バージョン 1.0.5:2

序章

ActiveServerは**GPayments Pty Ltd**が提供する**3Dセキュア2サーバー**のソリューションです。

また、GPaymentsは、**3Dセキュア2モバイルSDK**ソリューションである**ActiveSDK**も提供しています。SDKソリューションの詳細については、当社までお問い合わせください。

ドキュメントの使い方

バージョンと言語

デフォルトで最新の**ActiveServer**のリリースのドキュメントが表示されます。メニューバーの**ドキュメントVer**のドロップダウンをクリックすることで以前のバージョンのドキュメントがご覧になれます。**最新**をクリックすると最新のドキュメントがご覧になれます。

言語のドロップダウンをクリックすると英語のドキュメント、日本語のドキュメントが確認できます。

移動方法

すべてのページの左側に**チャプターの各リスト**があります。メニューから任意の章を選択してそのページに移動することができます。ウィンドウが狭すぎる場合、**チャプターの各リスト**が表示されない場合がありますが、左上のハンバーガーのアイコンからアクセスできます。

また、各ページには、右側に**目次**が表示されます。これにはそのページのサブセクションが表示されます。リストから任意のサブセクションを選択してスキップできます。ウィンドウ幅が

狭すぎるか、単純に小さすぎる場合、**目次**が表示されない場合があります。この場合、それらが表示されるようにウィンドウのサイズを変更する必要があります。

見出しの隣にある¶アイコンをクリックして、アドレスバーにリンクをコピーすると、セクションへのリンクを作成できます。

画面右上の**検索**ボックスに語句を入力すると、ドキュメント内を検索できます。検索結果はその語句を含むすべてのページについて表示され、ドキュメントの関連部分にリンクしています。

PDFとしてダウンロード

全てのページには**ActiveServerドキュメントをダウンロード**アイコンが右上にあり、これをクリックすることでActiveServerのドキュメントをPDFとしてダウンロードすることが可能です。このアイコンはオンラインドキュメントの全てのセクションをPDFとしてダウンロードします。

> **重要**
>
> **オンラインのドキュメントが常に最新のドキュメントになります。PDFとしてダウンロードした場合、使用しているActiveServerのバージョンのオンラインのドキュメントに変更がないかを日頃からチェックされることを推奨します。**

このドキュメントの概要

このドキュメントでは、ActiveServerの紹介、統合プロセスのガイド、トラブルシューティング手順について説明します。

このドキュメントは以下の章で構成されています：

- **序章** - ActiveServerの紹介、このドキュメントの概要。
- **クイックスタート** - インストール手順、ヒント、ActiveServerを稼働させるための必須情報。
- **機能まとめ** - システムの主な機能の説明。
- **ガイド** - 3DS2タスクの為のシステム機能の使用方法とその他の機能に関する広範なガイド。

- **APIレファレンス** - APIの使用方法の概要、APIリファレンス・ドキュメントおよびエラーコードリストへのリンク。
- **用語集** - ドキュメント全体で使用される3DS2およびActiveServer固有の用語、略語、定義。
- **ドキュメントバージョン** - ドキュメントの変更ログ。
- **リリースノート** - バージョンごとのActiveServerソフトウェア・リリースノート。
- **法的通知** - 機密保持、著作権、免責事項、責任に関する声明。

製品紹介

ActiveServerは、加盟店および決済代行会社向けの3Dセキュア2の **3DSサーバー**ソリューションです。ActiveServerを使用すると、加盟店は、決済フローに対して3DS2を実装でき、ライアビリティ・シフトによる非対面取引（CNP）詐欺に対する保護を保証できます。主要なすべての国際ブランドがサポートされており、PCI-DSS 3.2に完全に対応しており、使いやすいAPIを用いて簡単に統合できます。

ActiveServerは**インハウス**の導入に柔軟に対応しており、GPaymentsの**ホスティング・サービス**からもご利用したりできます。

中核機能

ActiveServerには、以下の中核機能があります。

- **インテリジェント・レポート** - 管理Webアプリケーションから提供されるレポート機能で主要なビジネス情報が利用できます。
- **アプリケーション・サーバーとOSに依存しない** - 一般に使用されているWebコンテナを使用し、WARファイル形式でActiveServerを起動するか、またはSpringを使用してスタンドアロンアプリケーションとして展開できます。この機能は、一般的に使用されているWindowsおよびLinuxベースのシステムを含むすべてのオペレーティング・システムにまで使用できます。
- **HSMに依存しない** - Thales、Gemalto、AWS KMS、またはPKCS11互換HSMを含む、主要なほとんどの暗号化のための汎用ハードウェア・セキュリティ・モジュールと互換性があります。

- **容易な製品アクティベーション** - GPaymentsを使用する組織のアカウントにリンクされたトークンベースのアクティブ化手順で展開されたすべてのActiveServerインスタンスを簡単に管理できます。
- **複数の3DSリクエスターと加盟店** - 複数の3DSリクエスターと加盟店を同じActiveServerインスタンスに追加できます。
- **移行の容易さ** - 既存のお客様は、GPaymentsを使用してActiveServerへの移行に必要な計画を立て、移行ツールを特定します。

ActiveServer API

ActiveServerは、加盟店の既存のシステムとの統合のため、業界標準に基づいた使いやすいRESTful APIを提供します。 リクエストはJSON形式で送受信できます。APIには詳細なドキュメントとサンプルコードが付属しており、シームレスな体験を提供します。

認証API

ActiveServerは認証コンポーネントを公開しており、加盟店はブラウザとモバイルの既存のチェックアウト・プロセスにAPIコードを埋め込むことができます。このコードはActiveServerを呼び出して、認証を実行し、認証レスポンスを返します。これは柔軟なモデルであり、加盟店自身のネットワークからインターネット経由でActiveServerを遠隔操作する機能を加盟店に提供します。

管理API

ActiveServerは、システム管理者や開発者が加盟店およびアクワイアラーを既存のインフラストラクチャに統合できるようにする管理サービスへのAPIを公開しています。 管理APIは、すでに加盟店情報を維持管理している加盟店アグリゲーターおよび決済代行会社に特に役に立ちます。これにより、加盟店自身のシステム内でActiveServerの加盟店管理タスクが統合され、管理上のオーバーヘッドが大幅に削減されます。

GPaymentsについて

GPaymentsはオーストラリアに本社を置く企業で東京にも支社があります。世界中のお客様に向けて、オンライン取引に関する決済認証製品の提供に特化しています。 弊社は国際ブランド、金融機関（イシュアーとアクワイアラーの両方）、オンラインサービスプロバイダー、加

盟店、カード会員向けにさまざまなソリューションを提供しています。 弊社の3DS2アプリケーション・スイートには、ActiveAccess（ACS）、ActiveServer（3DSサーバー）、ActiveSDK（3DSモバイルSDK）が含まれています。 また、GPaymentsはお客様に対して、エンドツーエンドのシステム統合テストのために、3DS2 TestLabsへのアクセスも提供しています。TestLabsには、新しい完全に開発済みのディレクトリ・サーバー、モバイルSDK、EMVCo準拠ACSがあります。

