトランザクションレポート」を参照してください。

追加トランザクション

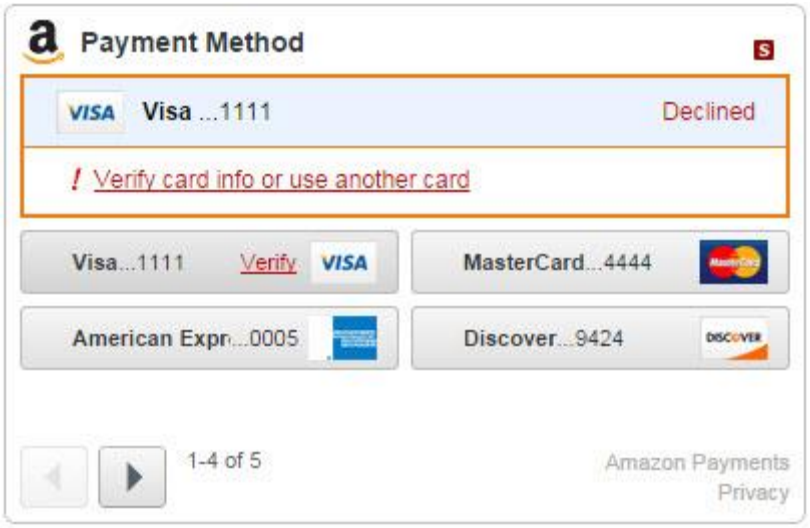
オーソリ失敗のハンドリング

オーソリ処理の呼び出しが失敗した場合は、レスポンス内に失敗した理由コードを確認できます。以下のリストでは理由コードとそれに対応する推奨方法を説明します。

同期モード

InvalidPaymentMethod
InvalidPaymentMethod は選択された支払方法に問題があることを示します。このシチュエーションでは、Order Reference は **Suspended** に遷移し、インスタント支払通知（IPN）の設定がされている場合はそれを発行します。（注意：SoftDecline パラメータは IPN のオーソリ通知（AuthorizationNotification）内でも示されます。）レスポンスパラメータに SoftDecline が Soft または Hard Decline なのか確認します。Amazon では次の対応方法を推奨します。

- SoftDecline パラメータが **true** の場合は、追加のオーソリ（Authorize）を少し経ってから発行できます。リトライした結果が成功ではなかった場合は、購入者に https://payments.amazon.co.jp ページにログインして支払方法の更新を行って頂くように案内します。
- トランザクション（オーソリ）失敗が Soft Decline ではなかった場合は、SoftDecline パラメータは **false** になります。Web サイトの購入処理フロー中に購入者へ選択された支払方法は失敗しましたとエラー通知を表示し、Web サイト上で支払方法を更新するように要求します。
- Web ページ上で Amazon Pay のお支払い方法ウィジェットを再表示します。（サンプルコードは「Step 4：アドレス帳とお支払い方法ウィジェットの追加）を参照してください。」
  レンダリングするときに、現在の Order Reference ID をお支払い方法ウィジェットの amazonOrderReferenceId 値にセットすることを確実にしてください。その後で、Amazon Pay は前回選択した支払方法に **Declined** とマークします。購入者が支払方法の更新をするための **Declined** リンクをクリックすること、または、ウィジェット内のその他の支払方法を選択することができることをメッセージで提供することができます。



上記のスクリーンショットは、お支払い方法ウィジェットが失敗情報をレンダリングする場合に、購入者へ表示される失敗メッセージのサンプルです。

- 購入者が支払方法の更新、または、新しい支払方法を選択した後は、Amazon Pay のウィジェットから onPaymentSelect コールバックをトリガーします。
- 購入者は完了するまでに何回か支払方法をクリックするかもしれません。購入者がいろいろなユーザーエクスペリエンスフローで「**買い物をする**」、「**更新する**」、または、よく似たボタンなどをクリックして最終的に選択した場合は、ConfirmOrderReference API を呼び出します。これによって、Order Reference は **Suspened** から **Open** 状態に遷移します。**Draft** 状態ではない Order Reference オブジェクトに対して、SetOrderReferenceDetails 処理を呼び出せません、また、追加属性をセットできません。ステータスが変更されたことを含むインスタント支払通知（IPN）を受け取ります、または、Order Reference ステータスが更新された確認するために GetOrderReferenceDetails を呼び出すこともできます。
- Order Reference が **Open** 状態である場合は、オーソリ（Authorize）API を利用して新しいオーソリを試みてください。購入者には「提供可能な購入者が受け取るメールコンテンツ」と同様のメールを受信します。

AmazonRejected

AmazonRejected は Amazon によってオーソリ処理が失敗として決定されたことを示します。Order Reference は Closed 状態に遷移し、トランザクションを失敗させる必要があります。

ProcessingFailure

ProcessingFailure は内部処理の結果として Amazon がトランザクションを処理できなかったことを示します。１、２分後にリクエストをリトライしてください。Order Reference が **Open** 状態のままで残ります。リトライが成功しなかった場合は、数分後に注文してくださいと案内してください。

TransactionTimeOut
TransactionTimeOut は 13 秒以内に Amazon がリクエストを処理できず、オーソリリクエストが失敗したことを示します。
一般的に購入者は新しい注文を実行するか、支払方法を変更してもこの失敗した理由を解決できません。購入者のために（Order Reference が Open であれば）非同期モードでリトライするか、トランザクションをキャンセルできます。

非同期モード

InvalidPaymentMethod
InvalidPaymentMethod は選択された支払方法に問題があることを示します。このシチュエーションでは、Order Reference は **Suspended** に遷移し、インスタント支払通知（IPN）の設定がされている場合はそれを発行します。（注意：SoftDecline パラメータは IPN のオーソリ通知（AuthorizationNotification）内でも示されます。）レスポンスパラメータに SoftDecline が Soft か Hard Decline なのか確認します。購入者は既に Web サイトには残っていないと想定して Amazon は次の通りの対応方法を推奨します。

- SoftDecline パラメータが **true** の場合は、追加のオーソリ（Authorize）を少し経ってから発行できます。リトライした結果が成功ではなかった場合は、購入者に次の方法を説明して、支払方法の更新を行って頂くように案内します。
- トランザクション（オーソリ）失敗が Soft Decline ではなかった場合は、SoftDecline パラメータは **false** になります。購入者にメールか電話で問合せし、支払方法の更新を案内します。購入者がログインして直接注文と支払方法が変更できる URL を含めたメール送信します。URL の形式は次の通りになります。

```
https://payments.amazon.com/jr/your-
account/apa?action=renderOrderDetails&contractId=<order reference
number>
```

以下はサンプルの URL です。

```
https://payments.amazon.com/jr/your-
account/apa?action=renderOrderDetails&contractId=P01-99999-9999999
```

その他の方法として、Web ページ上で購入者に上記の方法と同様に注文を確認して、支払方法を変更する案内を提示することもできます。

- 購入者が支払方法を更新した後は、Order Reference オブジェクトは **Suspended** 状態から **Open** 状態に遷移します。Amazon Pay はそれからインスタント支払通知（IPN）を送信し、

購入者には「提供可能な購入者が受け取るメールコンテンツ」と同様のメールを受信します。

AmazonRejected
AmazonRejected は Amazon によってオーソリ処理が失敗として決定されたことを示します。Order Reference は Closed 状態に遷移し、トランザクションを失敗させる必要があります。

ProcessingFailure
ProcessingFailure は内部処理の結果として Amazon がトランザクションを処理できなかったことを示します。１、２分後にリクエストをリトライしてください。Order Reference が **Open** 状態のままで残ります。リトライが成功しなかった場合は、数分後に注文してくださいと案内してください。

TransactionTimeOut
TransactionTimeOut はオーソリ処理の呼び出しがデフォルトタイムアウトの 24 時間か transactionTimeout リクエストパラメータで指定された時間内に処理されなかったことを示します。このコードの結果として高い失敗が発生した場合は、タイムアウト値を調整してトライしてください。

トランザクションが非同期モードでリトライすることは重要であり、購入者は注文が通常処理よりも長い時間掛かっていることを理解しますが、購入者は再注文を試みるべきではありません。

Soft Decline ハンドリングの代表的なフロー

顧客注文に対してオーソリ（Authorize）API を呼び出して、SoftDecline パラメータに **true** がセットされた InvalidPaymentMethod の理由コードを受け取った場合は、再オーソリのオプションがあります。
最初に、再オーソリする前に、Order Reference を **Open** 状態にリセットする必要があります。それには、ConfirmOrderReference API を呼び出します。
それを実行し Order Reference オブジェクトが **Open** 状態になれば、オーソリ（Authorize） API を再度呼び出します。
必要であれば、最初のオーソリに失敗してからこの処理は３回まで繰返します。

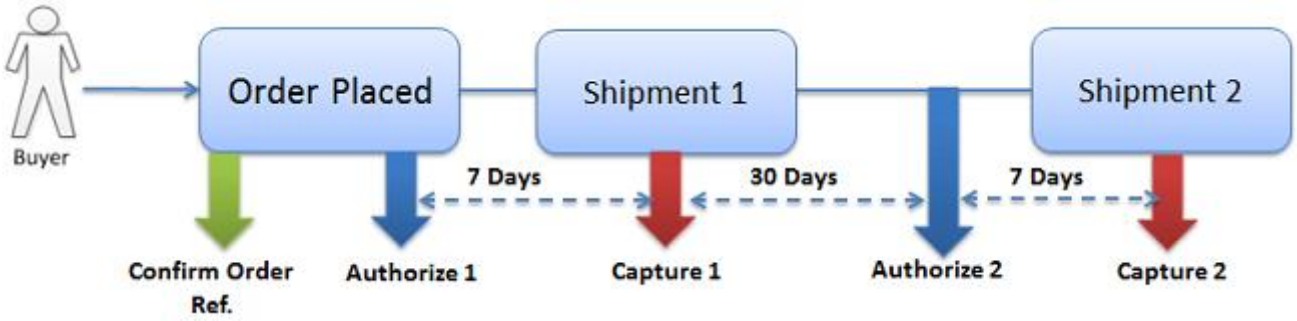
複雑な支払シナリオのハンドリング

このセクションでは、これまでよりもより複雑な支払シナリオのハンドリング方法を説明します。

支払を分割

支払分割のシナリオは、購入者は支払合計金額を複数の個々の支払を分割して広げます。例えば、受注生産の商品を提供している場合では、前金で部分支払を得て、処理が完了してから残りの支払を行うかもしれません。

よくある分割支払いのケースとしては、購入者の注文が複数の配送先である場合に、それぞれの配送ごとに売上を分割することがあります。この支払モデルは、次の図の通りに Amazon Pay ではサポートしています。



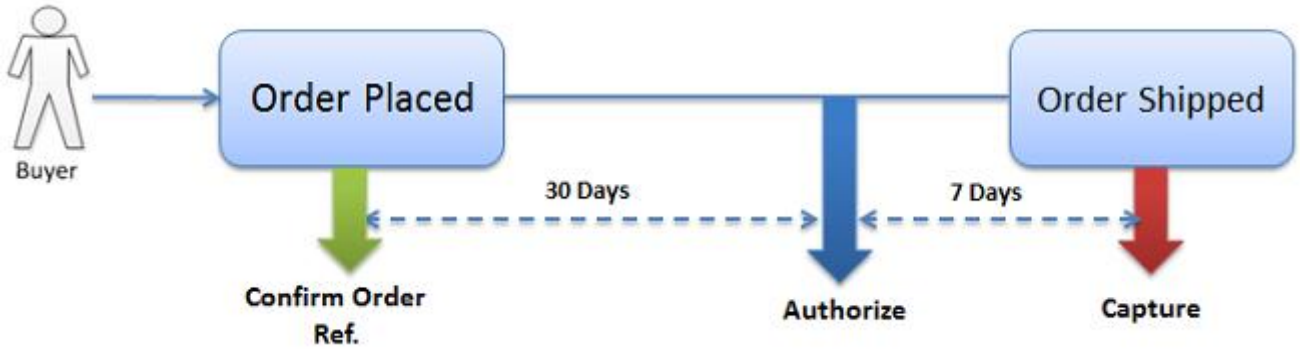
この例では、Order Reference オブジェクトはその時に確認されており、最初の出荷は 7 日までに行われています。最初の売上請求は最初の出荷時にリクエストできます。30 日経過後、オーソリ処理を再度呼び出します。最後に、購入者へ 2 回目の出荷を行ってから 2 回目の売上請求処理を呼び出します。

Amazon Pay では、Order Reference オブジェクトに対して最大 10 個のオーソリリクエストすることが許可されており、さらに、この分割支払はサポートされており、運用ルールに応じてこれらの支払方法を許可しております。**Declined** または **Closed** のオーソリはこの制限の数に含まれません。

延期された支払

延期された支払のシナリオは、購入者が EC サイトで注文をしましたが、30 日間までに売上請求（Capture）をしなかった場合です。例えば、予約注文を購入者に許可している、または、入荷待ち商品の場合は、その商品が出荷されるまで支払を行うことができない運用ルールかもしれません。

入荷待ち商品が出荷されてから支払を行う会社のルールであるならば、商品が出荷可能になったが、オーソリ呼び出してから 30 日を過ぎてしまい、売上請求処理の実行ができません。この支払が延期された支払シナリオは、次の図に似た状況です。

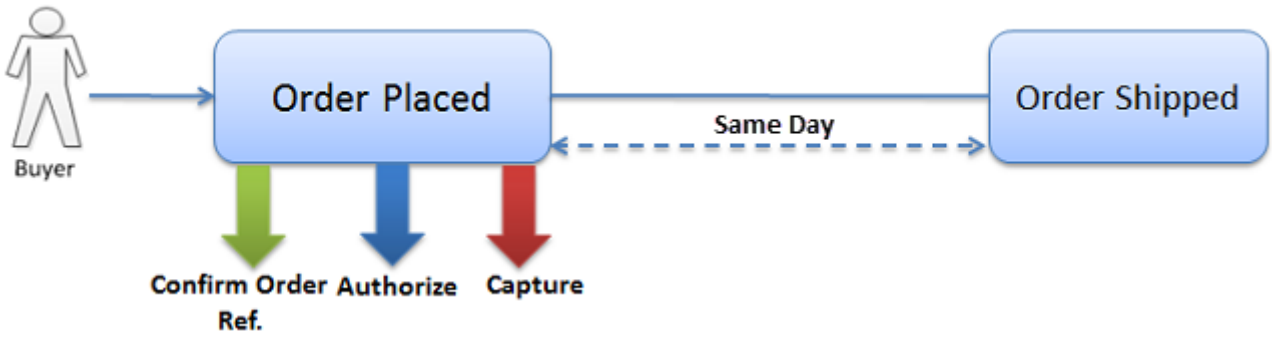


この図では注文が行われ、Order Reference オブジェクトが確認されましたが、オーソリ処理は商品の在庫が有効になり、出荷準備になるまで呼び出されません。売上請求処理の呼び出しは、予約注文の商品が購入者に出荷されてから実行されます。

Amazon Pay では、Order Reference オブジェクトを 180 日間保持するので、これらのシナリオはサポートします。この 180 日間は購入者の注文は有効ですが、出荷の準備ができるまではオーソリ処理を実行してはいけません、また、売上請求も実行してはいけません。

即時請求

購入者が注文を行い、Amazon に Order Reference を確認した後に購入者に即時に請求するには、すぐにオーソリ処理を呼び出すことができ、オーソリオブジェクトが **Open** 状態に遷移した後に、売上請求処理を呼び出します。または、オーソリ処理で **CaptureNow** リクエストパラメータを **true** にセットして呼び出すことができます。このアプローチは、例えば同日に注文商品を出荷することができる場合に利用できます。この支払シナリオは、次の図に似た状況です。



この図は購入者が注文を確認してからすぐに請求し、Order Reference は確認されていることを表しています。商品は即時に顧客に出荷されています。

注意：Amazon Pay のポリシーでは、商品が出荷されてから顧客に請求することになっています。注文を出荷する前に代金を回収してはなりません。

注文完了後の支払変更のハンドリング

Amazon Pay では、顧客が注文処理が完了してから登録されている注文を変更するいくつかの方法が許可されています。この変更のシナリオには以下の方法が Amazon Pay でサポートされています。

最初の注文金額以上の請求

いくつかのシチュエーションとして、Web サイトで注文された合計金額を減額や増額したい場合があります。例えば、購入者がカスタマーサービスに連絡して、配送オプションをアップグレードを要求するかもしれません。

配送料や注文変更などで購入者が最初に注文した金額に追加して請求したいかもしません。この追加できる金額は、最初に注文された金額の 15%または 8,400 円のどちらか低い額までです。例えば、最初の注文が 10,000 円の場合は 1,500 円まで追加請求できます。配送料などで金額が増えることが想定される場合は注文が完了される前に購入者にそのことを表示しなければなりません。

サンプルシナリオ – 支払金額を増やす

この支払を増やすシナリオを説明するためには、購入者が購入処理に入り、配送料込で 20,000 円の注文を確認していることを想定します。注文が出荷される前に、購入者がカスタマーサービスに連絡して、配送オプションを高額なものにアップグレード要求して、合計金額が 23,000 円になります。

このシナリオでは、23,000 円のオーソリ処理を呼び出すか、最初の 20,000 円のオーソリをしている場合は 3,000 円のオーソリ処理を呼び出します。追加の 3,000 円は最初の Order Reference オブジェクトの金額の 15%以下であるので、この 23,000 円の請求は許可されます。

注意：チャージバックと Amazon マーケットプレイス保証の可能性を削減するために最初の注文金額より高い金額を請求するよりも購入者からのオーソリを保証することを Amazon は推奨します。購入者に注文情報が変更されたことをメールすることと、それを保管することも Amazon は推奨します。

配送先住所の変更

購入者が問合せし、注文時に使われた Amazon Pay の**アドレス帳**ウィジェットの住所と異なる配送先住所への要求は Amazon に通知する必要はありません。Amazon へ変更されたことを通知なしで注文出荷処理で住所を変更することはできます。

注意：Amazon の**アドレス帳**ウィジェットを経由しない配送先住所を利用する場合は、Amazon Pay 支払保護ポリシーに準拠されません。詳しい情報は利用規約を参照してください。

注文のキャンセル

会社の会計やその他の内部処理での注文ステータスを追跡をサポートするために、Order Reference オブジェクトのキャンセルが必要かもしれません。Amazon Pay では Order Reference オブジェクトのステータスをキャンセルするために CancelOrderReference API を呼び出すことが可能です。

Order Reference オブジェクトが **Draft**、**Open**、**Suspended** 状態であり、Capture オブジェクトが **Pending**、**Completed**、**Closed** でないものに対して Order Reference オブジェクトを **Canceled** にするために CancelOrderReference API を呼べます。すべての注文をキャンセルする場合、または、代金を請求したくない場合にこれを実行します。Amazon は購入者にこの注文はキャンセルされた支払であることを通知します。CancelOrderReference 処理を呼び出すことで Order Reference オブジェクトはキャンセルされます。

- Order Reference オブジェクトから新しいオーソリを生成できず、**Open** または **Pending** 状態のオーソリは **Closed** されます。
- **Completed**、**Closed**、**Pending** 状態の Capture オブジェクトが無いものに対してのみ Order Reference オブジェクトをキャンセルできます。

詳しい情報は Amazon Pay API リファレンスガイドの CloseOrderReference を参照してください。

部分的な注文の実行

注文の部分的な実行のみの場合は、充当する費用を請求した後に、CloseOrderReference 処理を呼び出して Order Reference を **Closed** にすることができます。さらに、CloseAuthorization 処理を利用することで、未回収のオーソリを Close することができます。Order Reference とオーソリを **Closed** にすることで、内部の注文ステータスを Amazon と同期するが可能です。

次のサンプルは CloseAuthorization 処理を呼び出す方法を表します。

```
https://mws.amazonservices.com/OffAmazonPayments_Sandbox/2013-01-01
?AWSAccessKeyId=AKIAFBM3LG5JEEXAMPLE
&Action=CloseAuthorization
&AmazonAuthorizationId=S23-1234567-1234567-0000001
&ClosureReason=Closing%20the%20auhorization%20transaction
&SellerId=YOUR_SELLER_ID_HERE
&SignatureMethod=HmacSHA256
&SignatureVersion=2
&Timestamp=2013-12-19T19%3A01%3A11Z
&Version=2013-01-01
&Signature=WlQ708aqyHXMkoUBk69Hjxj8qdh3aDcqpY71hVgEXAMPLE
```