GPaymentsの詳細については、弊社のWebサイトhttps://www.gpayments.com/jpにアクセスするか、sales@gpayments.co.jpまでお問い合わせください。

このドキュメントで誤りを見つけた場合や、追加のサポートについてお問い合わせいただく場合は、support@gpayments.co.jpのGPaymentsテクニカルサポートまで電子メールでお問い合わせください。

クイックスタート

クイックスタートは、ActiveServerのダウンロード、セットアップ、および実行を円滑に行うことを目的としたガイドです。 ActiveServerの設定や管理の方法に関する特定のユーザーガイドについては、**ガイド**セクションを参照してください。(クイックスタートの２つ下のメニュー項目にあります。)

前提条件

仕様：

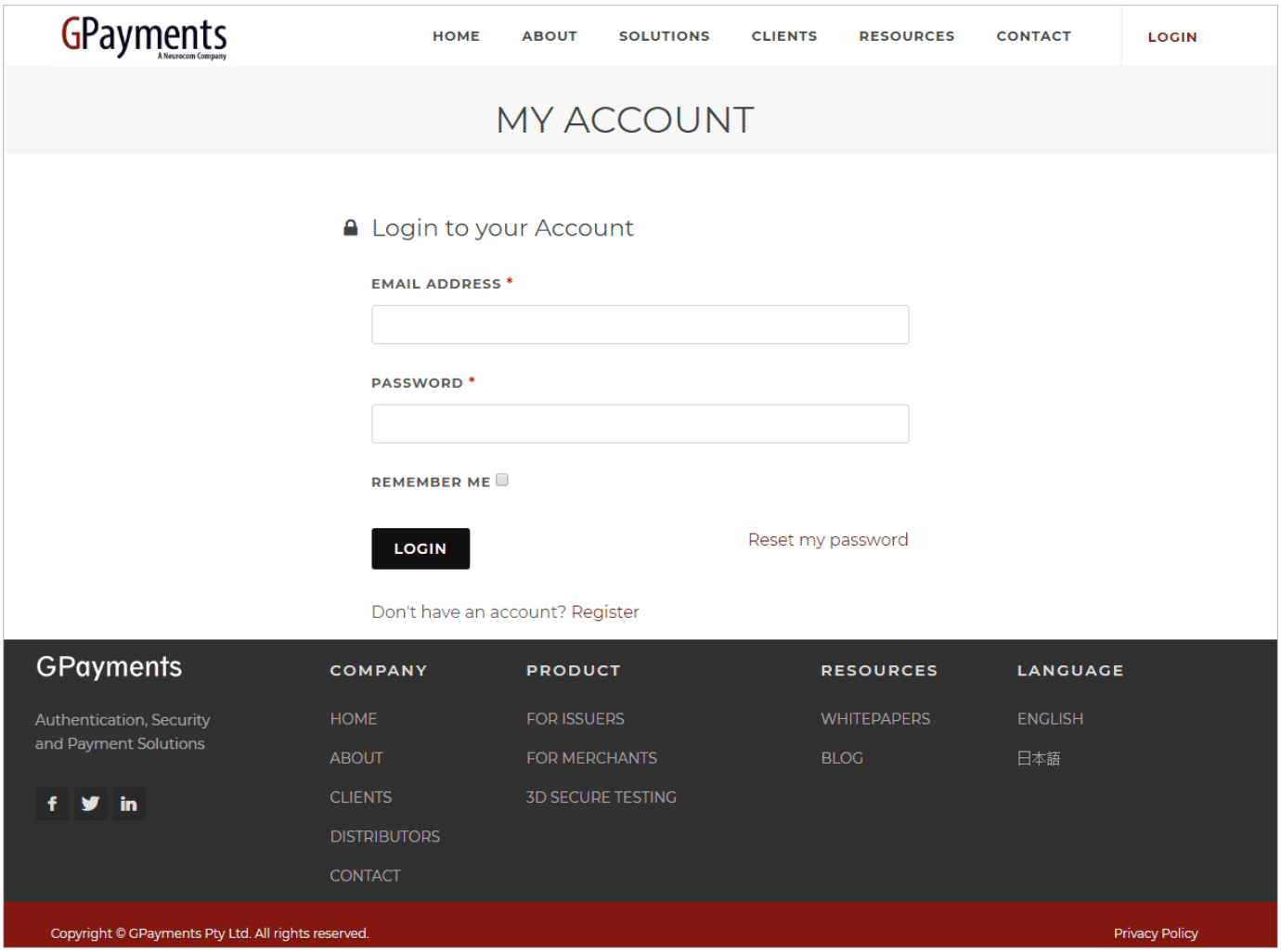
仕様	詳細
オペレーティング・システム(OS)	Linux、Windows Server
メモリ	2 GB RAM（推奨）
ディスク容量	最低要件はありませんが、すべてのデータがデータベースに格納されるため、データベース用に十分なディスク容量を用意してください。
Java Development Kit	Java SE Development Kit 8（Open JDK v1.8）
Javaコンテナ	`.jar` ファイルは、Servlet 2.4/JSP 2.0をサポートする任意のコンテナで実行できます。*デフォルト・コンテナは* `UnderTow` *です。*

データベース：

データベース	互換バージョン
MySQL/Amazon Aurora MySQL	5.7
Oracle	11g、12c
MSSQL	2008 R2、2012、2014、2016、2017

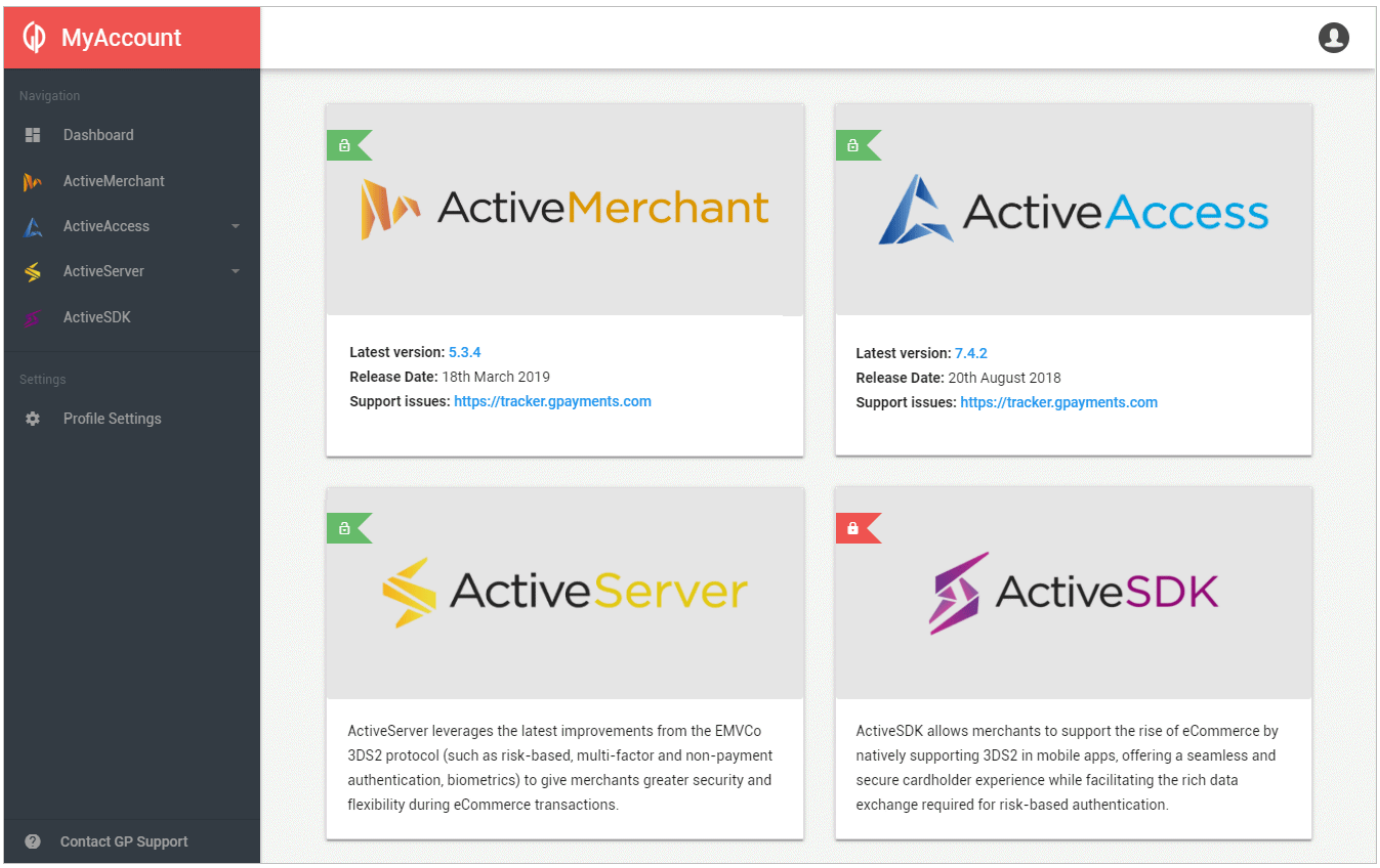
ActiveServerのダウンロード

https://login.gpayments.com/loginからGPaymentsの**MyAccount**にログインします。



アカウントがない場合は、次のURLから登録してください：https://login.gpayments.com/register

ログインすると、**MyAccountダッシュボード**が表示されます。



ActiveServer > Downloadを選択して、リリースパッケージをダウンロードします。

MyAccount 会社設定

MyAccountではユーザーが所属している会社(Organisation)によってソフトウェアへのアクセス権が得られるようになっています。ソフトウェアにアクセス権がある会社に所属しているユーザーはソフトウェアのダウンロードやインスタンスのアクティブ化、ライセンスの確認等の権限が得られます。

ソフトウェアをダウンロードする際に、会社に所属していない場合は新規の会社を**登録**するか、GPaymentsのMyAccountにすでに登録済みで会社を作成したユーザーに招待(invite)してもらう必要があります。**MyAccount > Profile Settings > My Organisation**からユーザーを招待できます。

> **重要**
>
> **ソフトウェアを既に購入しインスタンスの管理をする場合は既にMyAccountに登録された会社があるはずですので、新しい会社を登録する必要はありません。**

会社に既に所属しており、貴社が**ActiveServer**を購入している場合、パッケージのダウンロードができます。既に会社に所属しており、貴社が**ActiveServer**を購入しているのにも関わらずパッケージがダウンロードできない場合はtechsupport@gpayments.co.jp にお問合わせください。新規のお客様は弊社の営業部までsales@gpayments.co.jpお気軽にお問い合わせください。

インストール

ダウンロードした `.zip` ファイルを展開すると、以下のファイルが表示されます。

```
ActiveServer_vX.XX/
 ├─── application-prod.properties
 ├─── as.jar
 ├─── EULA.pdf
 ├─── README.txt
 ├─── release.txt
 ├─── startup.bat
 └─── startup.sh
```

ファイル：

- `application-prod.properties` - ActiveServerを初期化するための設定ファイル。
- `as.jar` - メインのActiveServer Javaパッケージ。
- `EULA.pdf` - エンド・ユーザー・ライセンス契約のコピー。
- `README.txt` - ActiveServer、ドキュメント、ライセンス、サポートに関する一般的な情報。
- `release.txt` - ActiveServerのすべてのバージョンのリリースノート。
- `startup.bat` - Windows用のスタートアップ・スクリプト。
- `startup.sh` - Linux用のスタートアップ・スクリプト。

設定

`application-prod.properties` ファイルを編集することで、ActiveServerのシステムプロパティを設定します。

アプリケーションプロパティを設定せずにActiveServerは実行する際の注意点

ダウンロードしたパッケージの `application-prod.properties` ファイルには、デフォルトのアプリケーションプロパティが含まれています。これらのデフォルト・アプリケーションプロパティを使用してActiveServerのインスタンスを起動すると、ActiveServerは：

- 一時的にのみRAMに保存され、ActiveServerがシャットダウンするとクリアされるデフォルト・データベースを使用します。
- `${AS_HOME}/conf/security/` にローカルに保存されているSun JCEを使用してキーストア・ファイルを作成します。
- 電子メールサーバー設定をスキップするため、システムがユーザーに電子メール通知を送信しなくなります。

デフォルト・プロパティを使用すると、設定を変更する必要なくActiveServerのインスタンスを迅速に起動できるため、ソフトウェアやインターフェイスを試す際に便利ですが、本番環境のインスタンスをセットアップする前にアプリケーションプロパティを設定する必要があります。

デフォルト・アプリケーション・プロパティを変更するには：

ファイル `application-prod.properties` を開き、各関連パラメータに関連付けられている対応する値を変更します。

システムプロパティには4つのカテゴリーがあります。

- データベース設定
- Webサーバー設定
- キーストア設定
- 電子メールサーバー設定

データベース設定

ActiveServerでは、以下のデータベースがサポートされています。

- MySQL
- Oracle
- MS SQL

各データベースには、設定可能な以下の一連のプロパティがあります。

- `as.db.vendor=`
  データベース・ベンダー/タイプ。デフォルト値は空であり、メモリ内のテスト・データ

ベースが使用されます。メモリ内のテスト・データベースを使用して本番環境に移行することはできません。可能な値は、`mysql`、`oracle`、または`sqlserver`です。

- `as.db.url=`
  データベースへの接続に使用されるデータベース接続URL。URLはJDBC形式である必要があります。
- `as.db.username=`
  データベースユーザー名。データベース管理者によって設定されたユーザー名を入力します。
- `as.db.password=`
  パスワード。データベース管理者によって設定されたパスワードを入力します。
- `as.db.password-plain=`
  上記のパスワードを暗号化するか否か。`application-prod.properties`ファイルに格納されている場合、ActiveServerはパスワードを暗号化できます。パスワード暗号化を有効化するには、`false`と入力します。パスワードをプレーン・テキストのままにするには、`true`と入力します。