CloseAuthorization 処理についての詳しい情報、含めるリクエストパラメータ、レスポンス要素は、Amazon Pay API リファレンスガイドの CloseAuthorization を参照してください。

注意：注文の支払が処理できた後は Order Reference を **Closed** にすることを Amazon は推奨します。別な方法では、180 日間 **Open** 状態の Order Reference は自動的に **Closed** になり、購入者 Amazon Pay の Web サイトでトランザクションを確認できます。しかし、注文が完全に終了するまでは **Closed** ではなく **Open** 状態にしておくことも重要です。

SANDBOX 環境でのインテグレーションテスト

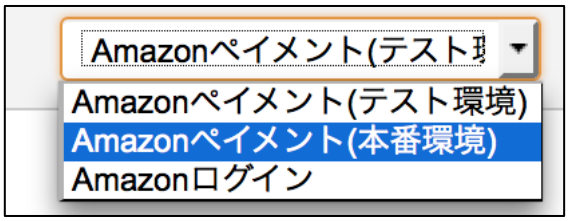
Amazon Pay では、本番環境に移行する前に徹底的にテストすることができます。SANDBOX モードでインプリメントすることで、Web サイト上で Amazon Pay のウィジェットを利用したナビゲートなどの購入者エクスペリエンスをシミュレートすることができます。

SANDBOX モードでは、Amazon への API 呼び出し設定が正しく確実にできるのかテストでき、登録された注文をトラックするに必要な全ての Amazon Pay パラメータを含むレスポンスもテストできます。購入処理中の良くないイベントである購入者エクスペリエンスを事前に確認するためのエラーシミュレートはインテグレーションに大いに役立ちます。

Amazon Pay SANDBOX のテストアカウント設定

インテグレーションテスト中にたくさんの支払シナリオをシミュレートするために、複数の購入者アカウントを設定することを勧めます。次はテストアカウントをセットアップする方法です。

1. https:// sellercentral.amazon.co.jp でセラーセントラルにログインします。
2. マーケットプレイススイッチャーのドロップダウンボックスより **Amazon Pay（テスト環境）**を選択します。**マーケットプレイススイッチャー**は画面上部メニューの中央部にドロップダウンボックスとして表示されます。

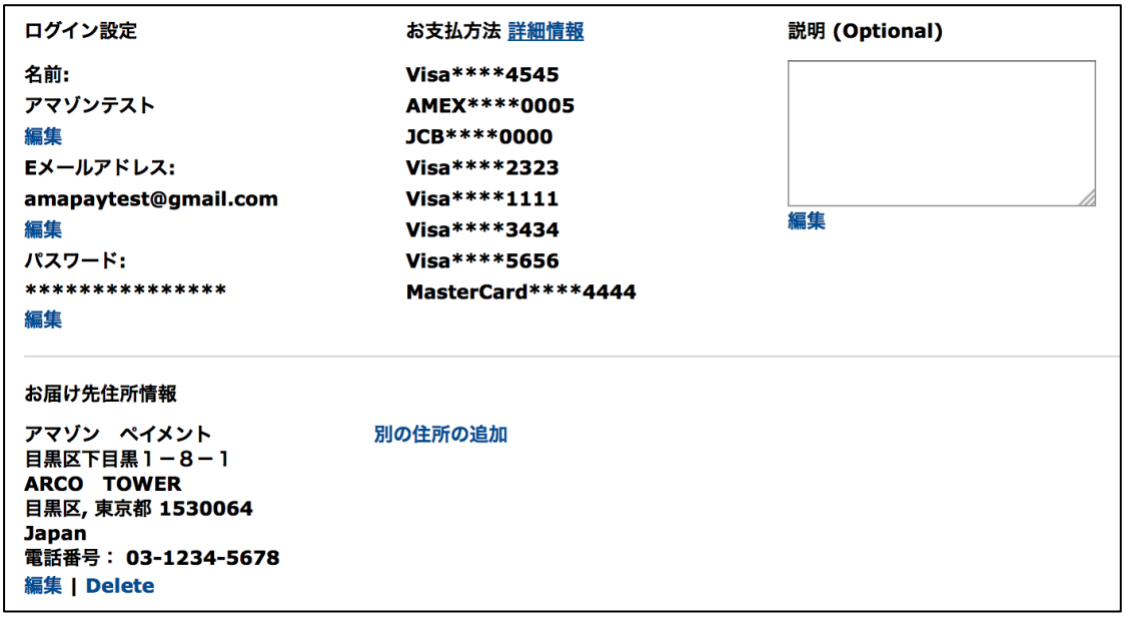


画面が小さい場合は**マーケットプレイススイッチャー**のドロップダウンボックスはアイコンに置き換わります。

3. **インテグレーション**メニューの**テストアカウント**をクリックします。
4. 下記の説明の通りに各項目に情報を入力します。

- **ログイン設定** – このログインは SANDBOX のために利用します。本番環境のアカウントとは異なるメールアドレスとパスワードが必要です。
  - **E メールアドレス** – 本番環境で利用しているアカウントのメールアドレスを利用したい場合は、@マークの前に”+sandbox”を追加して利用することができます。例えば、taro@example.com の場合では、taro+sandbox@examlple.com などです。個別にテストアカウントを作成できる場合はその限りではありません。
  - **パスワード** – 本番環境で使わているパスワードと異なるものを利用してください。
- **お支払方法** – SANDBOX 環境で利用できるテスト用のクレジットカードです。請求されることはありません。
- **お届け先住所情報** – 配送先住所を選択または追加します。

5. 「**アカウントの作成**」ボタンをクリックすることでテストアカウントが生成されます。



注意：当テストアカウントは US 環境をそのままコピーして利用しておりますので、マルチバイトの文字列では一部エンコーディングされない可能性がありますので、短い日本語にされるかシングルバイトの文字列をご利用ください。

サンプルの住所と支払方法

サンプルの配送先住所と支払方法は SANDBOX 環境に予め用意されておりますが、シングルバイト情報になりますので日本語環境用として新たに作成することを強くお薦めします。

サンプルの配送先住所

新規にテストアカウントを作成した場合は２つの住所が設定されます。この住所情報はシングルバイトデータになりますので、日本語環境ではほとんど利用されることはありませんので、「別の住所の追加」リンクより新しく日本語での住所を作成してください。

注意：この SANDBOX 環境は US 仕様の項目になっておりますので、一部日本の本番環境と項目が異なります。例えば、City 情報が SANDBOX 環境のテストアカウントにありますが、日本の本番環境にはありません。逆に Address Line3 は日本の本番環境では会社名として利用されていますが、SANDBOX 環境には同項目がありません。本番環境に移行した場合はこの点を考慮してください。

サンプルの支払方法

テストアカウントを作成した場合は、事前にテスト用のクレジットカード情報が下記の通りに生成されます。お支払い方法ウィジェットで表示されますのでテストで利用してください。なお、クレジットカード情報の登録はできません。また、エラーシミュレーションができるカードもあります、本番環境に移行する前に徹底したテストを行うことが可能です。

Visa****1111
Master****4444
AMEX****0005
JCB****0000
Visa****5656（**オーソリタイムアウト**を発生させます）
Visa****4545（**PaymentMethodNotAllowed** エラーを発生させます）
Visa****2323（**AmazonRejected** のオーソリエラーを発生させます）
Visa****3434（**InvalidPaymentMethod** のオーソリエラーを発生させます）

SANDBOX と本番環境の相違点

Amazon Pay では、利用される様々なオブジェクトが生成された後のオブジェクトの有効期限などのルールを運用ルールに合わせて全て管理します。Amazon Pay の SANDBOX のテストを容易にするために次のルールが調整されています。

オブジェクト	本番環境ビジネスルール	SANDBOX ルール
Order Reference オブジェクトの Open 状態	180 日間	180 日間

オーソリオブジェクトの Open 状態	Closed になるまで 30 日間	Closed になりまで 2 日間
オーソリに対しての売上請求可能な有効期限	30 日間	2 日間

SANDBOX シミュレーション

Amazon Pay は、シミュレーション指定した支払シナリオを利用することができます。これらのシミュレーションは運用ルールをテストするためのレスポンスを生成できます。次の表はレスポンスと状態遷移をシミュレートすることができることを説明します。

状態と理由コードの詳しい情報は、Amazon Pay API リファレンスガイドの状態と理由コードを参照してください。

注意：**状態**と**理由コード**はシミュレーション文字列で利用でき、希望した状態と理由コードを Amazon Pay API セクション処理での呼び出しで指定しなければなりません。不正な組み合わせは **InvalidSandboxSimulationSpecified** エラーが返されます。

Order Reference オブジェクト			
状態	**理由コード**	**シミュレーション文字列**	**SANDBOX でのシミュレート方法**
Draft		N/A	SANDBOX の Amazon アカウントでお支払いボタンをクリックしテストアカウントでログインして認証します。Order Reference オブジェクトは **Draft** 状態で生成します。
Open		N/A	**Draft** の Order Reference オブジェクトを ConfirmOrderReference 処理を呼び出して確認します。
Suspended	InvalidPaymentMethod	N/A	オーソリオブジェクトが **InvalidPaymentMethod** の理由コードで **Declined** 状態に遷移した時に、Order Reference

			オブジェクトは **Suspended** 状態に遷移します。
Canceled	SellerCanceled	N/A	**Open** または **Suspended** の Order Reference オブジェクトに CancelOrderReference 処理を呼び出してキャンセルします。
	Stale	N/A	**Draft** の Order Reference オブジェクトが生成されてから 3 時間以内に確認されなかった。Order Reference オブジェクトは **Stale** 状態に遷移します。
	AmazonCanceled	N/A	シミュレートできません。
Closed	Expired	N/A	Close または Cancel されなかった Order Reference オブジェクトです。**Open** または **Suspended** の Order Reference オブジェクトは生成されてから 180 日後に理由コードと一緒に Amazon によって **Closed** にされます。
	MaxAmountCharged	N/A	Order Reference オブジェクトに 15%か 8,400 円以上の売上請求処理を呼び出した場合です。
	MaxAuthorizationsCaptured	N/A	**Open** の Order Reference オブジェクトに対して完全または部分的に 10 個のオーソリに対して売上請求しました。
	AmazonClosed	{"SandboxSimulation": {"State":"Closed",	**Open** の Order Reference オブジェクトに **ClosureReason** リクエストパラメータに値を指

		"ReasonCode":"AmazonClosed"}}	定して CloseOrderReference 処理を要求します。
	SellerClosed	N/A	**AmazonClosed** 理由コードのシミュレーション文字列を指定せずに CloseOrderReference 処理を **Open** の Order Reference オブジェクトに呼び出して Close にします。

Authorization オブジェクト			
状態	**理由コード**	**シミュレーション文字列**	**SANDBOX でのシミュレート方法**
Pending		N/A	非同期モードでオーソリ処理で呼び出してリクエストします。全てのオーソリオブジェクトはオーソリリクエストが実行されてから 30 秒間は **Pending** 状態です。同期モードではシミュレートできません。
Open		N/A	オーソリ処理を呼び出してリクエストします。非同期モードでは、オーソリオブジェクトは 30 秒間 **Pending** 状態になってから **Open** 状態に遷移します。同期モードでは、オーソリオブジェクトはすぐに **Open** 状態に遷移します。
Declined	InvalidPaymentMethod	{"SandboxSimulation": {"State":"Declined", "ReasonCode":"InvalidPaymentMethod", "PaymentMethodUpdateTimeInMins":5}} Or	オーソリ処理で SellerAuthorizationNote リクエストパラメータにこの値を指定します。Order Reference は **Open** 状態から **Suspended** 状態に遷移します。Order Reference オブジェクトが **Suspended** 状態から **Open** 状態に遷移するための時間（1

		{"SandboxSimulation": {"State":"Declined", "ReasonCode":"InvalidPayment Method", "PaymentMethodUpdateTimeInMins":5, "SoftDecline":"true" }}	分から 240 分）を PaymentMethodUpdateTimeIn Mins パラメータを指定することができます。これは購入者が無効な支払方法を更新することをシミュレートし、Order Reference オブジェクトを **Open** 状態に戻します。 Soft Decline の再処理をシミュレートするためには、シミュレーション文字列に**"SoftDecline":"True"**を利用します。**Suspended** の Order Reference オブジェクトに ConfirmOrderReference 処理を呼び出して、**Open** 状態に遷移することができます。これは販売事業者が **Suspended** の Order Reference オブジェクトをシミュレートし、オーソリ処理をリトライして **Open** 状態に戻すことができます。
	AmazonRejected	{"SandboxSimulation": {"State":"Declined", "ReasonCode":"AmazonRejected"}}	オーソリ処理の **SellerAuthorizationNote** リクエストパラメータにこの値を指定します。
	ProcessingFaiure	N/A	シミュレートできません。
	TransactionTimeOut	{"SandboxSimulation": {"State":"Declined", "ReasonCode":"TransactionTimedOut"}}	オーソリ処理の **SellerAuthorizationNote** リクエストパラメータにこの値を指定します。
Closed	ExpiredUnused	{"SandboxSimulation": {"State":"Closed", "ReasonCode":"ExpiredUnused",	オーソリ処理の **SellerAuthorizationNote** リクエストパラメータにこの値を指定します。オーソリオブジェクトが **Closed** になるために、**ExpirationTimeInMins** パ

		"ExpirationTimeInMins":1}}	ラメータに時間（1 分から 60 分）を指定して利用できます。
	MaxCapturesProcessed	N/A	**Open** のオーソリオブジェクトに売上請求処理を呼び出して売上請求します。
	AmazonClosed	{"SandboxSimulation": {"State":"Closed", "ReasonCode":"AmazonClosed"}}	オーソリ処理の **SellerAuthorizationNote** リクエストパラメータにこの値を指定します。
	OrderReferenceCanceld	N/A	CancelOrderReference 処理を呼び出して Order Reference オブジェクトに関連したオーソリオブジェクトをキャンセルします。
	SellerClosed	N/A	CloseAuthorization 処理を呼び出して **Open** のオーソリオブジェクトを Close します。

Capture オブジェクト			
状態	**理由コード**	**シミュレーション文字列**	**SANDBOX でのシミュレート方法**
Pending		{"SandboxSimulation": {"State":"Pending"} }	売上請求処理の **SellerCaptureNote** リクエストパラメータにこの値を指定します。売上請求オブジェクトはそれから 30 秒間は **Pending** 状態です。
Declined	AmazonRejected	{"SandboxSimulation": {"State":"Declined", "ReasonCode":"AmazonRejected"}}	売上請求処理の **SellerCaptureNote** リクエストパラメータにこの値を指定します。
	ProcesseingFailure	N/A	シミュレートできません。

Completed		N/A	売上請求処理を呼び出して **Open** のオーソリオブジェクトに対して売上請求します。
Closed	MaxAmountRefunded	N/A	返金処理を呼び出して、売上請求額の 15%か 8,400 円以上の返金を行います。
	MaxRefundsProcessed	N/A	**Completed** の売上請求オブジェクトに対して返金処理を 10 回発行します。
	AmazonClosed	{"SandboxSimulation": {"State":"Closed", "ReasonCode":"AmazonClosed"}}	売上請求処理の **SellerCaptureNote** リクエストパラメータにこの値を指定します。

Refund オブジェクト			
状態	**理由コード**	**シミュレーション文字列**	**SANDBOX でのシミュレート方法**
Pending		N/A	返金処理を呼び出して返金をリクエストします。すべての返金オブジェクトは返金要求されてから 30 秒間は **Pending** 状態になります。
Declined	AmazonRejected	{"SandboxSimulation": {"State":"Declined", "ReasonCode":"AmazonRejected"}}	返金処理の **SellerRefundNote** リクエストパラメータにこの値を指定します。
	ProcessingFailure	N/A	シミュレートできません。
Completed		N/A	返金処理を **Completed** の売上請求オブジェクトに対して返金します。

次のサンプルコードは、**SellerAuthorizationNote** リクエストパラメータを利用して、**InvalidPaymentMethod** の理由コードでオーソリが **Declined** 状態をシミュレートする方法です。

```
https://mws.amazonservices.com/OffAmazonPayments_Sandbox/2013-01-01
?AWSAccessKeyId=AKIAFBM3LG5JEEXAMPLE
&Action=Authorize
&AmazonOrderReferenceId=S23-1234567-1234567
&AuthorizationAmount.Amount=50
&AuthorizationAmount.CurrencyCode=USD
&AuthorizationReferenceId=test_authorize_1
&SellerAuthorizationNote=
%7B%22SandboxSimulation%22%3A%7B%22State%22%3A%22Declined%22%2C
%22ReasonCode%22%3A%22InvalidPaymentMethod%22%7D%7D
&SellerId=YOUR_SELLER_ID_HERE
&SignatureMethod=HmacSHA256
&SignatureVersion=2
&Timestamp=2013-10-03T19%3A01%3A11Z
&TransactionTimeout=60
&Version=2013-01-01
&Signature=WlQ708aqyHXMkoUBk69Hjxj8qdh3aDcqpY71hVgEXAMPLE
```

本番環境へインテグレーション切り替え

インテグレーションを SANDBOX 環境から本番環境へ切り替えるためには、次の URL を SANDBOX から本番環境に変更する必要があります。

本番環境
https://mws.amazonservices.com/OffAmazonPayments/2013-01-01/
Widgets.js URL
Amazon Pay のボタンを配置しているそれぞれのページの header へ、Widgets.js のソースを更新します。 **https://static-fe.payments-amazon.com/OffAmazonPayments/jp/js/Widgets.js**
Login API URL
https://api.amazon.com
Profile API URL
https://api.amazon.com/user/profile

コードリファレンス情報

エラーハンドリング

エラーハンドリングの概要

Amazon Pay のウィジェットと Amazon Pay の API 処理の両方はエラーを発行します。Web サイトでの購入者エクスペリエンスを最大限にするためにこの両方からのエラーを取得してハンドリングします。

Amazon Pay API 呼び出しからのエラーハンドリング

Amazon Pay API 呼び出しからエラーを取得したときは、エラーの原因によっては、処理をリトライすることができるかもしれません。以下の表はリトライすることができるエラーのリストです。

ユニークな Reference ID を使って API 呼び出すことで、既に成功したリクエストと同じ処理を呼び出すことができますが、重なるトランザクションは生成されません。新しいリクエストを初期化する変わりに、その呼び出しは最初の呼び出しの結果を返します。

Order Reference リクエストを行うのためにはユニークな Reference ID である **AmazonOrderReferenceId** が必要です。オーソリ、売上請求、返金リクエストのためには、それぞれユニークな ID として **AuthorizationReferenceId**、**CaptureReferenceId**、**RefundReferenceId** が必要です。

例えば、同じ売上請求オブジェクトに対して、同じ **RefundReferenceId** を利用して返金することはできません。この機能は、以下の表で説明されている制約になったエラーについては安全にリトライするが認められています。

エラーを受け取った後で、処理呼び出しをリトライしたい場合は、最初のエラーレスポンス後にすぐにリトライできます。複数回リトライしたい場合は、設定された制限に対して**指数的バックオフ**アプローチのインプリメントを推奨します。それから、エラーを記録し、手動でフォローアップと調査を行います。例えば、1 秒、2 秒、4 秒、10 秒、30 秒と時間の間隔を上げてリトライを実行します。実際のバックオフ時間と制限は運用ルールに合わせます。

次の表はリトライ可能なエラーを説明します。

HTTP コード	HTTP ステータス	説明と対処方法
500	InternalServerError	リクエストを完了する前にサーバは予期せぬ状態になりました。このエラーはリトライしても安全です。 このエラーは本番環境エンドポイントに SANDBOX セッションを要求することでよく起こります。またその逆もあります。
502	Bad gateway	クライアントとサーバの間にあるゲートウェイやプロキシーサーバがリクエストを拒否しました。多くのケースでは、これは断続的に発生しますが、リトライしても安全です。継続的に発生する場合は、ネットワーク管理者に問合せしてください。
503	ServiceUnavailable RequestThrottled	サーバサイドで予期せぬ問題が発生したか、トランザクションの制限を超えたスロットルが発生しました。完全なスロットルの説明は MWS ディベロッパーガイドを参照してください。クライアントはリクエストするか、リクエスト数を減らします。
504	Gateway timeout	クライアントとサーバの間にあるゲートウェイやプロキシーサーバでタイムアウトがリクエストで発生しました。多くのケースでは、これは断続的に発生しますが、リトライしても安全です。

詳しい情報は、Amazon Pay API リファレンスガイドのエラーコードセクションを参照してください。

Amazon Pay ウィジェットのエラーハンドリング

Step 3：ボタンウィジェットの追加で説明した通りにボタン、アドレス帳、お支払い方法ウィジェットはエラー状況を通知するように設定できます。これらのウィジェットは、いくつかのインテグレーションエラーを生成してエラー通知を送ることができます。

次のコードサンプルは、エラーに関連された errorCode と errorMessage を参照する方法を説明します。これらのエラーコードは、素早くインテグレーションのデバッグを支援

し、ロギングしていれば、本番環境サイト上の潜在的な問題を確認することができます。

```
<label>Debug Error Code     :</label>
<div id="errorCode"></div>

<label>Debug Error Message  :</label>
<div id="errorMessage"></div>

<script>
var amazonOrderReferenceId;
new OffAmazonPayments.Widgets.Button ({
  sellerId: 'YOUR_SELLER_ID_HERE',
  onSignIn: function(orderReference) {
    amazonOrderReferenceId = orderReference.getAmazonOrderReferenceId();
    window.location = 'https://yoursite.com/redirect_page?session='
      + amazonOrderReferenceId;
    },
    onError: function(error) {
      document.getElementById("errorCode").innerHTML = error.getErrorCode();
      document.getElementById("errorMessage").innerHTML =
        error.getErrorMessage();
    }
  }).bind("payWithAmazonDiv");
</script>
```

次の表のリストは、Amazon Pay ウィジェットから返されたいくつかのエラーコードとそれに対応するエラーメッセージです。購入者はウィジェット内でこれを確認できます。

エラーコード	説明	購入者に表示されるエラーメッセージ
AddressNotModifiable	Order Reference が与えられた状態の場合は、配送先住所を変更できません。	この注文では配送先住所を変更できません。販売事業者にご連絡ください。
BuyerNotAssociated	購入者は与えられた Order Reference に関連付けされていません。購入者はウィジェットが表示される前にサインインしなければなりません。	このセッションは有効ではありません。Amazon アカウントでお支払いボタンをクリックして再度ご購入してください。
BuyerSessionExpired	購入者セッションは Amazon では期限切れです。購入者はウィジェッ	セッションは期限切れです。Amazon アカウントでお支払いボタンをクリ

	トが表示される前にサインインしなければなりません。	ックして再度ご購入ください。
InvalidAccountStatus	販売事業者アカウントはこのリクエストを実行するために提供されていません。例えば、アカウントは登録がまだ完了していないなどです。	ボタンウィジェットが表示されエラーが発生した場合は、ウィジェットは表示されません。これらは購入者にメッセージを表示しません。アドレス帳ウィジェットやお支払い方法ウィジェットが表示されてエラーが発生した場合は、購入者にこのエラーが表示されます。 申し訳ございません。この Web サイトで問題が発生しました。販売事業者にご連絡ください。
InvalidOrderReferenceId	指定された Order Reference の ID は無効です。	申し訳ございません。この Web サイトで問題が発生しました。販売事業者にご連絡ください。
InvalidParameterValue	指定されたパラメータに割り当てられた値は有効ではありません。	ボタンウィジェットが表示されエラーが発生した場合は、ウィジェットは表示されません。これらは購入者にメッセージを表示しません。アドレス帳ウィジェットやお支払い方法ウィジェットが表示されてエラーが発生した場合は、購入者にこのエラーが表示されます。 申し訳ございません。この Web サイトで問題が発生しました。販売事業者にご連絡ください。

InvalidSellerId	提供した販売事業者の ID は無効です。正しい ID を設定してください。	ボタンウィジェットが表示されエラーが発生した場合は、ウィジェットは表示されません。これらは購入者にメッセージを表示しません。アドレス帳ウィジェットやお支払い方法ウィジェットが表示されてエラーが発生した場合は、購入者にこのエラーが表示されます。 申し訳ございません。この Web サイトで問題が発生しました。販売事業者にご連絡ください。
MissingParameter	指定されたパラメータは間違っています、正しいパラメータを提供しなければなりません。	ボタンウィジェットが表示されエラーが発生した場合は、ウィジェットは表示されません。これらは購入者にメッセージを表示しません。アドレス帳ウィジェットやお支払い方法ウィジェットが表示されてエラーが発生した場合は、購入者にこのエラーが表示されます。 申し訳ございません。この Web サイトで問題が発生しました。販売事業者にご連絡ください。
PaymentMethodNotModifiable	Order Reference が与えられた状態の場合は、支払方法を変更できません。	この注文では支払方法を変更できません。販売事業者にご連絡ください。
ReleaseEnvironmentMismatch	Order Reference オブジェクトが提示された環境とは一致しない環境で生成されたものです。ウィジ	申し訳ございません。この Web サイトで問題が発生しました。販売事業者にご連絡ください。

	エットと Order Reference オブジェクトは同じ環境でなければなりません。	
StaleOrderReference	指定された Order Reference は許可された時間内で確認されずキャンセルされました。キャンセルされた Order Reference の支払方法と住所は関連付けできません。	セッションは期限切れです。Amazon アカウントでお支払いボタンをクリックして再度ご購入ください。
UnknownError	サービス内で未知のエラーがあります。	申し訳ございません。この Web サイトで問題が発生しました。販売事業者にご連絡ください。

JavaScript

処理開始前にデータをエンコーディング

HTML、JavaScript、URL 文字列などの FORM 内に出力する前に、Amazon から受け取ったデータをエンコードすることを推奨します。出力エンコーディングに悪意のあるスクリプトや Web サイトで実行可能なスクリプトを除外することを確実にします。

Open Web Application Security Project(OWASP) API である ESAPI は、無償で提供され、オープンソースのアプリケーションセキュリティコントロールライブラリです。これを利用することでローリスクなアプリケーションを素早く記述できます。この標準ライブラリをアウトプットエンコードに利用してください。

次のサンプルコードは、最も汎用タイプのデータエンコード方法です。

- HTML：
  `String safe = ESAPI.encoder().encodeForHTML( request.getParameter( "input" ) );`
- width や name などの HTML 属性の値：
  `String safe = ESAPI.encoder().encodeForHTMLAttribute( request.getParameter( "input" ) );`
- URL パラメータ値：
  `String safe = ESAPI.encoder().encodeForURL( request.getParameter( "input" ) );`
- JavaScript データ値：

```
String safe = ESAPI.encoder().encodeForJavaScript( request.getParameter( "input" ) );
```

Web ページに callback と widgets.js コードを追加

Web サイトに埋め込まれたウィジェットを正しく表示するためには、onAmazonLoginReady コールバックのハンドラーを提供し、Amazon Pay の JavaScript ファイルである Widgets.js のリファレンスを Web ページのソースコードに追加しなければなりません。Amazon Pay ウィジェットを表示したい Web ページにてこれを実行しなければなりません。

Amazon から提供されている JavaScript のコンテンツをローカルコピーや変更しないことにしてください。これはウィジェットのレンダリングや機能的エラーなど意図しない結果を引き起こします。

非同期 JavaScript

Amazon Pay はもっと速く Web ページでレンダリングするために、非同期ローディング JavaScript を限定的に提供しております。非同期ローディングの利用を選択する場合では、スクリプトタグ内の非同期属性だけサポートしています。

非同期で Amazon Pay ボタンをロードするためには、次の通りにします。

1. ページの head 部にある onAmazonLoginReady コールバックに **clientId** をセットします。
2. ページの body 部に、ボタンまたはウィジェットをレンダリングする JavaScript の前に **LoginWithAmazon** の を配置します。
3. window.onAmazonPaymentsReady 機能を定義します。この関数の中に JavaScript のリファレンスを移動します。
4. window.onAmazonPaymentsReady 関数の次に async 属性を Widgets.js のスクリプトタグに追加します。

タイミングの問題を防止するために、スクリプトタグは上記の順番に配置する必要があります。

次のサンプルコードは、非同期の Amazon Pay ボタンをロードする方法です。

```
<div id="PaymentsWidget"></div>

<script>
  window.onAmazonLoginReady = function(){
    amazon.Login.setClientId('YOUR_CLIENT_ID');
  };
  window.onAmazonPaymentsReady = function() {
    // Render the button here.
  };
</script>
```

```
<script async='async' type='text/javascript'
  src='https://static-na.payments-
amazon.com/OffAmazonPayments/us/sandbox/js/Widgets.js'>
</script>
```

上記のサンプルは SANDBOX 環境です。本番環境にコードを移行する場合は、適宜 URL を変更してください。

注意：

- ボタンまたはウィジェットをレンダリングする前に対応する div 要素を配置してください。ボタンとウィジェットは onAmazonPaymentsReady コールバック内でレンダリングして、widgets.js がロードされてから呼び出されます。
- widgets.js のローディングは一度だけ発生し、widgets.js がロードされた<script>タグ内の async 属性が必要です。

非同期属性のスクリプトタグの詳しい情報はを Mozilla Developer Network Script Tag Summary を参照してください。

インスタント支払通知（IPN）

インスタント支払通知メッセージのハンドリング

概要

Amazon はいくつかの非同期の処理での支払リクエストを行います。Amazon がリクエストを処理した後で、Amazon Pay オブジェクトに対応するステータスは更新されます。API を利用して定期的にこのステータスをチェックするかもしれませんが、インスタント支払通知（IPN）メッセージを受け取ることで対応することができます。

IPN メッセージは要求することなしに Amazon Pay から送信され、内部の注文管理システムで更新したり、注文を処理することができます。

次の条件で IPN は発生し、IPN メッセージの結果を待つ必要があります。

- 非同期のオーソリ（Authorize）API 呼び出しを実行した場合
- **Pending** の売上請求レスポンスが返ってきた場合
- 返金（Refund）API 呼び出しを実行した場合

ほとんどの変更通知はオブジェクトの状態変更です。しかしながら、その他の変更通知も IPN 発信されます。例えば、顧客が配送先住所を更新した場合や Amazon が支払方法を失敗にした場合です。

Amazon Pay オブジェクトの状態遷移についての詳しい情報は、Amazon Pay API リファレンスガイドの「状態と理由コード」セクションを参照してください。

受信 IPN メッセージの設定

IPN メッセージは、中身が XML ベースの通知データを含む HTTPS POST リクエストです。IPN メッセージを受信する前に次のことを実行してください。

1. セラーセントラルにエンドポイントを設定
    a. エンドポイントを設定するためには、セラーセントラルにログインして、**設定**ドロップダウンから**インテグレーション**を選択します。
    b. **インスタント通知設定**の**編集**ボタンをクリックして、インスタント通知を受信したいエンドポイントを**販売事業者様サイト URL** に入力します。本番環境では HTTPS である必要がありますが、SANDBOX 環境の HTTP は HTTPS の代理として許可されています。しかしながら、HTTPS のエンドポイントを指定できるのであれば、証明書は正確であり、完全な証明書である必要があります。
    c. 同じイベントで発生した IPN を 2 つ目のエンドポイントで受信したい場合は、**システムパートナー様 URL** に指定することができます。
2. エンドポイントと SSL 証明書が正しく動作しているのか確認します。
3. Web サービスが動作しているか、エンドポイントが HTTPS POST リクエストを受信できるか、通知を処理できるのか確認します。.htaccess のユーザーログインルールなどのアクセス制御している場合は除外します。
4. HTTPS が信頼されれた証明書提供会社からの正しい SSL 証明書を利用しているか確認します。詳しい情報は TLS/SSL 証明書を参照してください。

セキュア IPN の処理

なりすまし攻撃を防御するには、メッセージの確実性を確認するための IPN 署名を確実にしなければなりません。詳しい情報は、Amazon SNS メッセージの署名を確認を参照してください。

この確証に Amazon SDK を利用している場合は処理できます。利用しない場合は、SDK でインプリメントされているのと同じ方法で、Amazon SNS メッセージの署名を確認の中で説明している同じステップで行う必要があります。

カスタムデータフィールド

カスタムデータフィールドは利用可能です。これらは API 呼び出しで Amazon に送ることができ、IPN メッセージ経由で返されます。Amazon はデータを解析せず判断もしません。コンテンツが仕様として正しく同じタグに閉じるタグがあるかぎり、コンテンツは現状どおりに受け入れられ、同じ内容が返されます。

よって、カスタムデータは判断されず、Amazon に XSD やその他のカスタムデータフィールドの定義情報を送信する必要はありません。

次の表は利用可能なフィールドとデータタイプに関連されたリストです。

データタイプ	カスタムデータフィールド
OrderReferenceAttributes	SellerNote SellerOrderAttributes SellerOrderId StoreName* CustomInformation
Authorize	AuthorizationReferenceId SellerAuthorizationNote SoftDescriptor
Capture	CaptureReferenceId SellerCaptureNote SoftDescriptor
Refund	RefundReferenceId SellerRefundNote SoftDescriptor

***注意**：この値は、Amazon ログインにアプリケーションが関連付けられセットされます。

いくつかのデータフィールドは制限されていることに注意してください。いくつかは、購入者へのメール、購入者アカウント、販売事業者のトランザクションレポートで表示されることにも注意してください。

詳しい情報は、「提供可能な購入者メールコンテンツ」を参照してください。

サンプル通知

次は Amazon から利用可能な通知としてのリストです。

- OrderReferecne 通知
- Authorization 通知
- Capture 通知
- Refund 通知

次のサンプルコードは、それぞれの通知とそれに関連するカスタムデータフィールドを例示しております。

OrderReference IPN 通知サンプル

```
POST / HTTP/1.1
x-amz-sns-message-type: Notification
x-amz-sns-message-id: 432f33bf-9f84-5004-815f-7a6cfEXAMPLE
x-amz-sns-topic-arn: arn:aws:sns:EXAMPLE:59860EXAMPLE:TestTopic
x-amz-sns-subscription-arn:
    arn:aws:sns:EXAMPLE:59860EXAMPLE:TestTopic: EXAMPLE
Content-Length: 961
Content-Type: text/plain; charset=UTF-8
Host: ec2-EXAMPLE.compute-1.amazonaws.com
Connection: Keep-Alive
User-Agent: Amazon Simple Notification Service Agent
{
  "Type":"Notification",
  "MessageId":"cf5543af-dd65-5f74-8ccf-0a410EXAMPLE",
  "TopicArn":"arn:aws:sns:EXAMPLE:59860EXAMPLE:TestTopic",
  "Message":"{"NotificationReferenceId":"32d195c3-a829-4222-b1e2
      -14ab2EXAMPLE",
    "NotificationType":"OrderReferenceNotification",
    "SellerId":"YOUR_SELLER_ID_HERE",
    "ReleaseEnvironment":"Sandbox",
    "Version":"2013-01-01",
    "NotificationData":
    "<?xml version="1.0" encoding="UTF-8"?>
      <OrderReferenceNotification
        xmlns="https://mws.amazonservices.com/
        ipn/OffAmazonPayments/2013-01-01">
        <OrderReference>
        <AmazonOrderReferenceId>
          S23-1234567-1234567
        </AmazonOrderReferenceId>
        <OrderTotal>
        <Amount>106.00</Amount>
        <CurrencyCode>USD</CurrencyCode>
        </OrderTotal>
        <SellerNote>4%20Pack%20of%20BBQ%20Sauce</SellerNote>
        <SellerOrderAttributes>
          <SellerOrderId>5678-23</SellerOrderId>
          <StoreName>HammysBurgers.com</StoreName>
          <CustomInformation>
             Labor%20Day%20Promotion
          </CustomInformation>
        </SellerOrderAttributes>
        <OrderReferenceStatus>
        <State>CLOSED</State>
        <ReasonCode>SellerClosed</ReasonCode>
        <LastUpdateTimestamp>
          2013-04-01T10:49:59.532Z
        </LastUpdateTimestamp>
        </OrderReferenceStatus>
        <CreationTimestamp>
```

```
          2013-03-30T09:58:51.234Z
        </CreationTimestamp>
        <ExpirationTimestamp>
          2013-04-06T09:58:51.234Z
        </ExpirationTimestamp>
        </OrderReference>
        </OrderReferenceNotification>",
        "Timestamp":"2013-04-22T06:00:14Z"}",
        "Timestamp":"2013-04-22T06:00:15.108Z",
        "SignatureVersion":"1",
        "Signature":"deako5R0...CVmPQOI=",
        "SigningCertURL":"https://sns.EXAMPLE.amazonaws.com/
        SimpleNotificationService-f3ecfb7224c7233fe7bb5f59fEXAMPLE.pem",
        "UnsubscribeURL":"https://sns.EXAMPLE.amazonaws.com/
        ?Action=Unsubscribe
        &SubscriptionArn=arn:aws:sns:EXAMPLE:59860EXAMPLE:TestTopic:GUID"
 }
```

Authorization IPN 通知サンプル

AuthorizationReferenceId はリクエストで送信されたパラメータであり、Amazon が発行した ID であることに注意してください。

```
POST / HTTP/1.1
x-amz-sns-message-type: Notification
x-amz-sns-message-id: 6f7e123e-49c9-5c9d-a389-5bed0EXAMPLE
x-amz-sns-topic-arn: arn:aws:sns:EXAMPLE:59860EXAMPLE:TestTopic
x-amz-sns-subscription-arn:
arn:aws:sns:EXAMPLE:59860EXAMPLE:TestTopic: EXAMPLE
Content-Length: 961
Content-Type: text/plain; charset=UTF-8
Host: ec2-EXAMPLE.compute-1.amazonaws.com
Connection: Keep-Alive
User-Agent: Amazon Simple Notification Service Agent
{
  "Type":"Notification",
  "MessageId":"cf5543af-dd65-5f74-8ccf-0a410EXAMPLE",
  "TopicArn":"arn:aws:sns:EXAMPLE:59860EXAMPLE:TestTopic",
  "Message":
  "{"NotificationReferenceId":"32d195c3-a829-4222-b1e2-14ab28909513",
      "NotificationType":"PaymentAuthorize",
      "SellerId":"YOUR_SELLER_ID_HERE",
      "ReleaseEnvironment":"Sandbox",
      "Version":"2013-01-01",
      "NotificationData":
      "<?xml version="1.0" encoding="UTF-8"?>
        <AuthorizationNotification
          xmlns="https://mws.amazonservices.com/
                    ipn/OffAmazonPayments/2013-01-01">
        <AuthorizationDetails>
        <AmazonAuthorizationId>
          S23-1234567-1234567-0000001
        </AmazonAuthorizationId>
```

```
        <AuthorizationReferenceId>
          9bbe88cd5ab4435b85d717fd8EXAMPLE
        </AuthorizationReferenceId>
        <AuthorizationAmount>
        <Amount>5.0</Amount>
        <CurrencyCode>USD</CurrencyCode>
        </AuthorizationAmount>
        <CapturedAmount>
        <Amount>0.0</Amount>
        <CurrencyCode>USD</CurrencyCode>
        </CapturedAmount>
        <AuthorizationFee>
        <Amount>0.0</Amount>
        <CurrencyCode>USD</CurrencyCode>
        </AuthorizationFee>
        <SellerAuthorizationNote>
          Seller%20Auth%20 Note
        </SellerAuthorizationNote>
        <IdList/>
        <CreationTimestamp>
         2013-04-22T05:59:38.186Z
        </CreationTimestamp>
        <ExpirationTimestamp>
          2013-05-22T05:59:38.186Z
        </ExpirationTimestamp>
        <AuthorizationStatus>
        <State>Open</State>
        <LastUpdateTimestamp>
          2013-04-22T06:00:11.473Z
        </LastUpdateTimestamp>
        </AuthorizationStatus>
        <SoftDecline>false</SoftDecline>
        <OrderItemCategories/>
        <CaptureNow>false</CaptureNow>
        <SoftDescriptor>
             AMZ*Time Inc
        </SoftDescriptor>
        </AuthorizationDetails>
        </AuthorizationNotification>",
      "Timestamp":"2013-04-22T06:00:14Z"}",
  "Timestamp":"2013-04-22T06:00:15.108Z",
  "SignatureVersion":"1",
  "Signature":"W/cfaDzC...5glwqJk=",
  "SigningCertURL":"https://sns.EXAMPLE.amazonaws.com/
    SimpleNotificationService-f3ecfb7224c7233fe7bb5f59fEXAMPLE.pem",
  "UnsubscribeURL":"https://sns.EXAMPLE.amazonaws.com/
    ?Action=Unsubscribe
    &SubscriptionArn=arn:aws:sns:EXAMPLE:59860EXAMPLE:TestTopic:GUID"
}
```