MySQL

MySQLデータベースを使用するには、以下のプロパティが必要です。

MySQL データベース・プロパティー(例)

```
as.db.vendor=mysql
as.db.url=jdbc:mysql://<SQL DB ホスト名を入力>:3306/<DB名を入力>?
useUnicode=true&characterEncoding=utf8&useSSL=false
as.db.username=<user name>
as.db.password=<password>
as.db.password-plain=true
```

Oracle

Oracleデータベースを使用するには、以下のプロパティが必要です。

Oracle データベース・プロパティー(例)

```
as.db.vendor=oracle
as.db.url=jdbc:oracle:thin:@//<Oracle DBホスト名を入力>:1521/<Oracle DB名を入力>
as.db.username=<user name>
as.db.password=<password>
as.db.password-plain=true
```

MS SQL

MS SQLデータベースを使用するには、以下のプロパティが必要です。

MS SQL データベース・プロパティー(例)

```
as.db.vendor=sqlserver
as.db.url=jdbc:sqlserver://<MS SQL DBホスト名を入力>:1433
as.db.username=<user name>
as.db.password=<password>
as.db.password-plain=true
```

Webサーバー設定

Webサーバー設定では、デフォルトのサーバー・ポートやその他のネットワーク関連の値を設定できます。ネットワーク・セットアップによっては、HTTPを使用するかHTTPSを使用するかを選択できます。HTTPを使用する場合、エントリー・ポイントがインターネットからアクセス可能である必要があるため、HTTPSトラフィックやSSLターミネーションを処理するためにロードバランサーまたはリバースプロキシが必要です。

サーバーポート、プロトコル、SSL設定

```
## サーバーポート、プロトコル、SSL設定
## protocol http|https|both
as.server.protocol=http
as.server.http.port=8080
as.server.https.port=8443
as.server.https.key-store=<キーストアファイルパスを入力>
## キーストア・タイプ、 pkcs12またはjks
as.server.https.key-store-type=pkcs12
as.server.https.key-store-password=<キーストアファイルのパスワードを入力>
##trueにセットすることでコネクターを作成します
# as.server.enabled=false
```

- `as.server.https.key-store=`
  キーストア・ファイルパス。HTTPSの場合、キーストアが必須です。キーストアには、指定されたHTTPSコネクターのサーバー証明書が含まれている必要があります。本番環境のインスタンスの場合は、サーバー証明書がCAによって商業的に署名されている必要があることに注意してください。
- `as.server.https.key-store-type=`
  キーストア・タイプ。ActiveServerでは、2種類のキーストアがサポートされています。可能な値は、 `pkcs12` 、または `jks` です。CAが発行した商業的に署名された証明書は、通常「pkcs12」形式であり、ファイル拡張子は `.p12` または `.pfx` です。
- `as.server.enabled=`
  サーバー・コネクターを有効化または無効化します。サーバー・コネクターを有効化するには、 `true` と入力します。サーバー・コネクターを無効化するには、 `false` と入力します。

ネットワーク設定によっては、個別のポートで管理アクセスをセットアップすることをお勧めします。そのためには、以下の設定を適用する必要があります。デフォルトでは、管理ポート番号が無効化されています。有効化する場合、以下で設定したポート番号が他のポート番号と競合しないようにしてください。

管理ポート (例)

```
# 管理ポート、プロトコル、SSL設定
# 管理ポートの設定のデフォルトはサーバーポートの設定になります
# trueにセットすることでコネクターを作成します
as.admin.enabled=false
as.admin.http.port=9090
as.admin.https.port=9443
#プロトコル http|https|both
as.admin.protocol=http
as.admin.https.key-store=<キーストアファイルパスを入力>
#キーストア・タイプ、 pkcs12またはjks
as.admin.https.key-store-type=pkcs12
as.admin.https.key-store-password=<キーストアファイルのパスワードを入力>
```

認証および管理APIのポート。相互認証のHTTPSでのみ利用できます。このポートは、ActiveServerインスタンスがアクティブ化されると有効になります。このポートを有効化するには、サーバーの再起動が必要です。

認証APIポート (例)

```
#認証APIポート, HTTPSのみ設定可能
as.api.port=7443
```

以下のディレクトリ・サーバー・コネクター設定は、ActiveServerが相互認証でディレクトリ・サーバーとリクエストを送受信するためのものです。これらのコネクターでは常にHTTPSが有効です。ディレクトリ・サーバーのサーバーおよびクライアント証明書は、DS証明書の管理で説明されているように、後で設定できます。

> **Note**
>
> 管理者UIでの証明書の設定が完了したら、サーバーの再起動が必要です。

各ディレクトリ・サーバーには、設定可能な以下の一連のプロパティがあります。

- `as.<Card Scheme>.port=`

各国際ブランドのディレクトリ・サーバーに対してリスンするポート番号です。デフォルト値は以下にあります。

> **Note**
>
> ポート番号が他のポート番号と競合しないようにしてください。

- `# as.<Card Scheme>.enabled=false`

  このパラメータはデフォルトでコメント化されています。ディレクトリ・サーバー・コネクターのステータス（無効または有効）を決定します。
  ディレクトリ・サーバー・コネクターを無効化するには、`false`と入力します。そうでない場合は、コメント化したままにします。

American Express

American Expressディレクトリ・サーバープロパティ(例)

```
as.amex.port=9600
## falseにセットすることでDSのリスニングポートはオフになります
# as.amex.enabled=false
```

China UnionPay

China UnionPayディレクトリ・サーバープロパティ(例)

```
as.chinaunionpay.port=9601
## falseにセットすることでDSのリスニングポートはオフになります
# as.chinaunionpay.enabled=false
```

China UnionPayのサポートは今後のリリースで追加されます。

Discover / Diners Club International

Discover / Diners Club Internationalディレクトリ・サーバープロパティ(例)

```
as.discover.port=9602
## falseにセットすることでDSのリスニングポートはオフになります
# as.discover.enabled=false
```

JCB

JCB ディレクトリ・サーバープロパティ(例)

```
as.jcb.port=9603
## falseにセットすることでDSのリスニングポートはオフになります
# as.jcb.enabled=false
```

Mastercard

Mastercard ディレクトリ・サーバープロパティ(例)

```
as.mastercard.port=9604
## falseにセットすることでDSのリスニングポートはオフになります
# as.mastercard.enabled=false
```

Visa

Visa ディレクトリ・サーバープロパティ(例)

```
as.visa.port=9605
## falseにセットすることでDSのリスニングポートはオフになります
# as.visa.enabled=false
```

キーストア設定

ActiveServerでは暗号化キーの格納のオプションを3つ提供しています。

- ローカル・キーストア（Sun JCE）
- Amazon S3キーストア
- PKCS11 HSM

以下のプロパティを使用してキーストア・タイプを設定します。

```
as.keystore.type=<キーストア・タイプを入力>
```

- `as.keystore.type=`
  キーストア・タイプ - 可能な値は、 `local` 、 `s3` 、 `pkcs11` です。

ローカル・キーストア（Sun JCE）

ローカル・キーストア・ファイルを使用するには、以下のプロパティを使用します。

ローカル・キースとトア(Sun JCE) (例)

```
as.keystore.local.path=${AS_HOME}/security/keystores/
```