Capture IPN 通知サンプル

サンプルコードに例示されている CaptureReferenceId は前述のサンプルで選択された AuthorizationReferenceId と同じです。

また、売上請求に成功した場合は、C で始まる注文 ID として Capture の ID は含まれます。例えば、P01-89574386-4998340-C064528 の通りです。

次は Capture 通知のサンプルです。

```
POST / HTTP/1.1
x-amz-sns-message-type: Notification
x-amz-sns-message-id: 64f5f75c-5799-53e5-b4c3-be8f1EXAMPLE
x-amz-sns-topic-arn: arn:aws:sns:EXAMPLE:59860EXAMPLE:TestTopic
x-amz-sns-subscription-arn:
arn:aws:sns:EXAMPLE:59860EXAMPLE:TestTopic: EXAMPLE
Content-Length: 961
Content-Type: text/plain; charset=UTF-8
Host: ec2-EXAMPLE.compute-1.amazonaws.com
Connection: Keep-Alive
User-Agent: Amazon Simple Notification Service Agent
{
  "Type":"Notification",
  "MessageId":"cf5543af-dd65-5f74-8ccf-0a410EXAMPLE",
  "TopicArn":"arn:aws:sns:EXAMPLE:59860EXAMPLE:TestTopic",
  "Message":
  "{"NotificationReferenceId":"32d195c3-a829-4222-b1e2-14ab2EXAMPLE",
    "NotificationType":"PaymentCapture",
    "SellerId":"YOUR_SELLER_ID_HERE",
    "ReleaseEnvironment":"Sandbox",
    "Version":"2013-01-01",
    "NotificationData":
    "<?xml version="1.0" encoding="UTF-8"?>
    <CaptureNotification
       xmlns="https://mws.amazonservices.com/
       ipn/OffAmazonPayments/2013-01-01">
    <CaptureDetails>
    <AmazonCaptureId>
      P01-8574386-4998340-C064528
    </AmazonCaptureId>
    <SellerCaptureNote>
       Seller%20Capture%20Note
    </SellerCaptureNote>
    <CaptureReferenceId>
      6f4d9dea0c234279a65e77994EXAMPLE
    </CaptureReferenceId>
    <CaptureAmount>
      <Amount>5.0</Amount>
      <CurrencyCode>USD</CurrencyCode>
    </CaptureAmount>
    <RefundedAmount>
      <Amount>0.0</Amount>
      <CurrencyCode>USD</CurrencyCode>
    </RefundedAmount>
    <CaptureFee>
      <Amount>0.0</Amount>
```

```
      <CurrencyCode>USD</CurrencyCode>
    </CaptureFee>
    <IdList/>
    <CreationTimestamp>
      2013-04-22T06:02:22.026Z
    </CreationTimestamp>
    <CaptureStatus>
      <State>Completed</State>
      <LastUpdateTimestamp>
        2013-04-22T06:02:25.227Z
      </LastUpdateTimestamp>
    </CaptureStatus>
    <SoftDescriptor>
      AMZ*softdescriptor
    </SoftDescriptor>
    </CaptureDetails>
    </CaptureNotification>",
    "Timestamp":"2013-04-22T06:00:14Z"}",
  "Timestamp":"2013-04-22T06:00:15.108Z",
  "SignatureVersion":"1",
  "Signature":"dUWd9lrs...iNGKnR4=",
  "SigningCertURL":"https://sns.EXAMPLE.amazonaws.com/
  SimpleNotificationService-f3ecfb7224c7233fe7bb5f59fEXAMPLE.pem",
  "UnsubscribeURL":"https://sns.EXAMPLE.amazonaws.com/
  ?Action=Unsubscribe
  &SubscriptionArn=arn:aws:sns:EXAMPLE:59860EXAMPLE:TestTopic:GUID"
}
```

Refund IPN 通知サンプル

```
POST / HTTP/1.1
x-amz-sns-message-type: Notification
x-amz-sns-message-id: 5f43584c-1f96-5880-9c98-119f5EXAMPLE
x-amz-sns-topic-arn: arn:aws:sns:EXAMPLE:59860EXAMPLE:TestTopic
x-amz-sns-subscription-arn:
arn:aws:sns:EXAMPLE:59860EXAMPLE:TestTopic: EXAMPLE
Content-Length: 961
Content-Type: text/plain; charset=UTF-8
Host: ec2-EXAMPLE.compute-1.amazonaws.com
Connection: Keep-Alive
User-Agent: Amazon Simple Notification Service Agent
{
  "Type":"Notification",
  "MessageId":"cf5543af-dd65-5f74-8ccf-0a410EXAMPLE",
  "TopicArn":"arn:aws:sns:EXAMPLE:59860EXAMPLE:TestTopic",
  "Message":
  "{"NotificationReferenceId":\
    "32d195c3-a829-4222-b1e2-14ab2EXAMPLE",
    "NotificationType":"PaymentRefund",
    "SellerId":"YOUR_SELLER_ID_HERE",
    "ReleaseEnvironment":"Sandbox",
    "Version":"2013-01-01",
    "NotificationData":
```

```
    "<?xml version="1.0" encoding="UTF-8"?>
    <RefundNotification xmlns="https://mws.amazonservices.com/
      ipn/OffAmazonPayments/2013-01-01">
      <RefundDetails>
        <AmazonRefundId>
          S23-1234567-1234567-0000003
        </AmazonRefundId>
        <RefundReferenceId>
          07fff0c4e05046958db7e47607e7db17
        </RefundReferenceId>
        <RefundType>SellerInitiated</RefundType>
        <RefundAmount>
          <Amount>5.0</Amount>
          <CurrencyCode>USD</CurrencyCode>
        </RefundAmount>
        <FeeRefunded>
          <Amount>0.0</Amount>
          <CurrencyCode>USD</CurrencyCode>
        </FeeRefunded>
        <CreationTimestamp>
         2013-04-22T06:07:34.617Z
        </CreationTimestamp>
        <RefundStatus>
          <State>Completed</State>
          <LastUpdateTimestamp>
            2013-04-22T06:09:20.178Z
          </LastUpdateTimestamp>
        </RefundStatus>
        <SellerRefundNote>
          Seller%20Refund%20Note
        </SellerRefundNote>
        <SoftDescriptor>
          AMZ*softDescriptor
        </SoftDescriptor>
      </RefundDetails>
      </RefundNotification>",
    "Timestamp":"2013-04-22T06:00:14Z"}",
  "Timestamp":"2013-04-22T06:00:15.108Z",
  "SignatureVersion":"1",
  "Signature":"kjac14DH...oQT6FbA=",
  "SigningCertURL":"https://sns.EXAMPLE.amazonaws.com/
  SimpleNotificationService-f3ecfb7224c7233fe7bb5f59fEXAMPLE.pem",
  "UnsubscribeURL":"https://sns.EXAMPLE.amazonaws.com/
    ?Action=Unsubscribe
    &SubscriptionArn=arn:aws:sns:EXAMPLE:59860EXAMPLE:TestTopic:GUID"
}
```

IPN メッセージの応答

それぞれの通知を受け取った場合は、”200 OK”レスポンスを Amazon に送信するようにエンドポイントを設定します。異なるレスポンスを送信した場合は、Amazon SNS は 14

日の間に毎時間毎にリトライを実行します。IPN を送信したがサーバに届かなかった、または、4xx のレスポンスを返した場合は、リトライを行いません。

注意：Amazon は非同期方法で支払要求を処理します。一般的には、それぞれの通知は、指定されたエンドポイントに正確に送信されます。しかしながら、Amazon SNS の配送や一時的なネットワークコンディションによって、時たま重複したメッセージになることがあります。重複した IPN メッセージを受信しても、支払ワークフローが誤動作しないように、アプリケーションを設計してください。

インスタント支払通知（IPN）で返されたコンテンツの通知データメンバは、次の有効な XSD で説明しております。また、すべての IPN メッセージは Amazon によって署名されております。
https://amazonpayments.s3.amazonaws.com/documents/payments_ipn.xsd

IPN ハンドリングのベストプラクティス

インスタント支払通知（IPN）を受信した後は、IPN で受信された情報以上のものが必要であるかチェックし、それぞれに一致する GET 系の処理を実行します。GetOrderReferenceDetails、GetAuthorizationDetails、GetCaptureDetails、GetRefundDetails が対象になります。

関連する API を呼び出し Amazon Pay オブジェクトの完全な詳細情報を取得することで、理由コードを取得して、エラーハンドリングすることができます。

理由コードの詳しい情報は、Amazon Pay API リファレンスガイドの状態と理由コードセクションを参照してください。

API インフォメーション

API の概要

すべての API 呼び出しは、適切に動作するために署名されなければなりません。API リクエストを署名するためには AmazonMWS キーが必要です。キーを取得するためには、セラーセントラルにログインし、インテグレーションにマウスをのせ、MWS Access キーをクリックしたページから参照できます。署名方法については、MWS ディベロッパーガイドに記載がありますが、Amazon Pay の SDK を利用することをお勧めします。

アクセストークン

アクセストークンは、ユーザーがサイトにログインした時に認証サーバで承認されます。アクセストークンは、クライアント、ユーザー、アクセススコープが指定されています。クライアントは顧客情報を取得するためにアクセストークンを利用しなければならず、配送先と支払方法にアクセスすることが許可されます。

アクセストークンは 350 文字以上の英数字と記号で構成されております。先頭は Atza| で始まります。

アクセストークンを受信した場合は、３つ部分から構成される JSON フォーマットで構成されています。

- access_token
- token_type
- expires_in（トークン有効期限の秒数）

サンプルは下記の通りです。

```
{
  "access_token":"Atza|IQEBLjAsAhRmHjNgHpi0U-Dme37rR6CuUpSR...",
  "token_type":"bearer",
  "expires_in":3600,

  "refresh_token":"Atzr|IQEBLzAtAhRPpMJxdwVz2Nn6f2y-tpJX2DeX..."
}
```

“expires_in”フィールドは、アクセストークンの有効期限が切れる前にリフレッシュするために利用できます。アクセストークンの期限が切れた場合は、retrieveProfile()呼び出しでエラーハンドリングを行う必要があります。

アクセストークンの取得

ユーザーがログイン後に、Web サイトやモバイルアプリに戻ります。この時点で認証コードはクライアントに送信されます。クライアント ID とクライアントシークレットを利用して Amazon ログイン認証サービスを呼び出すことでアクセストークンを含めることができます。

リダイレクトフロー中ではアクセストークンは URL に含まれます。ポップアップフロー中では、Widgets.js がアクセストークンを引き出すコードがあります。

提供可能な購入者メールコンテンツ

Amazon は次のシチュエーションで購入者にメールを送信します。

- Order Reference オブジェクトが承認（**Confirm**）された時
- 代金が売上請求（**Capture**）されたいかなる時
- 返金（**Refund**）が発行されたいかなる時
- 支払方法が変更されたいかなる時

Amazon は多くのコンテンツをメールで提供します。また、販売事業者様がメール本文に提供できる要素はいくつかあります。下記の表の購入者メールコンテンツは、アカウント設定と API 処理パラメータで定義することができます。

メールフィールド	参照元
販売事業者カスタマーサービスメールアドレス	Amazon Pay のアカウント作成時に指定されたもの、もしくは、セラーセントラルの**設定**の出品者**アカウント情報**から参照します。
販売事業者カスタマーサービス電話番号	
販売事業者ストア名	Amazon Pay のアカウント作成時に指定されたもの、もしくは、セラーセントラルの**設定**の出品者**アカウント情報**から参照します。SetOrderReferenceDetails 処理の **StoreName** リクエストパラメータを指定することで上書きすることができます。
販売事業者注文番号	SetOrderReferenceDetails 処理の **SellerOrderId** リクエストパラメータで指定します。
お支払い合計金額	SetOrderReferenceDetails 処理の **OrderTotal** リクエストパラメータで指定します。
注文日付	Order Reference オブジェクトが承認（**Confirm**）され、**Open** 状態に遷移した日時です。
販売事業者メモ	SetOrderReferenceDetails 処理、オーソリ（Authorize）処理、売上請求（Capture）処理、返金（Refund）処理の **SellerNote** リクエストパラメータで指定します。

ベストプラクティス

ベストプラクティスとしては、API 呼び出しで次の情報を設定することを推奨します。

- お支払い合計金額
- 販売事業者ストア名
- 販売事業者ご注文番号
- 追記事項

販売事業者ストア名と販売事業者情報の設定は次の詳細に影響します。

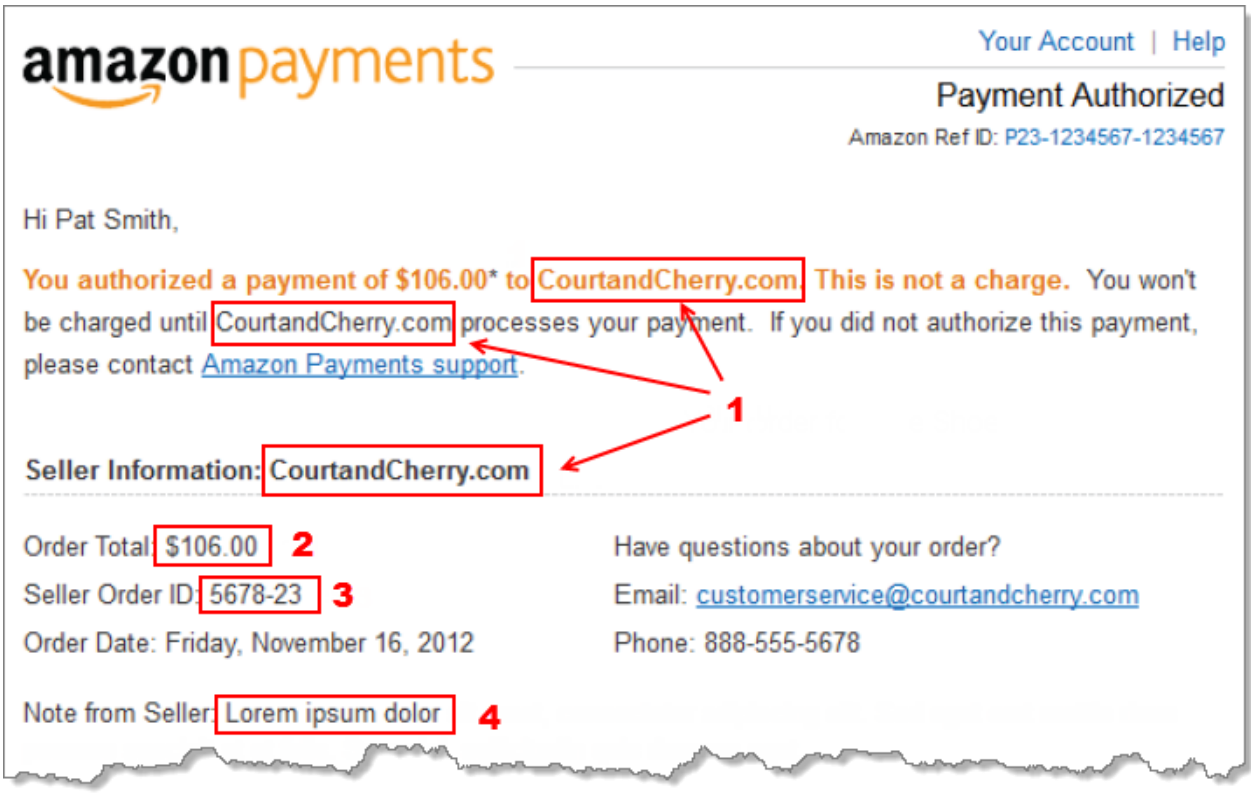
購入者メールとアカウントステータス
IPN メッセージ情報
決済とトランザクションレポート

複数のストア/サイトで利用している場合では、決済レポート内で販売事業者ストア名でフィルターすることができます。販売事業者注文番号でレポートを比較と一致することができます。

Amazon Pay ご利用内容の確認

Amazon にて購入者の認証がされ、Amazon Pay が Order Reference オブジェクトのシステムオブジェクトを生成し、Order Reference オブジェクトの追加属性を設定して ConfirmOrderReference 処理を呼び出した時にメールが送信されます。

購入者に送信されるご利用内容の確認メールは下記の通りです。数値が記載されている箇所は Order Reference オブジェクトと一致するパラメータです。



注意：販売事業者情報のメールアドレスと電話番号は SetOrderReferenceDetails API 呼び出しによって上書きすることができません。セラーセントラルの**設定**の**出品者アカウント情報**から参照されます。

注意：購入者に送信されるご利用内容の確認メールは以下の場合では送信されません。

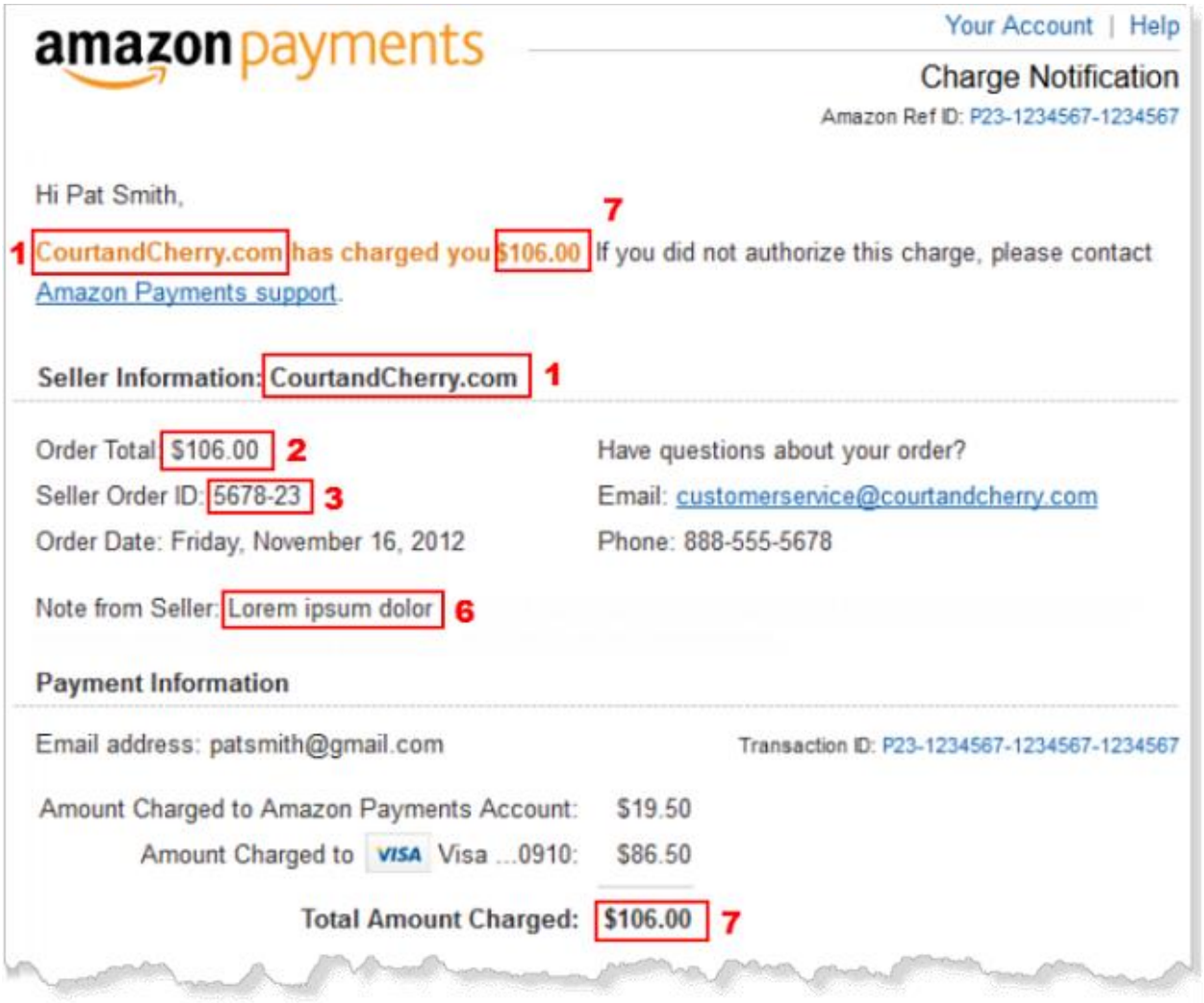
- Order Reference を確認(ConfirmOrderReference)してから 5 分以内にキャンセル(CancelOrderReference)にした場合
- Order Reference を確認(ConfirmOrderReference)してから 5 分以内に Closed にした場合

注意：SANDBOX 環境では Order Reference の確認（ConfirmOrderReference）してもメールは送信されます。

1	ストア名は販売事業者情報に配置されます。 &OrderReferenceAttributes.SellerOrderAttributes.StoreName この属性に設定しなければ、アカウント作成時の値をデフォルト値として利用します。 例：&OrderReferenceAttributes.SellerOrderAttributes.StoreName=CourtAndCherry.com
2	お支払い合計金額は２つのパラメータで設定します。 &OrderReferenceAttributes.OrderTotal.Amount &OrderReferenceAttributes.OrderTotal.CurrencyCode 例： &OrderReferenceAttributes.OrderTotal.Amount=106 &OrderReferenceAttributes.OrderTotal.CurrencyCode=USD
3	販売事業者ご注文番号は、次のパラメータで設定します。 &OrderReferenceAttributes.SellerOrderAttributes.SellerOrderId 例： &OrderReferenceAttributes.SellerOrderAttributes.SellerOrderId=5678-23
4	追記事項は、次のパラメータで設定します。 &OrderReferenceAttributes.SellerNote 例： &OrderReferenceAttributes.SellerNote=Your%20order%20for%20Blue%20Shoes

ご請求内容のお知らせ

売上請求通知メールで表示されるパラメータは返金通知メールと似ています。

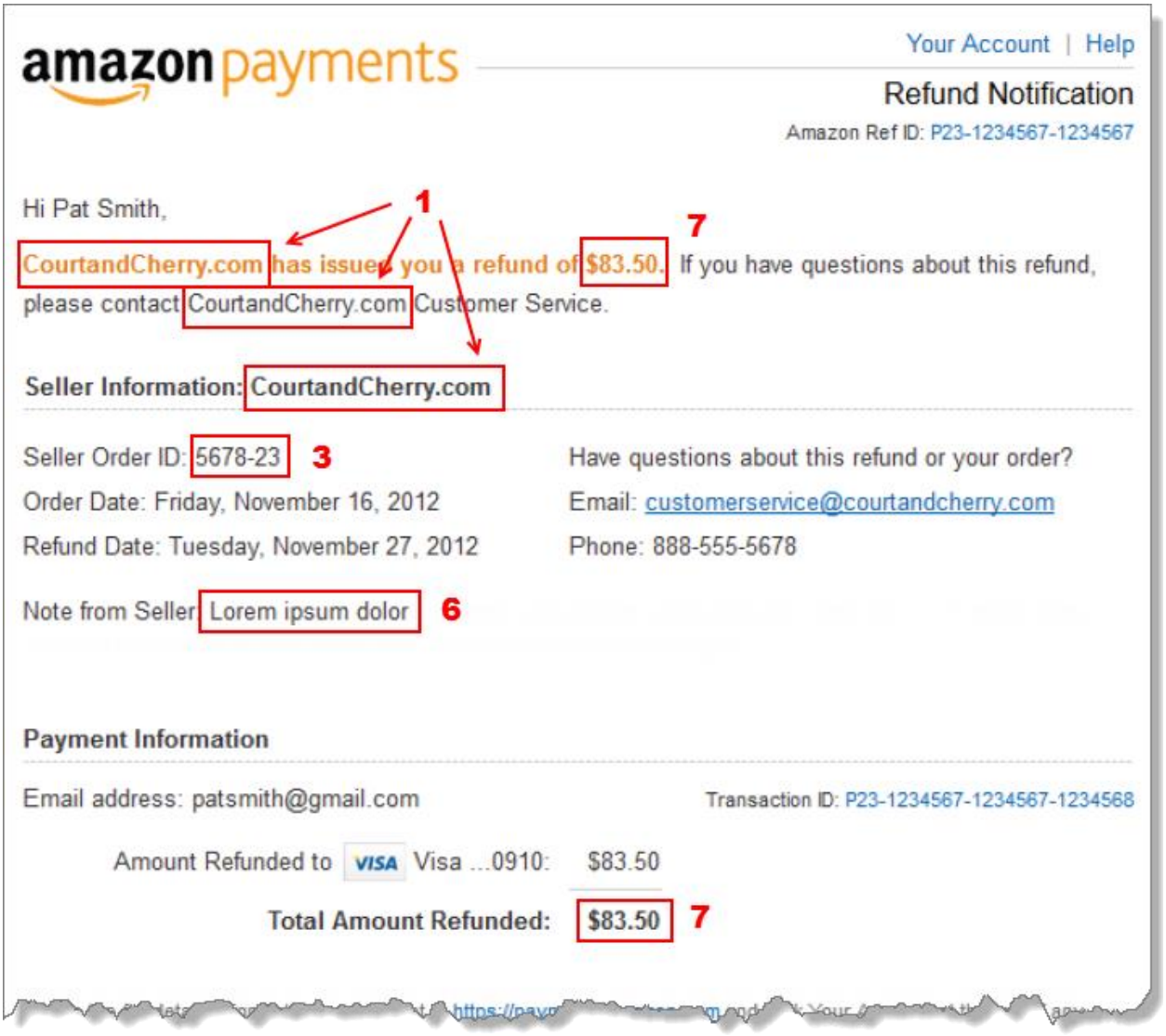


1	ストア名は販売事業者情報に配置されます。 &OrderReferenceAttributes.SellerOrderAttributes.StoreName この属性に設定しなければ、アカウント作成時の値をデフォルト値として利用します。 例：&OrderReferenceAttributes.SellerOrderAttributes.StoreName=CourtAndCherry.com
2	お支払い合計金額は２つのパラメータで設定します。 &OrderReferenceAttributes.OrderTotal.Amount &OrderReferenceAttributes.OrderTotal.CurrencyCode 例： &OrderReferenceAttributes.OrderTotal.Amount=106

	&OrderReferenceAttributes.OrderTotal.CurrencyCode=USD
3	販売事業者ご注文番号は、次のパラメータで設定します。 &OrderReferenceAttributes.SellerOrderAttributes.SellerOrderId 例： &OrderReferenceAttributes.SellerOrderAttributes.SellerOrderId=5678-23
6	追記事項は、次のパラメータで設定します。 &SellerCaptureNote 例： &SellerCaptureNote=Payment%20for%20Blue%20Shoes
7	ご請求金額合計は、２つのパラメータで設定します。 &CaptureAmount.Amount &CaptureAmount.CurrencyCode 例： &CaptureAmount.Amount=106 &CaptureAmount.CurrencyCode=USD

ご返金内容のお知らせ

次のサンプルは返金通知メールです。ストア名と販売事業者ご注文番号は Order Reference から参照され、以外の情報は Refund 処理の呼び出しでメールに設定されます。

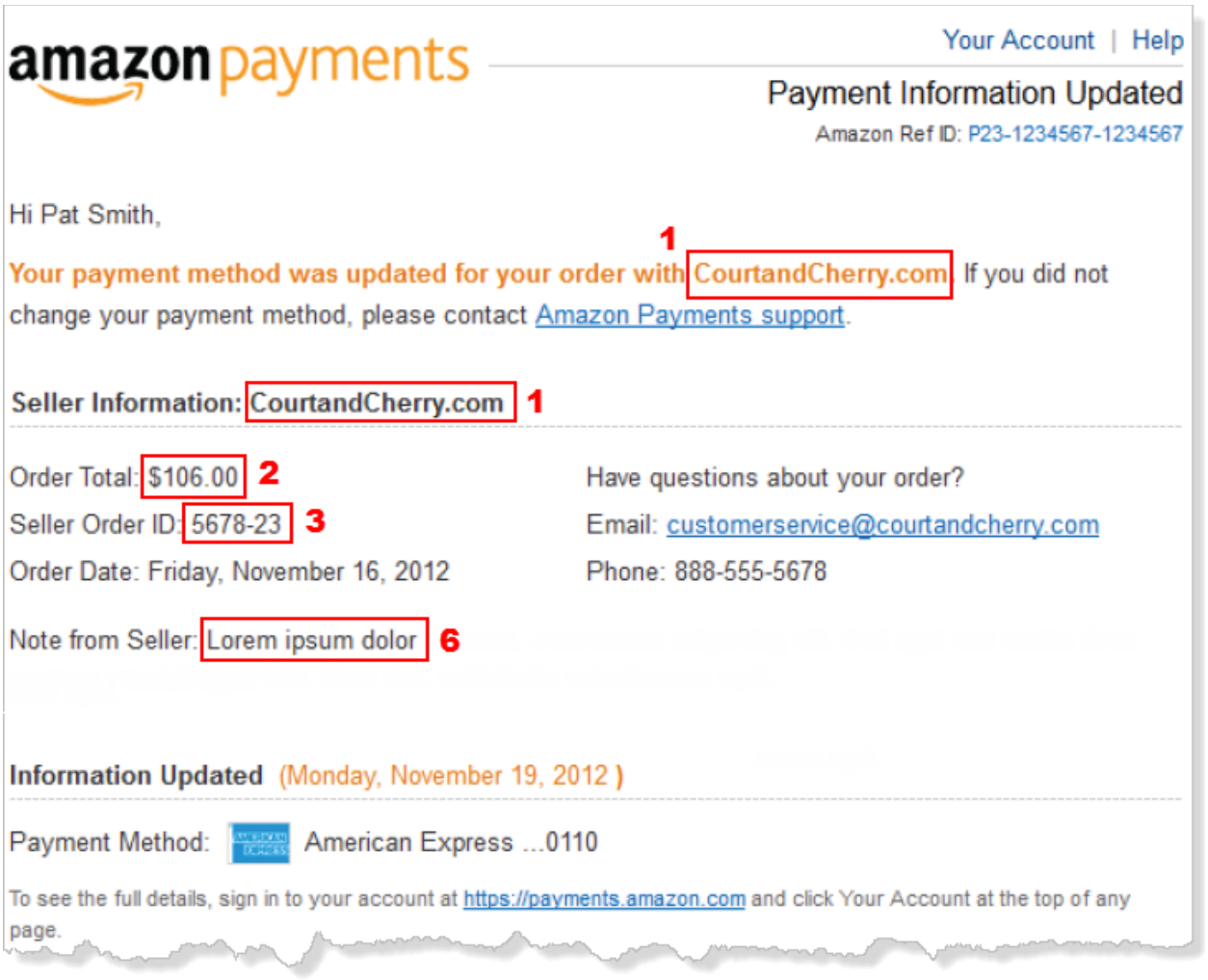


1	ストア名は販売事業者情報に配置されます。 &OrderReferenceAttributes.SellerOrderAttributes.StoreName この属性に設定しなければ、アカウント作成時の値をデフォルト値として利用します。 例：&OrderReferenceAttributes.SellerOrderAttributes.StoreName=CourtAndCherry.com
3	販売事業者ご注文番号は、次のパラメータで設定します。 &OrderReferenceAttributes.SellerOrderAttributes.SellerOrderId 例： &OrderReferenceAttributes.SellerOrderAttributes.SellerOrderId=5678-23
6	追記事項は、次のパラメータで設定します。

	&SellerRefundNote 例： &SellerRefundNote=Payment%20for%20Blue%20Shoes
7	返金額合計は、２つのパラメータで設定します。 &RefundAmount.Amount &RefundAmount.CurrencyCode 例： &RefundAmount.Amount=106 &RefundAmount.CurrencyCode=USD

支払方法変更のお知らせ

売上請求通知メールで表示されるパラメータは返金通知メールと似ています。



1	ストア名は販売事業者情報に配置されます。 &OrderReferenceAttributes.SellerOrderAttributes.StoreName この属性に設定しなければ、アカウント作成時の値をデフォルト値として利用します。 例：&OrderReferenceAttributes.SellerOrderAttributes.StoreName=CourtAndCherry.com
2	お支払い合計金額は２つのパラメータで設定します。 &OrderReferenceAttributes.OrderTotal.Amount &OrderReferenceAttributes.OrderTotal.CurrencyCode 例： &OrderReferenceAttributes.OrderTotal.Amount=106 &OrderReferenceAttributes.OrderTotal.CurrencyCode=USD
3	販売事業者ご注文番号は、次のパラメータで設定します。 &OrderReferenceAttributes.SellerOrderAttributes.SellerOrderId

	例： &OrderReferenceAttributes.SellerOrderAttributes.SellerOrderId=5678-23
6	追記事項は、次のパラメータで設定します。 &SellerCaptureNote 例： &SellerCaptureNote=Payment%20for%20Blue%20Shoes

ウィジェット

ボタンウィジェット

Amazon ではサイズが異なる複数のボタンを提供しており利用できます。ボタンサイズは利用されている環境に適切なものをご利用ください。

次の表はそれぞれのボタンウィジェットのオプションです。ボタンのテキストは、お客様が混乱しないようにするために変更はできません。

ボタン	ユースケース
Amazon アカウントでログイン 	Amazon アカウント情報を利用してサイトにログインすることを購入者に許可するボタンです。購入者は購入処理前、または途中でログインすることができます。購入者が Amazon アカウントでログインボタンをクリックした後は次の方法を利用できます。 • **購入者の個人情報にアクセス** ボタンウィジェットの scope パラメータを設定内容に応じて、購入者の Amazon 個人情報（名前、メールアドレス、ユニークなユーザーID）にアクセスできます。 購入者の購入エクステンションをカスタマイズするために個人情報を利用でき、ブラウザ履歴をトラックで

	きます。ユーザーID は購入者が再訪したときに利用できます。 • **購入者のためにローカルアカウントを作成** Amazon ユーザープロファイルに含まれる情報を利用してショップアカウントを作成します。このアカウントは通常のショップアカウントと同じであり、同じベネフィットを提供できます。異なるのは、顧客は Amazon ログインでパスワードを生成できないことです。
Amazon アカウントでお支払い 	このボタンは、Amazon アカウントを利用してカートページや多くの商品ページから登録されている配送先と支払方法を利用して注文を作成するために利用します。 購入者が Amazon アカウントでお支払いボタンをクリックした場合は、配送先と支払情報にアクセスする前に Amazon のユーザー名とパスワードを利用してログインします。 このボタンタイプでは、次の機能を提供することができます。 • **ログインして購入** Amazon ログインと同様に、サイトに初めてログインした時に購入者の個人情報を共有して良いか購入者に要求できます。 • **ゲスト購入** Amazon アカウントでお支払いボタンは、トランザクションを完了する前に、個人情報の共有なしでお買い物を完了することができます。 配送先と支払情報にアクセスするために購入者がログインした場合は、

	住所情報、郵便番号など送料計算するための情報のみ受け取ります。 ゲストとして Amazon Pay を利用してログインした場合は、購入者に同意画面は表示されません。 購入者が注文処理を完了した後で、購入者の名前、メールアドレス、配送先住所を取得できます。

ボタンウィジェットパラメータ

Amazon Pay ボタンウィジェットは、Web サイトの HTML コードの body 内に JavaScript コードを利用してレンダリングされます。JavaScript コードは、ボタンタイプ、カラー、サイズなどに影響するパラメータを含みます。

次は JavaScript ボタンウィジェットコードに含められるパラメータと値を説明します。

MerchantID

YOUR_MERCHAN_ID にアカウント作成で取得された販売事業者の ID（出品者 ID）と置き換えてください。販売事業者 ID（出品者 ID）が不明な場合は、セラーセントラルにログインし、インテグレーションメニューの MWS Access Key ページを参照してください。

Type

type パラメータは、Web ページに選択したいボタンイメージのタイプを指定するためのオプションパラメータです。type を指定しなかった場合は、LwA（Amazon ログイン）ボタンがデフォルト値としてセットされます。

次の表は有効な type パラメータ値、ボタンの説明、サンプルボタンイメージです。

type	説明	イメージ
LwA	Amazon アカウントでログインボタンを指定します。type が指定されていない場合のデフォルトボタンです。	
PwA	Amazon アカウントでお支払いボタンを指定します。	

Color

color パラメータは、ボタンの色を選択したい場合のオプションパラメータです。次の表は color コード、color の説明、color のサンプルです。

color	説明	サンプル
Gold	Gold（デフォルト値）	
LightGray	Light gray	
DarkGray	Dark gray	

Size

size パラメータはボタンのサイズを選択したい場合のオプションパラメータです。有効な size code は次の通りです。

パラメータ		
size	small(156px × 32px) medium(174px × 46px) large(312px × 64px) x-large(390px × 92px)	small(148px × 30px) medium(200px × 45px) large(296px × 60px) x-large(400px × 90px)

Scope

scope パラメータの値は、呼び出しレスポンスで返されたコンテンツと個人情報の共有するための同意画面のタイプに影響します。

次の scope パラメータの組み合わせが利用できます。

パラメータ	説明	要求事項
profile	ログイン後にすべての個人情報（名前、メールアドレス、ユーザーID）へのアクセスを提供します。	

profile:user_id	ログイン後に個人情報からユーザーID だけへのアクセスを提供します。	
postal_code	メインの配送先住所の郵便番号だけ提供します。	**postal_code** を単独、または、**profile**、**profile:user_id** スコープと複合してアクセス要求した場合は、購入者は情報共有の同意画面が表示されます。
payments:widget	Amazon Pay のウィジェット（住所と支払ウィジェット）を Web サイト上で表示させる場合に必須です。次のパラメータ無しで利用した場合は、ConfirmOrderReference 呼び出し後に完全な配送先住所へのアクセスが与えられます。	
payments:shipping_address	アドレス帳ウィジェットに選択された住所を直ぐに GetOrderReferenceDetails API 呼び出し経由で、完全な配送先住所としてアクセスできます。	• scope parameter に **Payments:widget** • GetOrderReferenceDetails 呼び出しの AddressConsentToken(=AccessToken)をセットします。

例　scope:profile payments:widget payments:shipping_address

payments:shipping_address は購入者の配送先住所へのアクセスを提供します。完全な配送先住所を取得するためには、GetOrderReferenceDetails API を呼び出す必要があります。詳しい情報は、Step 4：アドレス帳とお支払い方法ウィジェットの追加か、Amazon Pay API リファレンスガイドの GetOrderReferenceDetails を参照してください。

Popup

popup パラメータは、認証ためにポップアップウィンドウを表示するか、Amazon Pay ページでリダイレクト認証するかを決定します。

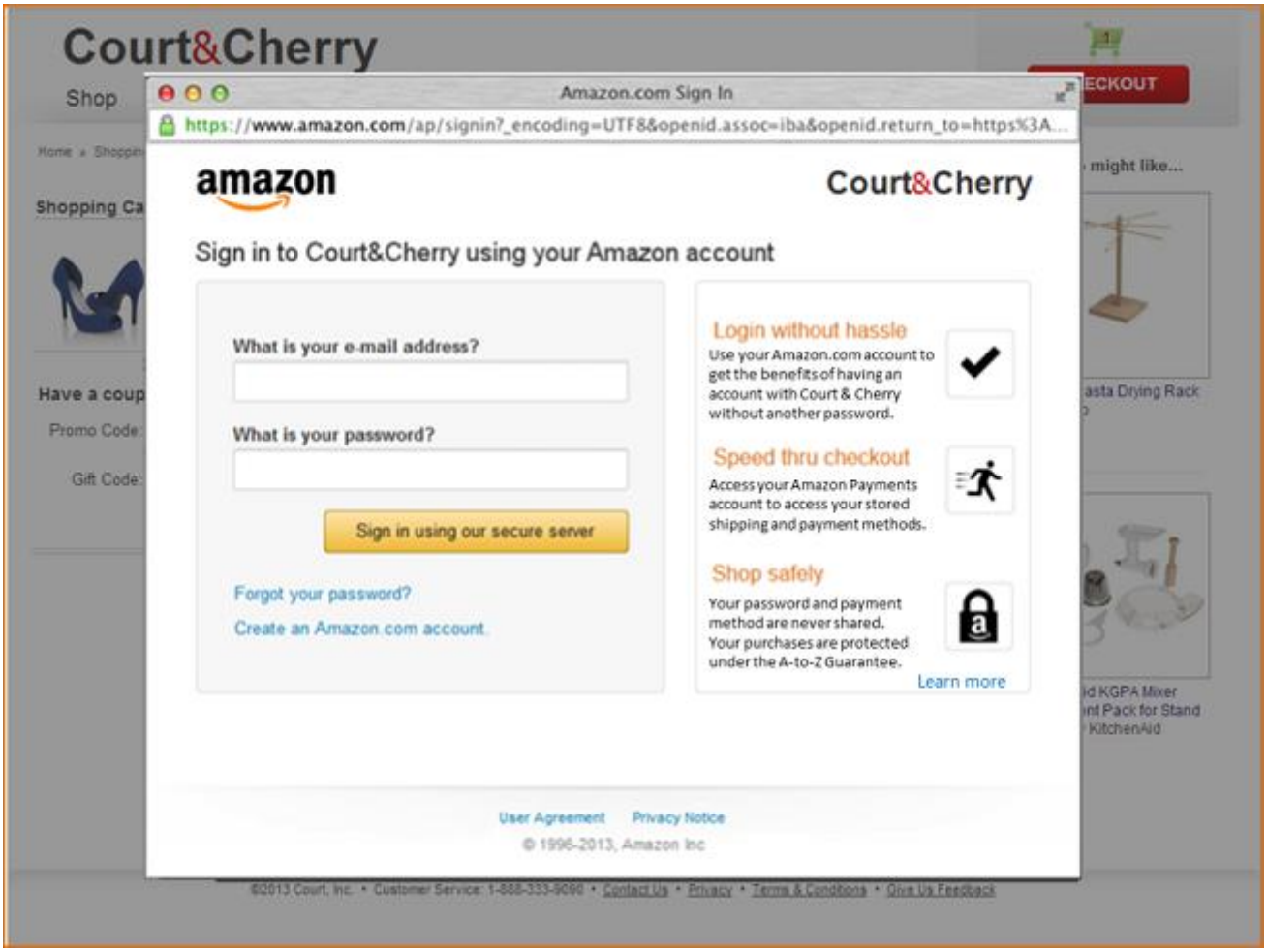
popup パラメータは次の中から 1 つを利用します。

- *true* – Amazon Pay は、購入者は Web サイトから離れずに認証することができるポップアップダイアログを認証画面として表示します。この値は暗号化されたデスクトップ PC での利用を推奨します。このオプションは正しい SSL 証明書のある HTTPS ページに属するボタンでは必要です。デフォルト値です。