- `as.keystore.local.path=`
  キーストア・ファイルパス。キーストア・ファイルのファイルパスを入力します。これは、キーストア・ファイルを含むフォルダを参照している必要があります。

Amazon S3キーストア

ActiveServerでは、キーストアとしてのAmazon S3の使用がサポートされています。Amazon S3キーストアを使用するには、*AWS Bucket*、*AWS Region*、*AWS Credentials*設定を設定する必要があります。

AWSバケット

以下のプロパティでAWSバケットパスを設定します。

Amazon S3キーストア(例)

```
as.keystore.s3.bucket-name=<Amazon S3バケツ名を入力>
...
```

AWSのリージョン

AWS Regionはいくつかの方法で設定できます。リージョン・コードのリストは以下の表の*Region*列にあります：Amazon AWS - Regions and Availability Zones。

1.以下のプロパティでAWSのリージョンコードを設定します。

```
``` properties tab="Amazon S3キーストア(例)"
...
as.keystore.s3.region=<Amazon S3のリージョンコードを入力>
...
```
```

2.あるいは、ローカル・システムのAWS設定ファイルでAWSのリージョンを設定します。設定ファイルは以下の場所にあります：Linux、macOS、Unixの場合は `~/.aws/config` 、Windowsの場合は `C:\Users\USERNAME\.aws\config` 。このフィールドには、以下の形式で行を含める必要があります。

```
```
[default]
region = <S3のリージョンコード>
```
```

AWS認証情報

AWS認証情報には `access_key_id` および `secret_access_key` が含まれます。AWS認証情報はいくつかの方法で設定できます。

1.以下のプロパティでAWS認証情報を設定します。

```
``` properties tab="Amazon S3キーストア(例)"
...
as.keystore.s3.credentials.access-key-id=<Amazon S3のアクセスキーIDを入力>
as.keystore.s3.credentials.secret-access-key=<Amazon S3の秘密アクセスキーを入力>
...
```
```

2.あるいは、ローカル・システムのAWS認証情報プロファイル・ファイルでAWS認証情報を設定します。認証情報プロファイル・ファイルは以下の場所にあります：Linux、macOS、Unixの場合は `~/.aws/credentials` 、Windowsの場合は `C:\Users\USERNAME\.aws\credentials` 。このフィールドには、以下の形式で行を含める必要があります。

```
```
[default]
aws_access_key_id = <Amazon S3のアクセスキーIDを入力>
aws_secret_access_key = <Amazon S3の秘密アクセスキーを入力>
```
```

3.あるいは、AWS EC2インスタンス上に**ActiveServer**を展開する場合、IAMロールを指定してからそのロールにEC2インスタンス・アクセス権を付与できます。この場合、Amazon AWS - Using IAM Roles to Grant Access to AWS Resources on Amazon EC2ガイドに従う必要があります。

PKCS11 HSM

ActiveServerでは、キーストアとしてのHSMの使用がサポートされています。HSMはPKCS11 APIをサポートする必要があります。PKCS11 HSMでハードウェア暗号化を使用するには、以下のプロパティが必要です：

PKCS11 HSM (例)

```
as.keystore.pkcs11.library=<pkcs11ドライバーのライブラリーを入力>
as.keystore.pkcs11.slot=<スロット番号を入力>
```

- `as.keystore.pkcs11.library=`
  HSMドライバ・ライブラリ。Linuxの場合、これは通常 `.so` ファイルです。Windowsの場合、これは通常 `.dll` ファイルです。使用する必要があるライブラリの詳細については、HSMのドキュメントを参照してください。

- `as.keystore.pkcs11.slot=`
  HSMのスロット番号。使用する必要があるスロット番号の詳細については、HSMのドキュメントを参照してください。

> **Info**
>
> HSMのセットアップと設定はこのドキュメントの対象外であることに注意してください。ActiveServerで設定する前に、HSMが完全に機能していることを確認してください。

電子メールサーバー設定

ActiveSeverでは、ユーザーに電子メール通知を送信できます。電子メール通知は、アクティブ化URLをユーザーに通知したり、ライセンスの期限が近づいたらユーザーにリマインドしたりするのに使用できます。

電子メール通知をセットアップするには、認証情報と関連付けられている電子メールアカウントとサーバー詳細が必要です。

Email Server Properties (例)

```
as.mail.host=<SMTPサーバーホストを入力>
as.mail.port=<SMTPサーバーポートを入力>
as.mail.user-name=<Eメールアドレス>
as.mail.password=<Eメールパスワード>
as.mail.auth=true
as.mail.start-tls=true
```

- `as.mail.host=`
  電子メールサーバーのSMTPドメイン。
- `as.mail.port=`
  電子メールサーバーのポート番号。
- `as.mail.user-name=`
  電子メールの送信元のアカウントの電子メールアドレス。
- `as.mail.password=`
  電子メールアカウントのパスワード。
- `as.mail.auth=`
  電子メールアカウントにSMTP認証が必要かどうか。電子メールアカウントに認証が必要な

場合、`true`と入力します。そうでない場合は、`false`と入力します。よくわからない場合は、詳細について電子メールサーバーの管理者と相談してください。

- `as.mail.start-tls=`
  SMTPサーバーにTLSが必要かどうか。SMTPサーバーにTLSが必要な場合、`true`と入力します。そうでない場合は、`false`と入力します。よくわからない場合は、詳細について電子メールサーバーの管理者と相談してください。

> **Info**
>
> 電子メールサーバーのセットアップと設定はこのドキュメントの対象外であることに注意してください。
> ActiveServerで設定する前に、電子メールサーバーが完全に機能していることを確認してください。

ActiveServerの起動

すべてのプロパティが正しく設定され、ActiveServerのインスタンスを起動できるようになりました。

Linuxを使用している場合は、**ターミナル**を開きます。Windowsを使用している場合は、**コマンドプロンプト**を開きます。

スタートアップ・スクリプト（`.sh`または`.bat`ファイル）が含まれているフォルダに作業ディレクトリを変更します。

以下のコマンドを使用してActiveServerを起動できるようになりました。

Linux Windows

```
./startup.sh
```

> ✕ **`as.jar`ファイルがスタートアップ・スクリプトと同じディレクトリ内にあることを確認してください**
>
> `as.jar`ファイルがスタートアップ・スクリプトと同じディレクトリにないと、スタートアップ・コマンドは動作しません。

ターミナルまたは**コマンドプロンプト**に以下の出力が表示されるはずです。次のステップで必要になるため、**Administration**URLをメモします。

ActiveServerインスタンスの情報

```
--------------------------------------------------------------------------------

ActiveServer by GPayments

_______        _____ _____                 ________
___    |_________  /____(_)___   _______ __  ___/_____ ___________   _______
________
__  /| |_  ___/_  __/__  / __ | / /_  _ \_____ \ _  _ \__  ___/__ | / /_  _
\__  ___/
_  ___ |/ /__  / /_  _  /  __ |/ / /  __/____/ / /  __/_  /    __ |/ / /  __/
_  /
/_/  |_|\___/  \__/  /_/   _____/  \___/ /____/  \___/ /_/     _____/  \___/ /_/

--------------------------------------------------------------------------------

.
.
.

--------------------------------------------------------------------------------
        ActiveServer by GPayments is up and running.

        Version:                    1.0.0
        Git Commit Id:              da369ec

        Activation:                 NOT ACTIVATED, please contact GPayments
        Authentication API Port:    7443

        Server:                     http://10.0.75.1:8081
        Administration:             http://10.0.75.1:8081

        Key Store Type:             SUNJCE

        Profile(s):                 [prod]
--------------------------------------------------------------------------------
```