- *false* - 購入者は同じブラウザウィンドウ内で Amazon Pay ページにリダイレクトされます。この値は、デバイスに最適化されたスマートフォンや、非セキュアページボタンウィジェットをレンダリングする場合での利用に推奨します。リダイレクト先の URL は正しい SSL 証明書のある HTTPS プロトコルである必要があります。

  詳しい情報は、リダイレクト認証の利用を参照してください。

Amazon Pay にポップアップパラメータを true にセットして導入した場合は、購入者は Amazon アカウントのメールアドレスとパスワードが求められます。



上記画面イメージでは、ログイン認証画面に会社のロゴを追加することができます。セラーセントラルにログインして Amazon ログイン画面より各アプリケーション毎にロゴを設定することができます。

リダイレクト URL

redirect URL のパラメータでは、購入者が認証に成功した後に指定されたページにリダイレクトします。例えば、認証に成功した後で、購入者に配送先住所または支払方法の詳細を選択する購入フローの画面に遷移させることができます。ボタンコードに popup:false を利用することで、リダイレクトログインエクスペリエンスができます。それには、セラーセントラルにログインして、Amazon ログインのアプリケーションにリダイレクト先の URL を事前に登録しなければなりません。

エラーハンドリング

Amazon はコード内の onError ハンドラの設定を推奨します。この onError コードはオプションですが、インテグレーションの問題を解決するために考慮するべきです。詳しくは、SANDBOX 環境でのインテグレーションテストを参照してください。

ログアウトオプションの追加

ユーザーがログインしている場合は、Web サイト上にログアウトオプション（ハイパーリンク）を提供するべきです。ログアウトオプションはキャッシュされたトークンと個人情報（名前など）を Web サイトから削除します。それから、再度 Web サイトにログインボタンを表示します。

Amazon ログイン SDK for JavaScript を利用している場合は、キャッシュされたトークンを削除するために amazon.Login.logout メソッドを呼びます。

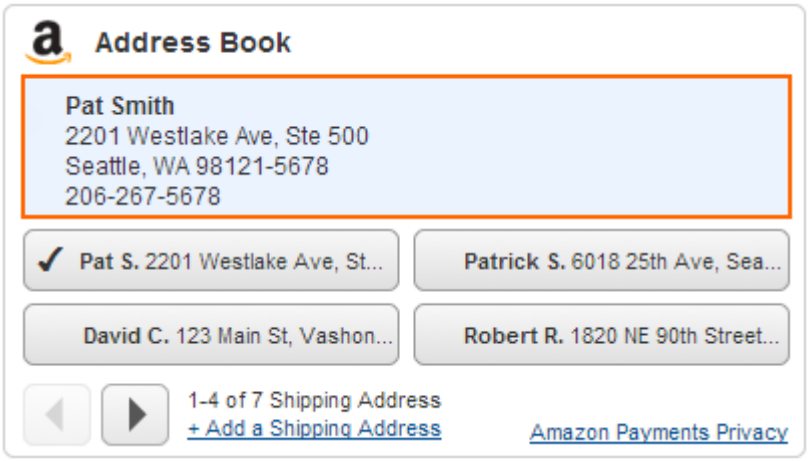
```
<script type="text/javascript">
  document.getElementById('Logout').onclick = function() {
    amazon.Login.logout();
  };
</script>
```

この次に、デフォルトで amazon.Login.authorize を呼び出して、ログイン画面を表示します。

Amazon Pay ウィジェット

Amazon Pay ウィジェットには、アドレス帳とお支払い方法ウィジェットの 2 種類があります。購入者は、Amazon アカウントに登録された配送先と支払情報にアクセスするために、埋め込まれた Amazon Pay ウィジェットを利用します。

アドレス帳ウィジェットは、購入者の Amazon アカウントに登録された住所を表示します。



お支払い方法ウィジェットは、購入者の Amazon アカウントに登録された支払方法を表示します。

Amazon Pay ウィジェットの CSS コード

Amazon Pay のアドレス帳と支払ウィジェットを表示する場合は、width と height パラメータを指定する必要があります。指定しなければウィジェットはレンダリングされません。許可されたパラメータの範囲を設定します。

次の表では、Amazon が推奨する 1 カラムと 2 カラムウィジェットを指定するために width と height の有効なパラメータです。

	2 カラム		1 カラム	
パラメータ	推奨	有効	推奨	有効
width	400px	400px – 600px	300px	300px – 399px
height	228px	228px – 400px	228px	228px – 400px

CSS ユニット単位としては、パーセント、REM、EM などが利用できます。

次のサンプルコードは CSS ファイル内に配置します。このコードはモバイルとデスクトップ Web ページの両方の値をセットしています。

```
/* Please include the min-width, max-width, min-height
/* and max-height if you plan to use a relative CSS unit
/* measurement to make sure the widget renders in the
/* optimal size allowed.
*/

#addressBookWidgetDiv {
  min-width: 300px;
  max-width: 600px;
  min-height: 228px;
  max-height: 400px;
}


#walletWidgetDiv {
  min-width: 300px;
  max-width:600px;
  min-height: 228px;
  max-height: 400px;
}


/* Mobile optimized and small window */

#addressBookWidgetDiv {
  width: 100%;
  height: 228px;
}

#walletWidgetDiv {
  width: 100%;
  height: 228px;
}

/* Desktop and tablet */

@media only screen and (min-width: 768px) {

  #addressBookWidgetDiv {
    width: 400px;
    height: 228px;
  }

  #walletWidgetDiv {
    width: 400px;
    height: 228px;
  }
}
```

その他の方法としては、ウィジェットを表示したそれぞれの HTML ページの<head>タグ内にアドレス帳とお支払い方法ウィジェットのための CSS スタイルを配置することもできます。

```
<!-- Include the following in your  HTML file -->
<style type="text/css">
  #addressBookWidgetDiv{width: 400px; height: 228px;}
  #walletWidgetDiv {width: 400px; height: 228px;}</style>
```

スマートフォンのための折りたたみウィジェットの利用

全ての Amazon Pay のウィジェットは、タブレットやスマートフォンなどのタッチスクリーンに最適化されています。しかしながら、スマートフォン向けに Web サイトが別れている場合は、Amazon は Web サイトに折りたたみできるウィジェットの利用を推奨します。

折りたたみウィジェットは 1 カラムユーザーインターフェイスに合うようにデザインされており、購入者は最小サイズでスクロールできるように広げたり、折りたたむことができます。小さいサイズのウィジェットを配置することで Web サイトに充分なスペースを確保でき、購入者デバイスの全画面で利用することもできます。標準的な方法である高さと幅を固定化することは、購入者が広げたり折りたたむことができません。この標準的なウィジェットは Step 1：登録で説明されています。

折りたたみ可能なアドレス帳とお支払い方法ウィジェットは、スマートフォンの横幅に合わせて縮小します。折りたたみウィジェットは、折りたたみ時は 135 ピクセルの高さで固定されます。購入者は予め選択された配送先住所またはお支払い方法を変更するためにウィジェットを広げることができます。このような場合では、ウィジェットはデバイスに応じて最大の高さと幅で広がります。購入者が選択した場合は、ウィジェットは自動的に折りたたまれ購入処理を続けることができます。

次のステップは、折りたたみウィジェットを導入する方法です。

1. スマートフォンに最適化されたページそれぞれの head セクションに META タグを追加します。この META タグは、購入者がスマートフォンでページズームする必要がない参照可能で適切なウィジェットを表示します。

```
<meta name="viewport" content="width=device-width, initial-scale=1.0,
maximum-scale=1.0"/>
```

2. **アドレス帳**または**お支払い方法**ウィジェットを生成するコードに、designMode パラメータに **smartphoneCollapsible** をセットして変更します。次はサンプルです。

```
new OffAmazonPayments.Widgets.AddressBook({
  sellerId : 'YOUR_SELLER_ID_HERE',
  onOrderReferenceCreate: function(orderReference) {
```

```
    orderReference.getAmazonOrderReferenceId();
  },
  design : {
    designMode: 'smartphoneCollapsible'
  },
  onAddressSelect : function(orderReference) {
  },
  onError : function(error) {
  }
}).bind("addressBookWidgetDiv");
```

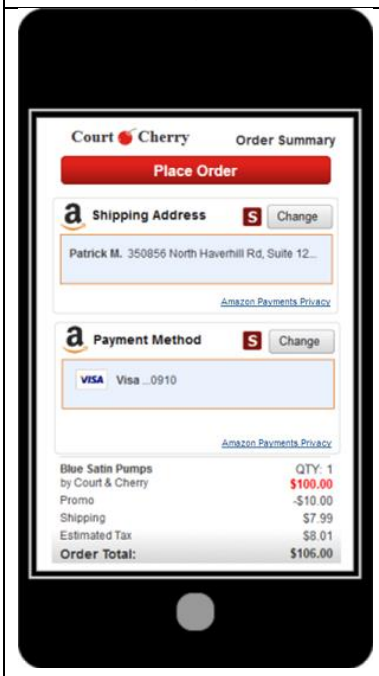
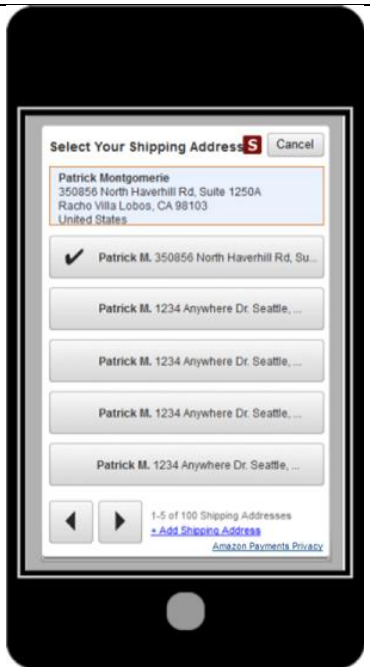
上記のサンプルはアドレス帳ウィジェットを変更する方法です。必要であれば、同様の変更をお支払い方法ウィジェット、参照のみのアドレス帳とお支払い方法ウィジェットに行います。この折りたたみウィジェットには、width と height を指定できません。

次のサンプルは購入者が影響する折りたたみウィジェット方法です。

- 図１では、最初にウィジェットは折りたたまれます。ウィジェットはデフォルトの配送先住所と支払方法を表示します。

- 図２では、購入者が配送先住所または支払方法を変更したい場合に、購入者が変更ボタンをクリックします。それからウィジェットは入力画面を全画面に広げます。

- 購入者が異なる配送先住所または支払方法を選択、もしくは、キャンセルボタンをタップした後では、ウィジェットは再度折りたたまれます。

図１：折りたたまれたアドレス帳とお支払い方法ウィジェット	図２：広げられたアドレス帳ウィジェット
	

TLS/SSL 証明書

TLS/SSL イントロダクション

Amazon は常にセキュアな通信を利用することを推奨します。これらには２つの事例があり、サーバは信頼できる証明書認証機関から発行された正しい TLS/SSL を必要とします。

- **ログイン**–購入者がログイン中は、ポップアップウィンドウか、購入者を他のページにリダイレクトする２つのオプションがあります。ログインタイプに応じてどちらかを選択します。これらは次のセキュアな通信を利用しなければなりません。
    - ポップアップログイン：TLS/SSL はボタンが埋め込まれている Web サイトで必要です。
    - リダイレクトログイン：購入者がリダイレクトされる URL はセキュアなページでなければなりません。

  注意：ローカル環境でテストする場合は、TLS/SSL 証明書は必要ありません。（例：http://localhost）

- **IPN メッセージ** – IPN（インスタントペイメント通知）メッセージはセキュアなエンドポイントにのみ送信されます。Amazon の正確な証明なしでは、販売事業者が所有する実際の IPN メッセージなどのデータを誰かが盗むことができるので送信しません。

TLS/SSL とは

トランスポート・セキュリティ・レイヤー（Transport Security Layer：TLS）とセキュア・ソケット・レイヤ（Secure Sockets Layer：SSL）は、Web サーバとブラウザ間のデータを暗号アルゴリズムを利用して安全に転送をするため設計されたプロトコルです。TLS/SSL は正しく送信元からデータが転送されており、転送中は第三者によって変更や参照されていないか確実にします。Amazon がサポートしている TLS/SSL の詳細なバージョンは Amazon Pay の Web サイトに追加しております。

HTTP vs HTTPS
URL アドレスに HTTPS プロトコルが含まれている場合は先頭は https となり、データは安全に転送されていることを示します。HTTP と HTTPS の違いは、HTTPS のデータは最初から TLS/SSL プロトコルで転送され、安全が確保されています。

TLS/SSL 証明書

TLS/SSL はセキュアと転送データを保護するために証明書を利用します。証明書は、組織、国、有効期限、Web サイトのアドレス、この情報を認証した者の証明書 ID などの情報を含む証明書の所有者の情報を含みます。更に、データ改ざんされない証明であるパブリックキーとハッシュ値も含みます。次は証明書のサンプルです。

```
Company Root CA 9
==================
-----BEGIN CERTIFICATE-----
MIIDQTCCAimgITBmyfz5m/jAo54vB4iXxxababbmljZbyjANBgkqhkiG9w0BAQsF
ADA5MQswCQYDVQQGEwJVUzEPMA0GA1UEChMGQW1hem9uMRkwFwYDVQQDExBBbWF6
b24gUm9vdCBDQSAxMB4XDTE1MDUyNjAwMDAwMFoXDTM4MDExNzAwMDAwMFowOTEL
N+gDS63pYaACbvXy8MWy7Vu33PqUXHeeE6V/Uq2V8viTO96LXFvKWlJbYK8U90vv
o/ufQJVtMVT8QtPHRh8jrdkPSHCa2XV4cdFyQzR1bldZwgJcJmApzyMZFo6IQ6XU
5MsI+yMRQ+hDKXJioaldXgjUkK642M4UwtBV8ob2xJNDd2ZhwLnoQdeXeGADbkpy
rqXRfboQnoZsG4q5WTP468Sample
-----END CERTIFICATE-----
```

証明書チェーン

グローバルで大きな公開鍵証明書認証局（CAs）は、TLS/SSL 証明書をその他の代理として認証します。通常は地域レベルで処理されます。サーバの証明書が中間証明によって発行されている場合は、サーバは中間証明の認証がされなければなりません。これはローカルに保存されたルート証明書に対して順番に確認することになります。

次のステップは確認するための順番です。

1. サーバから証明書をダウンロードします。
2. サーバの証明書が Web サイトの名前に合致しており、中間証明によって署名されているか確認します。
3. 中間証明がローカルに保存されている信頼されるルート認証書の 1 つに署名されているか確認します。

中間証明書認証局は、その他の中間証明書認証局の証明書を発行できます。これは、証明書チェーンは 3 つの証明書より多くなることを意味します。

なぜ TLS/SSL 証明書が必要なのでしょうか。

TLS/SSL 証明書を利用するいくつかの理由は次の通りです。

- **セキュリティ**：TLS/SSL 証明書を利用する大きな理由は、購入者のブラウザとサーバ間でセキュアなデータを保つことです。これはむき出しのインターネットから、注文、支払詳細、購入者の名前、パスワードなどの購入者のデータを盗まれることを防止します。
- **購入者の信頼**：TLS/SSL 証明書を含む場合は、証明書認証局は Web ページに表示されるシールを発行します。このシールは Web サイトの信頼性に重要です、なぜなら、購入者はデータがセキュアであることを知るからです。次のイメージはいくつかのサンプルシールです。

- **トラフィック**：Google などの検索エンジンは e コマースサイトのランキングについて、セキュアの通信を利用しているものより、非セキュアな通信をしているサイトを低く処理します。

Amazon が要求する TLS/SSL 証明書

MD5 with RSA の制限

Amazon は証明書チェーンの下（通常は example.com などのドメイン名）を確認し、MD5 with RSA 暗号を利用した証明書署名アルゴリズムを利用しているサイトの TLS/SSL コネクションを成立させません。

共通 TLS/SSL エラー

中間証明書の間違い
中間証明書の間違は、正しい証明書をインストールする時によく起こりますが、サーバには中間証明書は保存されません。これが発生したときは、信頼のチェーンは成立しません。この問題を防止するためには、ローカルに保存されたチェーン内の全ての証明書を確認します。

証明書名の不一致
このエラーはインストールされた証明書の名前が Web サイトのアドレスと異なるからです。このエラーを解決するためには、Web サイトの新しい証明書を購入する必要があります。

証明書の購入

証明書は多くのホスティング会社からインターネット上で購入することができます。TLS/SSL 証明書は公開鍵証明書認証局（CAs）ネットワークで発行され維持されています。これらは、厳格なセキュリティ標準として IT 業界では通常はよく知られた会社です。

標準と拡張の２つのタイプの証明書があります。標準バージョンは Amazon Pay での要求に合い拡張タイプより高くなく、数分で購入で発行されます。証明書価格は多様です。多くの証明書は 30 日間の無料トライアルがあります。

付録

インテグレーション概要

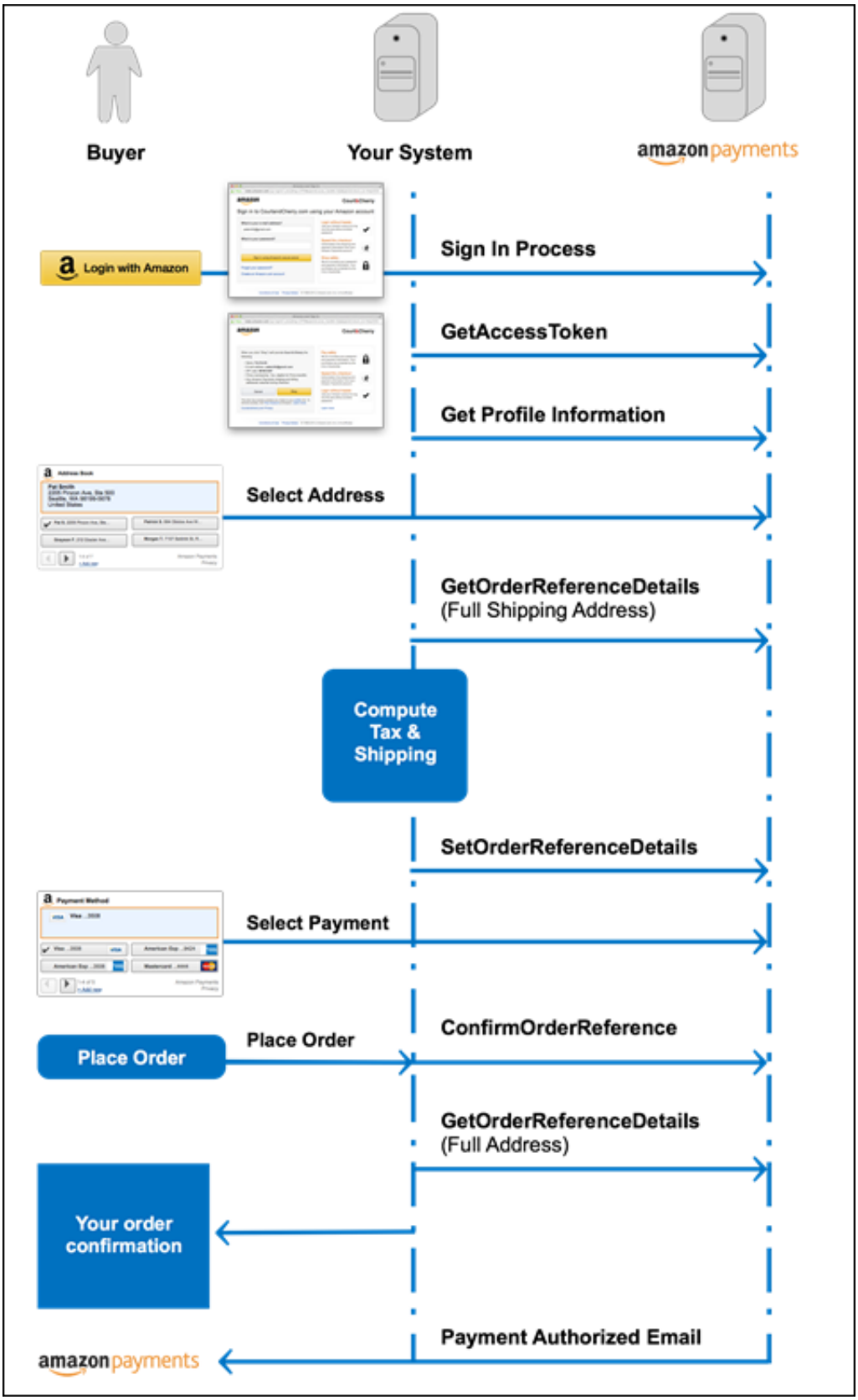
次の概要は、Web サイトで代表的な Amazon Pay の利用方法です。このステップは、購入での利用と支払処理の２つのグループで構成されています。

Amazon Pay を購入で利用

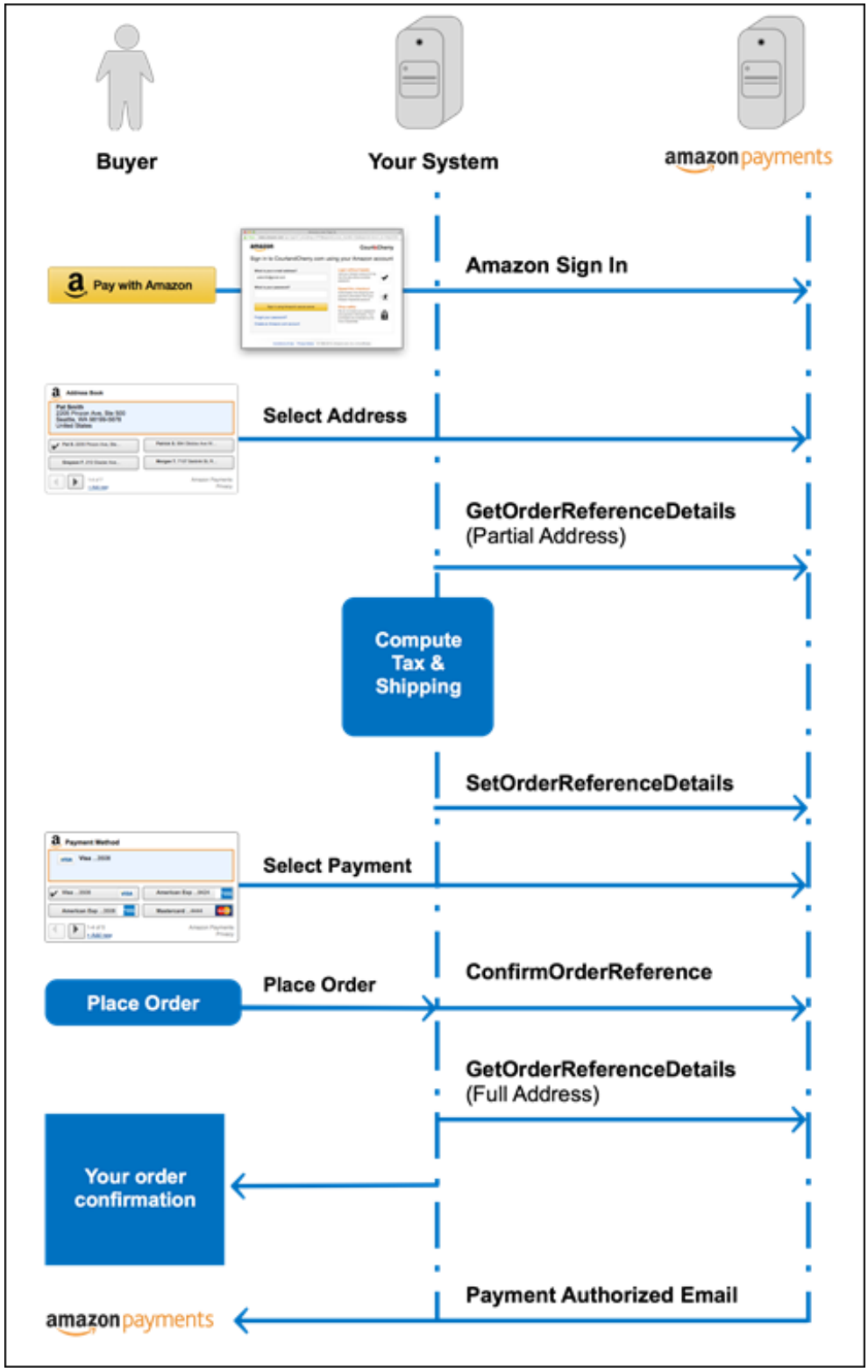
1. 購入者が購入フローを開始するページ（例えば、アカウント生成ページ、ショッピングカート、商品詳細ページ）に **Amazon アカウントでログイン**ボタンまたは **Amazon アカウントでお支払い**ボタンを追加します。Amazon アカウントでログインボタンは、購入者に Amazon 認証を利用します。Amazon アカウントでお支払いボタンは認証と支払で利用できます。
2. 購入者の証明が成功した後は、配送先住所を収集するページ上に Amazon の**アドレス帳**ウィジェットを表示し、支払情報を収集するページ上に Amazon の**お支払い方法**ウィジェットを表示します。
3. 配送料を計算するために GetOrderReferenceDetails 処理を呼び出して配送先住所を取得します。その後で購入者に配送料を表示します。
4. （オプション）注文確認ページに参照のみの**アドレス帳**と**お支払い方法**ウィジェットを表示します。
5. Amazon に SetOrderReferenceDetails 処理を呼び出して、注文合計金額とその他のオプション情報を提供します。
6. 購入者が注文を完了した後で、Amazon に ConfirmOrderReference 処理を呼び出して注文を確定します。
7. **Amazon アカウントでお支払い**ボタンのゲストアウトチェックアウトオプションを利用している場合は、Amazon に GetOrderReferenceDetails 処理を呼び出して完全な配送先住所を取得します。
8. 購入者に購入完了画面を表示し、注文完了メールを送信します。

注意：Amazon も「ご利用内容の確認」メールを Order Reference を確定させた時に購入者へ送信します。これは Amazon Pay を利用して購入支払が生成されたことを確認する情報です。このメールは注文された時と後ほど実行されるオーソリ要求とは関係ありません。

次のイメージは Amazon ログインと Amazon Pay を利用した注文で有効な方法を説明します。


Buyer
Your System
amazonpayments
Login with Amazon
Sign In Process
GetAccessToken
Get Profile Information
Select Address
GetOrderReferenceDetails
(Full Shipping Address)
Compute Tax & Shipping
SetOrderReferenceDetails
Select Payment
Place Order
Place Order
ConfirmOrderReference
GetOrderReferenceDetails
(Full Address)
Your order confirmation
Payment Authorized Email
amazonpayments

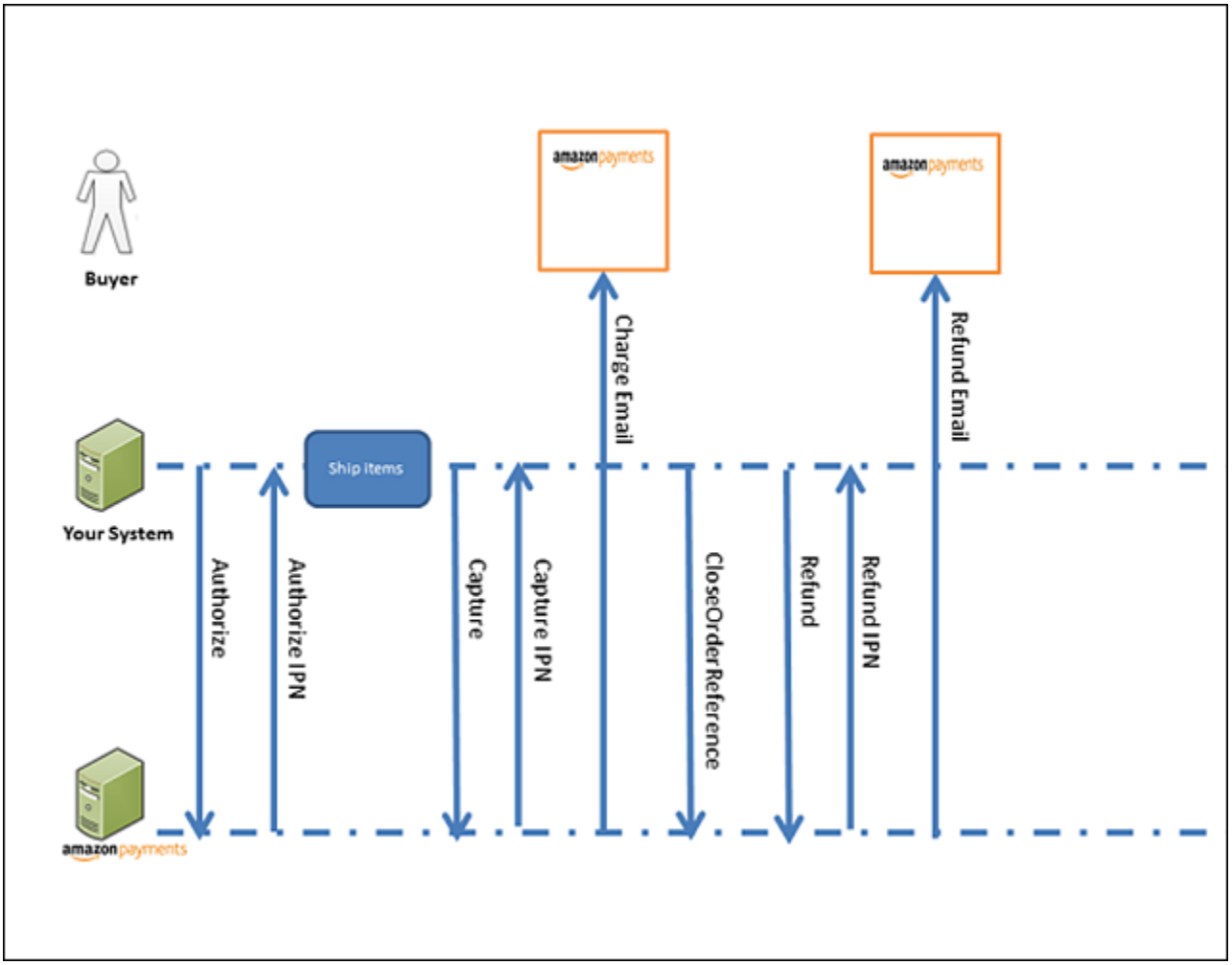
次のイメージは Amazon Pay を利用した注文で有効な方法を説明します。



支払処理

1. Amazon にオーソリ処理を呼び出して購入者の支払をオーソリします。
2. オーソリリクエストの処理ステータスをオーソリのインスタント支払通知で確認します。Amazon は同期と非同期モードのオーソリ処理の両方を提供していることに注意してください。
   同期モードでは、すぐに **Open** か **Declined** の処理ステータスを受け取ります。非同期モードでは、最初にオーソリオブジェクトの **Pending** ステータスを受け取ります。非同期モードが完了した後は、Amazon は最終処理ステータスをインスタント支払通知（IPN）経由で通知します。
   非同期モードを利用している場合は、Amazon からの最終処理ステータスを受け取るまで購入者の注文完了を待たせることはありません。Amazon に注文を確定した後は、すぐに購入者には注文完了ページを表示するべきです。
3. 注文を出荷し、売上請求処理を呼び出して支払の売上請求を行います。
4. 売上請求要求の処理ステータスを **Capture** IPN で確認します。Amazon は売上請求に成功した後で購入者へ「ご請求内容のおしらせ」メールを送信します。
5. 注文金額の売上請求した後で、OrderReference オブジェクトを **Closed** にするために CloseOrderReference 処理を呼び出します。Amazon はこの注文が完了したことを購入者に示すことに利用します。一度 **Closed** にした OrderReference オブジェクトは追加のオーソリが出来なくなり、追加の売上請求ができなくなることに注意してください。
6. 返金が必要になった場合は、返金処理を呼び出して実行できます。
7. 返金要求の処理ステータスを **Refund** IPN で確認します。Amazon は返金に成功した後で購入者へ「ご返金内容のおしらせ」メールを送信します。

次のイメージは Amazon Pay を利用した方法を説明します。



スタイルガイドライン

これらのガイドラインは Amazon アカウントでログインボタンと Amazon アカウントでお支払いボタンを利用するためのベストプラクティスを提供します。Amazon からの事前承認なしでサイトやアプリでこれらのボタンを利用でき、これらのボタンを変更することなく提供されています。

サイトに Amazon アカウントでお支払いボタンを追加

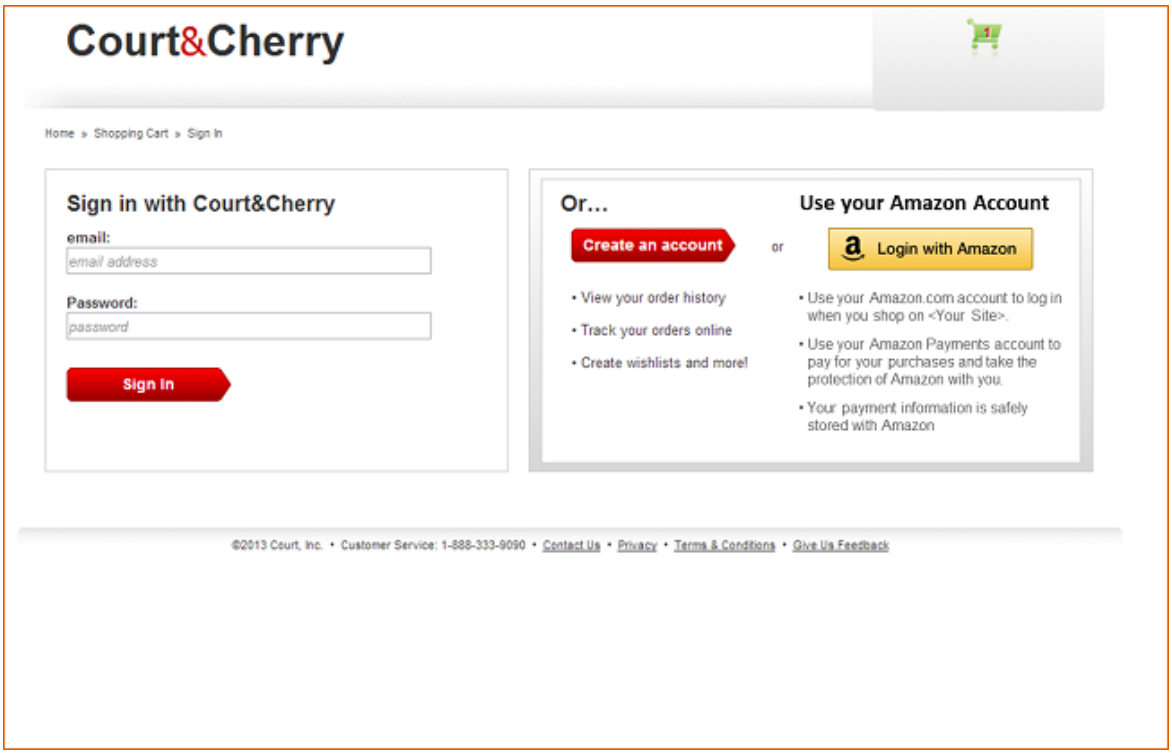
Amazon アカウントでログインボタンは、提供されているサイトやアプリでログインする所はどこにでも配置されるべきです。これは新しい顧客と同様に既存の顧客（例えば購入中）がサインアップする場所は含みます。私たちはログインを促進するためにカスタマイズされたエクスペリエンスとして、小さいボタン（以下参照）をホームページ上に表示することを推奨します。ALT テキストに“Amazon アカウントでログイン”を利用します。次がサンプルです。

ユーザーがログインしている場合は、ログインボタンは表示させず、それをログアウトメッセージに置き換えるべきです。

ボタンのガイダンスを追加

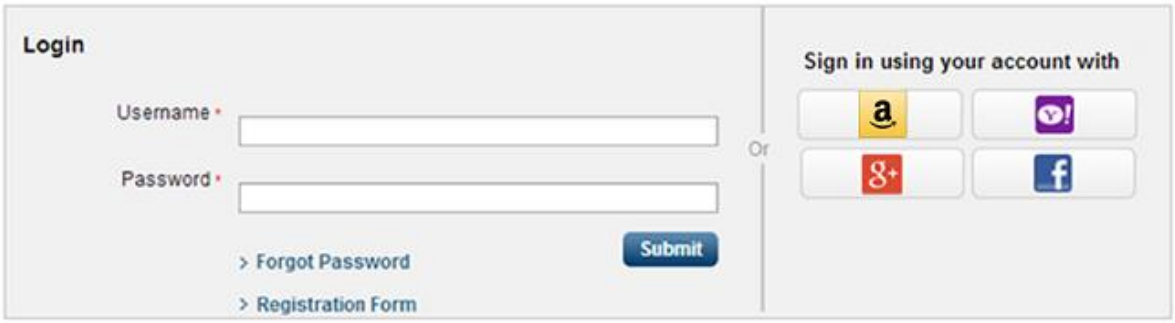
Amazon はログイン画面に次のテキストを表示することを推奨します。

- *<Your Site>*では、Amazon.co.jp アカウントでログインできます。
- お買い物をするために Amazon Pay アカウントでお支払いでき、Amazon の保護を受けられます。
- お支払い情報は Amazon に保存されているので安全です。