スタートアップ・スクリプト

スタートアップ・スクリプトでは、環境変数 `AS_HOME` が、 `application-prod.properties` が存在するディレクトリに設定されています。ActiveServerは、 `AS_HOME` を使用して、設定ファイルを探したり、キーストアを管理したり、ログを出力したりします。別の場所を参照するように `AS_HOME` を設定すると、同じサーバーで複数のActiveServerインスタンスを実行できます。これは、別のディレクトリにパッケージをコピーするか、別のスタートアップ・スクリプトを作成

して、それらのスクリプトで別の場所を参照するように `AS_HOME` を設定することで実行できます。

> **Note**
>
> 同じサーバーに複数のインスタンスがある場合、 `application-prod.properties` ファイルのいずれかでポート番号が競合しないようにしてください。

ActiveServerプロファイル

`AS_HOME` の設定に加えて、スタートアップ・スクリプトは、環境変数 `AS_PROFILES` も設定します。これは、プロファイルベースの設定を指定するための便利な仕組みです。

デフォルトでは、プロファイルは `prod` に設定されています。

ActiveServerはパターン `application-<profile>.properties` を使用して、プロファイルの設定ファイルを読み込んでいます。そのため、デフォルトの `prod` プロファイルでは、`application-prod.properties` が読み込まれます。ただし、新しいプロファイル（ `test` など）を作成して、ActiveServerの異なる設定をセットアップできます。

新しいプロファイルを作成するには：

- `application-test.properties` という名前の新しい設定ファイルを作成し、 `prod` 設定ファイルと同じディレクトリに貼り付けます。
- スタートアップ・スクリプトを開き、 `AS_PROFILES` の値を `test` に設定します。

  ActiveServerは、古い `prod` プロパティの代わりに新しいプロファイルを読み込みます。

  ActiveServerは同時に複数のプロファイルを読み込むこともできます。このためには、`AS_PROFILES` の値を `prod,test` に設定することで、ActiveServerが `prod` と `test` の両方のプロファイルからプロパティ・ファイルを読み込みます。

  これらのオプションが利用可能な場合、個別の `.properties` ファイルで別々に設定を管理できます。

> **Tip**
>
> すべてのデータベース設定を `application-db.properties` で、Webサーバー設定を `application-web.properties` で管理する場合、 `AS_PROFILES` の値を `db,web` に設定すると、さまざまな管理者が管理するためのActiveServer設定を提案できます。

セットアップ・ウィザード

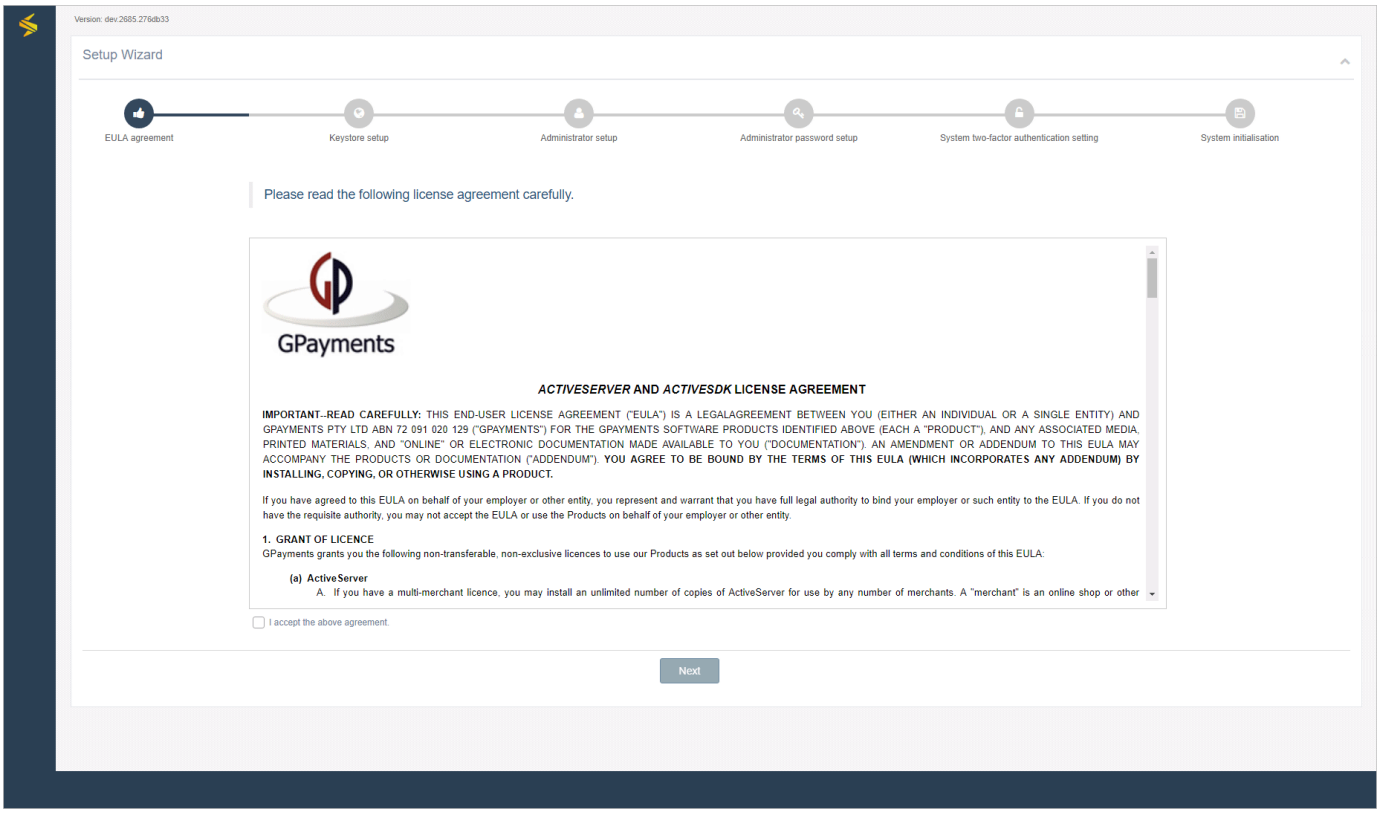
ActiveServerが稼働すると、**Administration** URLから管理者UIにアクセスできます。

初めてActiveServerを実行する場合、セットアップ・ウィザードが表示され、セットアップ・プロセスがガイドされます。

セットアップ・ウィザードには、以下の手順が含まれます。

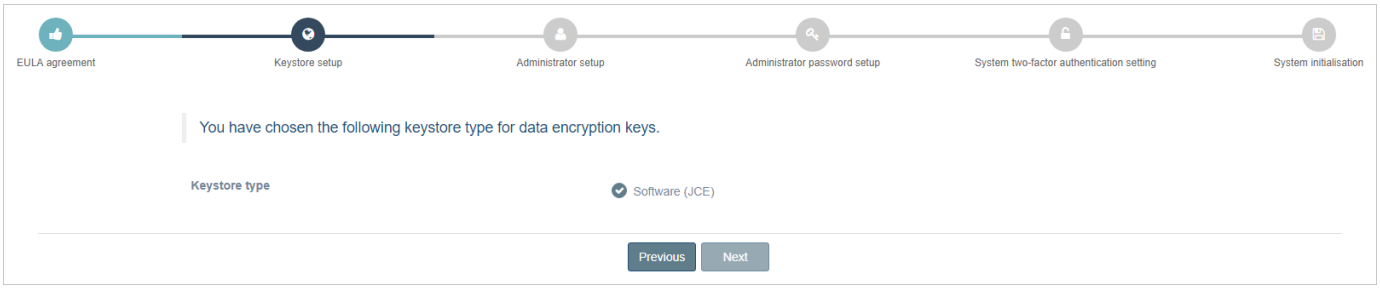
- EULA契約
- キーストア・セットアップ
- 管理者セットアップ
- 管理者パスワード・セットアップ
- システム2要素認証設定
- システム初期化

EULA契約



EULA契約を確認します。利用契約に同意する場合は、**I accept the above agreement.**チェックボックスを選択して進んでください。

キーストア・セットアップ

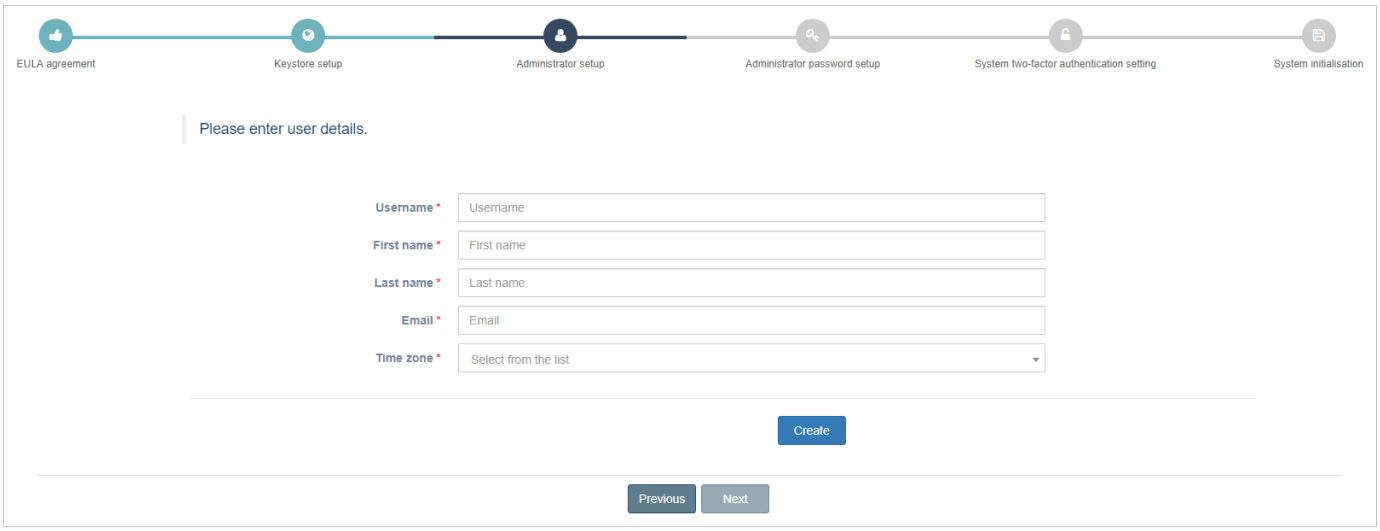


- **Keystore type**を選択します。

  セットアップ中に**Software**が選択された場合、SUN JCEが使用されます。

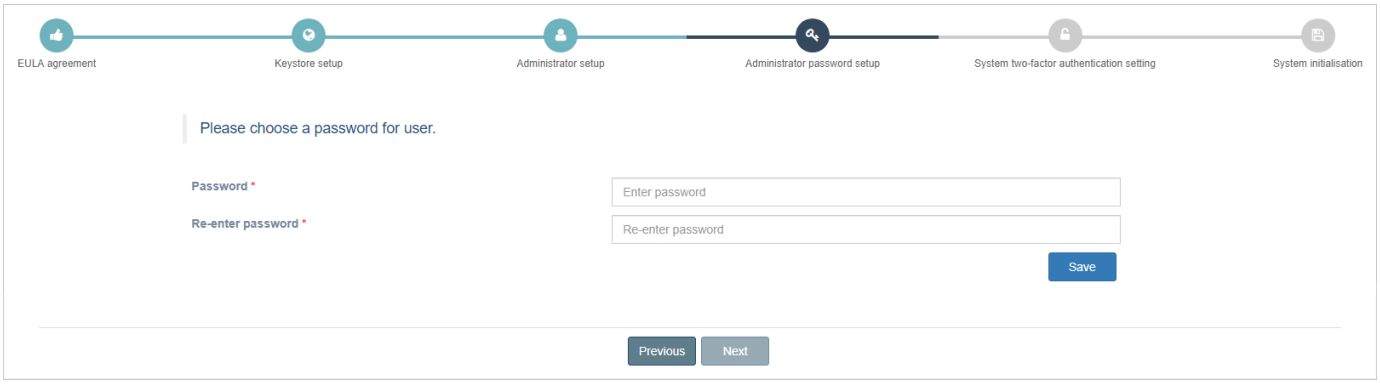
  `application-prod.properties` ファイルに適切な詳細を入力することで、PKCS#11 HSMを使用するオプションも利用できます。PKCS#11 HSMのセットアップ方法については、暗号化モジュールを参照してください。
- 続行するには、***次***ボタンを選択します。

管理者セットアップ



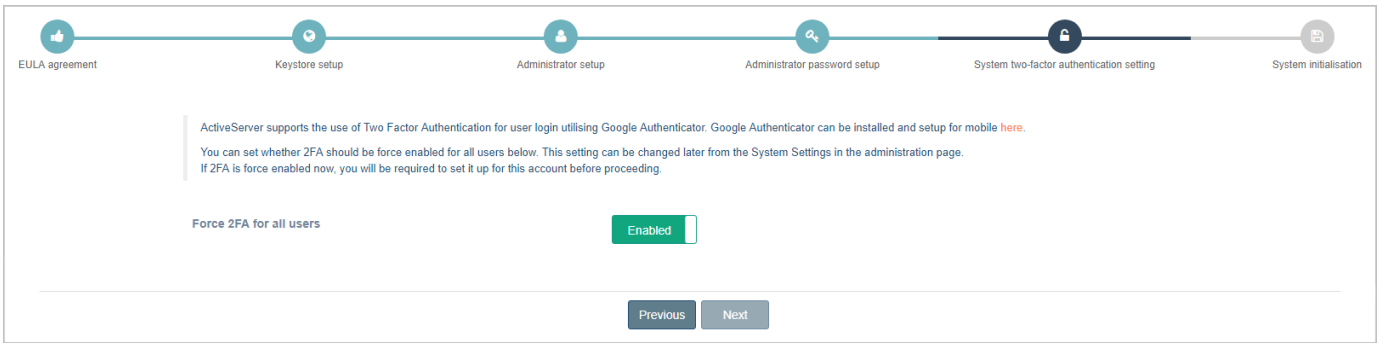
- 管理者アカウントのユーザー詳細を入力します。
- ***Create***ボタンを選択すると、アカウントが作成されます。
- 続行するには、***次***ボタンを選択します。