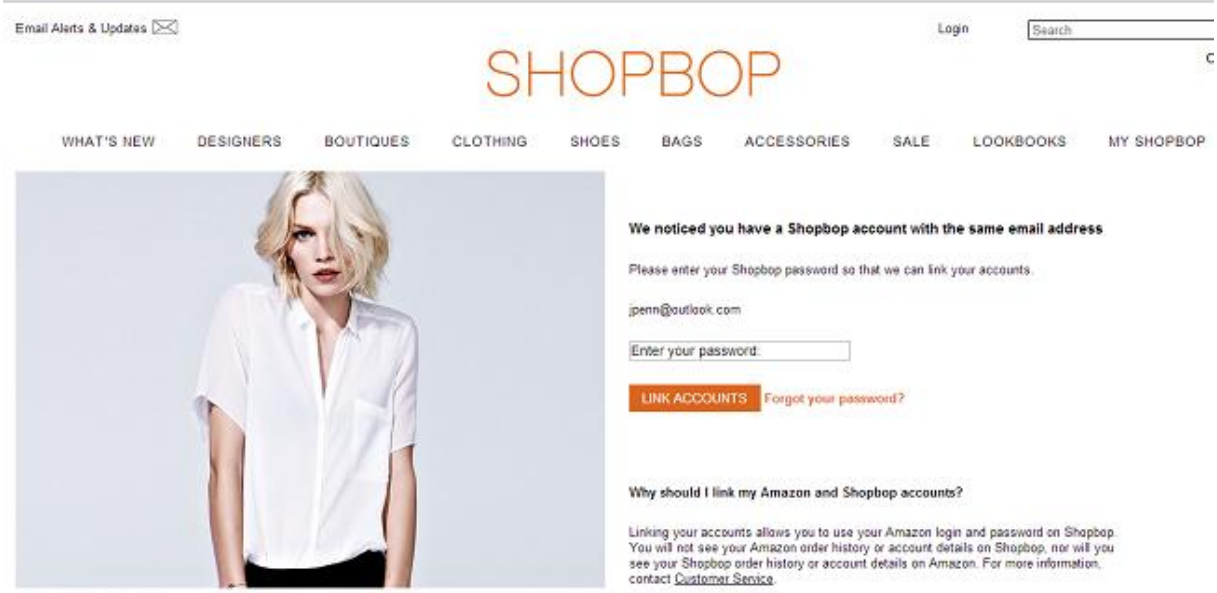
その他のサインインインテグレーションの提供

我々はシンプルなアイコンを提供しています。この画像はサイトやアプリで、異なるログインオプションをアイコン表示する時に役立ちます。



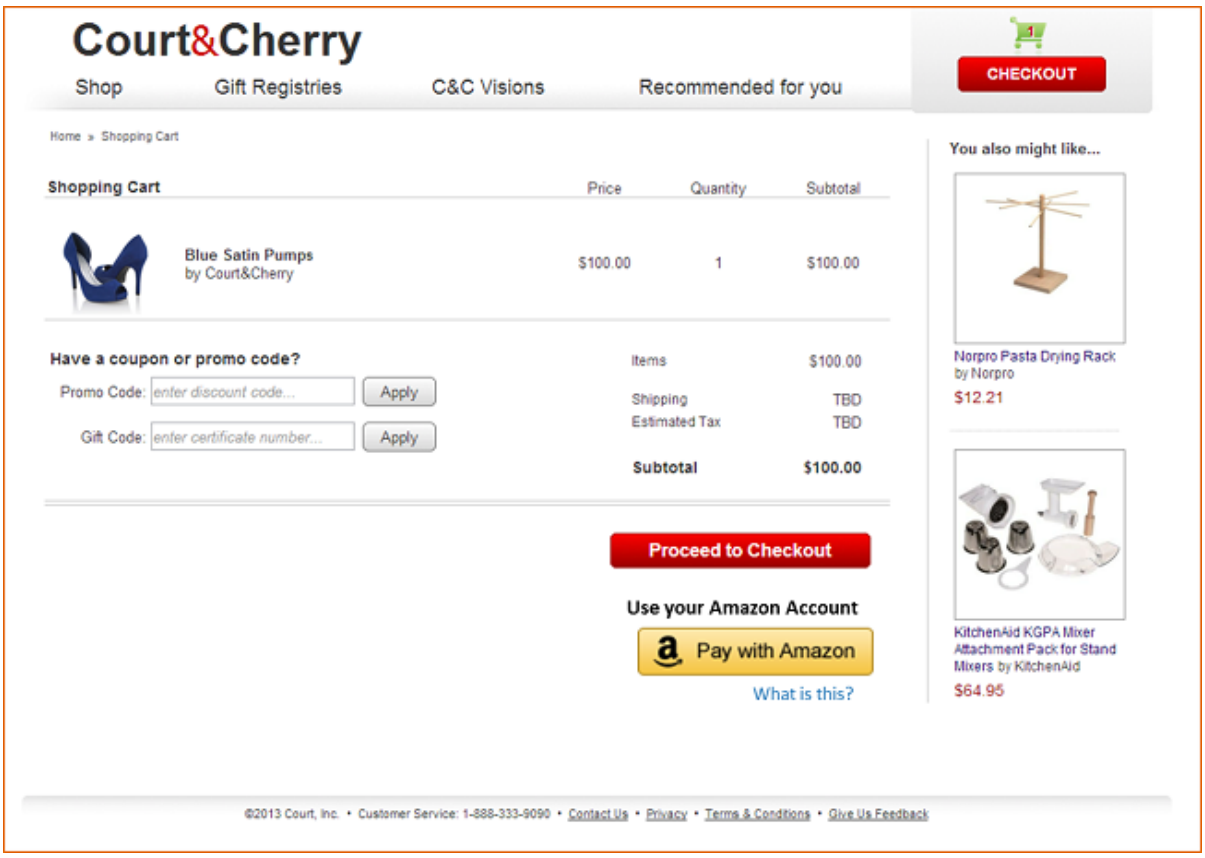
顧客アカウントを Amazon とリンク

現在の顧客が Amazon アカウントを利用してログインしている場合は、メールアドレスを識別でき、システムに登録されているアカウントとリンクできます。詳しい情報は存在するユーザーアカウントシステムとインテグレーションを参照してください。



サイトに Amazon アカウントでお支払いボタンを追加

顧客がカートページにおり、購入処理を開始するための準備されている場合は、Amazon アカウントでお支払いボタンを次のスクリーンショットのように利用できます。



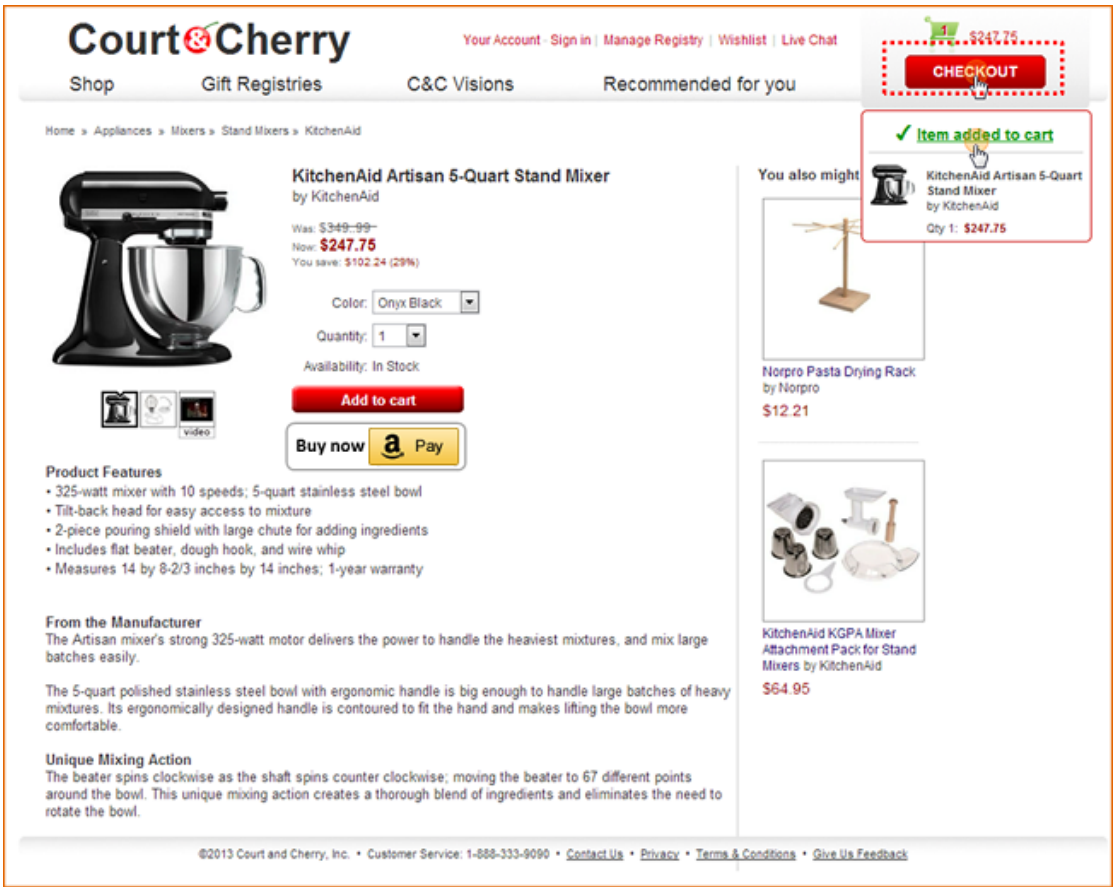
Amazon は、コンバージョン率を最大化するために、“What is this?” ポップアップを追加し、Amazon Pay とは何かを説明するために次のテキストを利用することを推奨します。

amazon pay

Amazon Pay では、<*Your Site*>の注文で Amazon アカウントに登録されている住所と支払情報を利用できます。簡単で信頼された支払方法である沢山の人々が利用しております。お客様は多くの情報を再入力する必要はなく、クレジットカードの番号と支払詳細は共有されません。Amazon Pay を選択することで、Amazon アカウントで安全にサインイン要求されます。お客様は注文を完了するために Amazon の配送先と支払方法を安全にアクセスして提供されます。
<*Your Site*>でお買い物を楽しんでください。

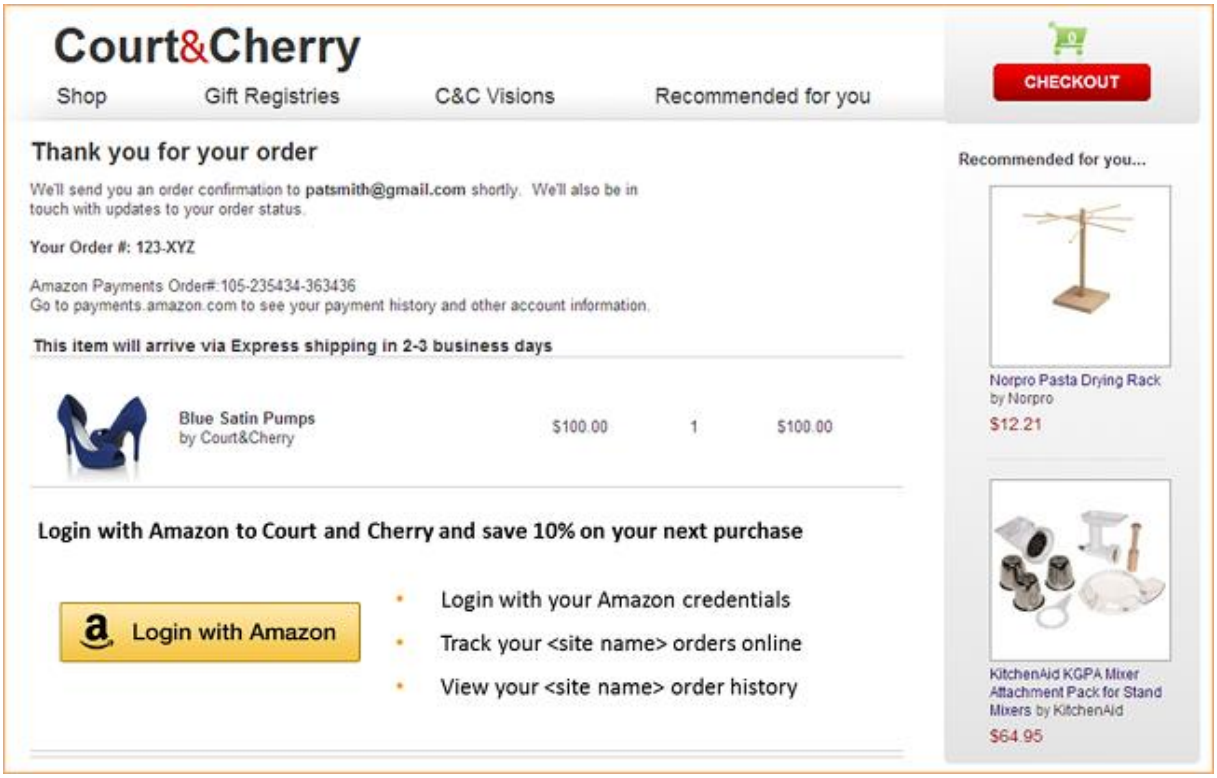
商品詳細ページに Amazon Pay を追加

次のスクリーンショットで表示しているように、Amazon アカウントでお支払いボタンを追加することで、顧客はサイトのページから望み通りの商品を素早く購入できます。



ゲスト購入後に顧客にログインを尋ねる

次のスクリーンショットで表示しているように、ゲスト購入後に素早く会員登録することを顧客に申し出ることができます。


Court&Cherry
Shop
Gift Registries
C&C Visions
Recommended for you
CHECKOUT
Thank you for your order
We'll send you an order confirmation to patsmith@gmail.com shortly. We'll also be in touch with updates to your order status.
Your Order #: 123-XYZ
Amazon Payments Order#:105-235434-363436
Go to payments.amazon.com to see your payment history and other account information.
This item will arrive via Express shipping in 2-3 business days
Blue Satin Pumps
by Court&Cherry
$100.00
1
$100.00
Login with Amazon to Court and Cherry and save 10% on your next purchase
Login with Amazon
Login with your Amazon credentials
Track your <site name> orders online
View your <site name> order history
Recommended for you...
Norpro Pasta Drying Rack
by Norpro
$12.21
KitchenAid KGPA Mixer Attachment Pack for Stand Mixers by KitchenAid
$64.95

用語集

Access scope（アクセススコープ）

アクセススコープは、クライアントが要求するユーザー個人情報の種類を定義します。最初にユーザーがログインし、アクセススコープの内容リストを確認し、クライアントが処理するためのデータを提供するか同意しなければなりません。

Access Token（アクセストークン）

アクセストークンは、ユーザーがサイトにログインした時に認証サーバーによって認証されます。アクセストークンはクライアント、ユーザーが定義されています。クライアントはアクセストークンを利用して顧客の個人情報を受け取り、配送先と支払情報へのアクセスをしなければなりません。

Allowed JavaScript origin（JavaScript の種類）

JavaScript の種類は、JavaScript 呼び出し元のプロトコル、ドメイン、ポートの組み合わせです。デフォルトでは、Web ブラウザは元のものから他のスクリプトを呼び出しことをブロックします。アプリケーションの一部として指定している場合では、Amazon ログイン SDK for JavaScript は他からの呼び出しを許可します。

Allowed Retuned URL（許可された戻り URL）

戻り URL は、Amazon ログインで利用する Web サイトのアドレスです。Amazon ログイン認証サーバーは、ログインが完了したときに、このアドレスへユーザーをリダイレクトします。詳しくは Redirect URL（リダイレクト URL）を参照してください。

Application Program Interfaces(APIs)（アプリケーションプログラムインターフェイス）

情報交換するために Amazon Pay API 処理を呼び出し、Amazon Pay と内部システム間で連携します。例えば、これらの処理を呼び出して、顧客個人情報を含めたり、購入者へ売上請求を Amazon に要求したり、返金を発行したり、購入者の配送先住所を取得したり、Order Reference をキャンセルしたりできます。詳しい情報は Amazon Pay API リファレンスガイドを参照してください。

Client Identifier（クライアント ID）

クライアント ID は、Amazon ログインに登録したクライアントに割り与えられた値です。クライアントを識別するために、認証サーバと認証承諾するためのクライアントシークレットと連結して利用します。クライアント ID はシークレットではありません。

Client Secret（クライアントシークレット）

くアイアン識別子に似ているクライアントシークレットは、Amazon ログインに登録したクライアントに割り与えられた値です。クライアントを識別するために、認証サーバと認証承諾するためのクライアントシークレットと連結して利用します。クライアントシークレットは秘匿扱いにしなければなりません。

Consent Screen（同意画面）

ユーザーが最初に Web サイトやモバイルアプリにログインした時に、個人情報をリクエストするために同意画面を表示します。同意画面は、リクエストされたアクセススコープに従って、サイトやアプリに関連した名前、ロゴイメージ、プライバシーポリシー URL を表示します。

Customer profile（個人情報）

個人情報は Amazon ログインの顧客についての情報として名前、メールアドレス、郵便番号、ユニークな ID を含みます。個人情報を含める前に、Web サイトはアクセストークンを取得する必要があります。返される個人情報はアクセススコープで定義します。

Implicit grant（Implicit 認証）

Implicit 認証は、ユーザーの Web サイトでのみで利用して完了できる認証承諾です。Implicit 認証を利用している場合は、ブラウザは URL フラグメントでアクセストークンを取得します。Implicit 認証はクライアント識別子とアクセストークンを要求します。Implicit 認証はリフレッシュトークンを返しません。

Instant Payment Notification（IPN）（インスタント支払通知）

Amazon は非同期の方法で Amazon Pay リクエスト（Authorize、Capture、Refund）を処理します。Amazon がそれぞれのリクエストを処理した後で、リクエストの最終ステータスを通知するためにインスタント支払通知を受け取ります。
さらに、Amazon Pay オブジェクトのステータスは内部システムや Amazon の内部ビジネスルールによってリクエストされた結果として変更されます。Amazon Pay オブジェ

クトの状態が変更された場合は、Amazon はシステムと状態を同期させるために IPN を送信します。詳しい情報はインスタント支払通知（IPN）を参照してください。

Login screen（ログイン画面）

ログイン画面は購入者が最初に Web サイトやモバイルアプリに Amazon ログインを利用してログインした時に表示される HTML ページです。ユーザーは存在する Amazon アカウントかログイン画面から新しいアカウントを入力することができます。

Login with Amazon（Amazon ログイン）

Amazon Pay サービスは、Amazon ログインと Amazon Pay の２つのパーツで構成されています。これらの２つパーツは技術的に密接に関連しており、同じプラットフォーム上で動作します。

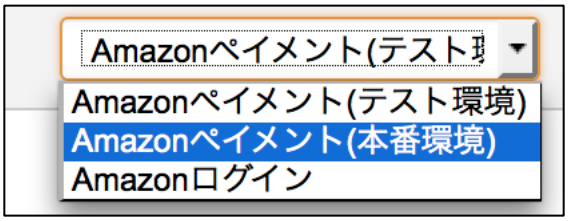
Amazon ログインは、顧客は Amazon 認証でログインすることでシステムの顧客になることを許可します。それから、Amazon Pay は、顧客のローカルアカウントを生成するための許可された顧客情報を転送します。これは、利用可能な顧客のメールアドレスを取得でき、直接コンタクトできることを意味します。

Logo image file（ロゴイメージファイル）

アプリケーションに設定された時に PNG ファイルがクライアントに提供されます。ロゴファイルは、顧客がクライアント Web サイトにアクセス認証していなかった場合に表示される同意画面上で表示されます。

Marketplace Switcher（マーケットプレイススイッチャー）

マーケットプレイススイッチャーは、セラーセントラル画面の上部近くのドロップダウンボックスであり、マーケットプレイス間を切り替えるために利用されます。例えば、マーケットプレイススイッチャーは、SANDBOX から本番環境に切り替える時に利用します。



注意：画面が小さい場合は、マーケットプレイススイッチャーのドロップダウンボックスは下記の通りのアイコンで表示されます。アイコンが表示された場合は、画面を大きくして表示することもできます。

Package name（パッケージ名）

パッケージ名は、アンドロイドアプリのユニークな ID です。パッケージ名は通常は com.companyname.appname の形式を取ります。

Amazon Pay

Amazon Pay サービスは、Amazon ログインと Amazon Pay の 2 つのパーツで構成されています。これらの 2 つパーツは技術的に密接に関連しており、同じプラットフォーム上で動作します。

Amazon Pay は、提供されるウィジェットを利用してサイトの購入フローに完全にインテグレートされます。Amazon Pay の購入フローは Web サイトの一部であり、購入者は Amazon.co.jp に登録されているアカウントに保存されている配送先住所と支払方法を選択します。

Web サイトで事前に Amazon ログインを利用することに関係なく、ゲスト購入の Amazon Pay を利用することができたり、アカウントを購入フロー中で会員登録することもできます。ゲスト購入と会員登録の両方を提供している場合は、顧客に選択肢を提供します。

Privacy notice URL（プライバシーポリシーURL）

プライバシーポリシーURL は、アプリケーション登録している時に提供します。この URL は顧客が Web サイトに最初にログインした時に確認のために表示されます。この URL はプライバシーポリシーURL にリンクされます。

Redirect URL

URL は認証サービスに提供されます。顧客がログインした後に、サービスは顧客のブラウザをこのアドレスにリダイレクトします。詳しくは許可された戻り URL を参照してください。

Signature（署名）

署名はアプリを識別するために、モバイルアプリ内に埋め込まれた MD5 ハッシュ値です。署名は通常は 01:23:45:67:89:ab:cd:ef の形式になります。