管理者パスワード・セットアップ



- 管理者アカウントのパスワードを入力します。
- パスワードを再入力して確認します。
- ***Save***ボタンを選択すると、パスワードが作成されます。
- 続行するには、***次***ボタンを選択します。

システム2要素認証設定



ActiveServerでは、管理者UIへのサインイン用に2要素認証がサポートされています。

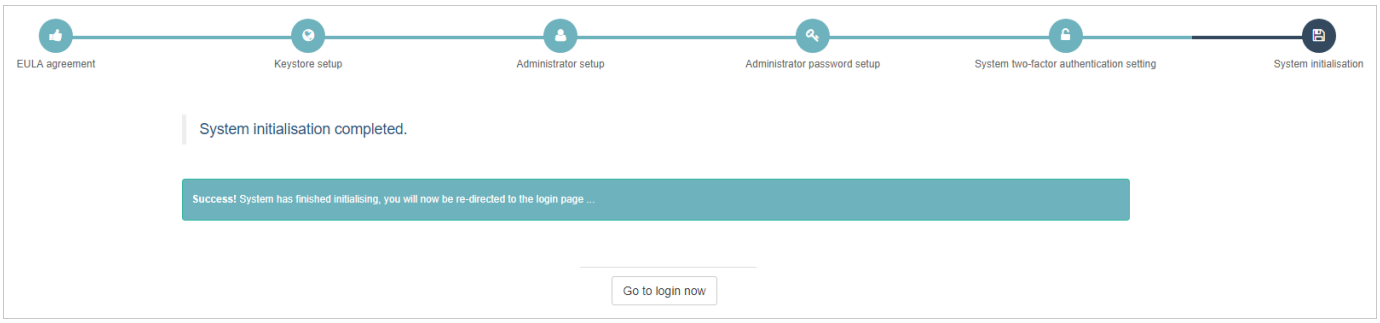
> **Note**
>
> この機能を使用するには、モバイル・デバイスにGoogle認証システムがインストールされている必要があります。
>
> Google Authenticatorのセットアップ手順については、Google Authenticatorをインストールを参照してください。

デフォルトでは、ActiveServerはユーザーに2要素認証の使用を強制しません。

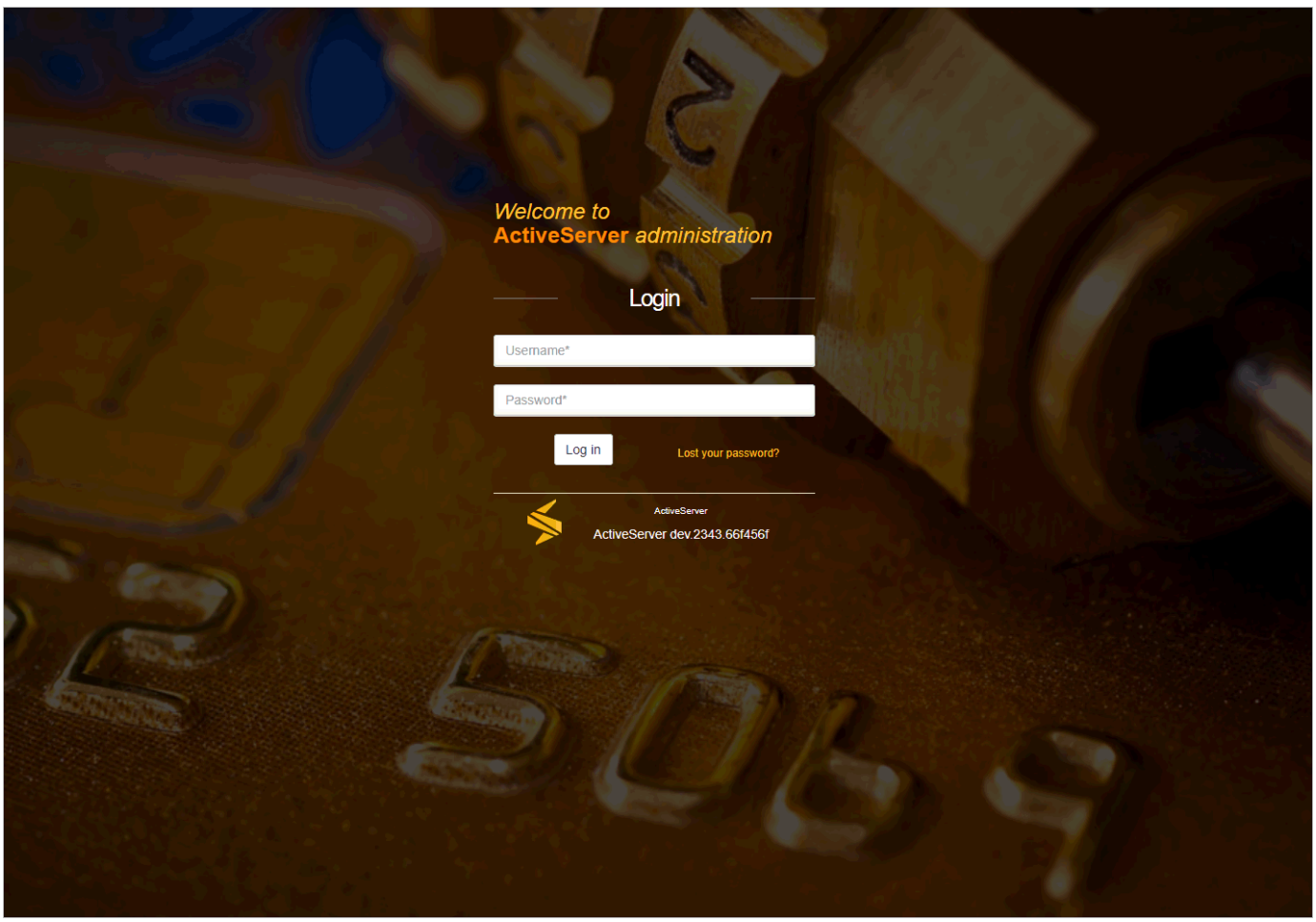
すべてのユーザーに2要素認証を強制するには：

- **Force 2FA for all users**に隣接するトグルを***Enable***にします。
- 続行するには、***次***ボタンを選択します。

システム初期化



セットアップ・ウィザードは、システムの初期化が完了したことを通知し、ActiveServerのログインページにリダイレクトします。



次の手順

この後は、以下を実行できます。

- ActiveServerのアクティブ化
- システム設定の管理
- ActiveServerのAPIを統合する

機能まとめ

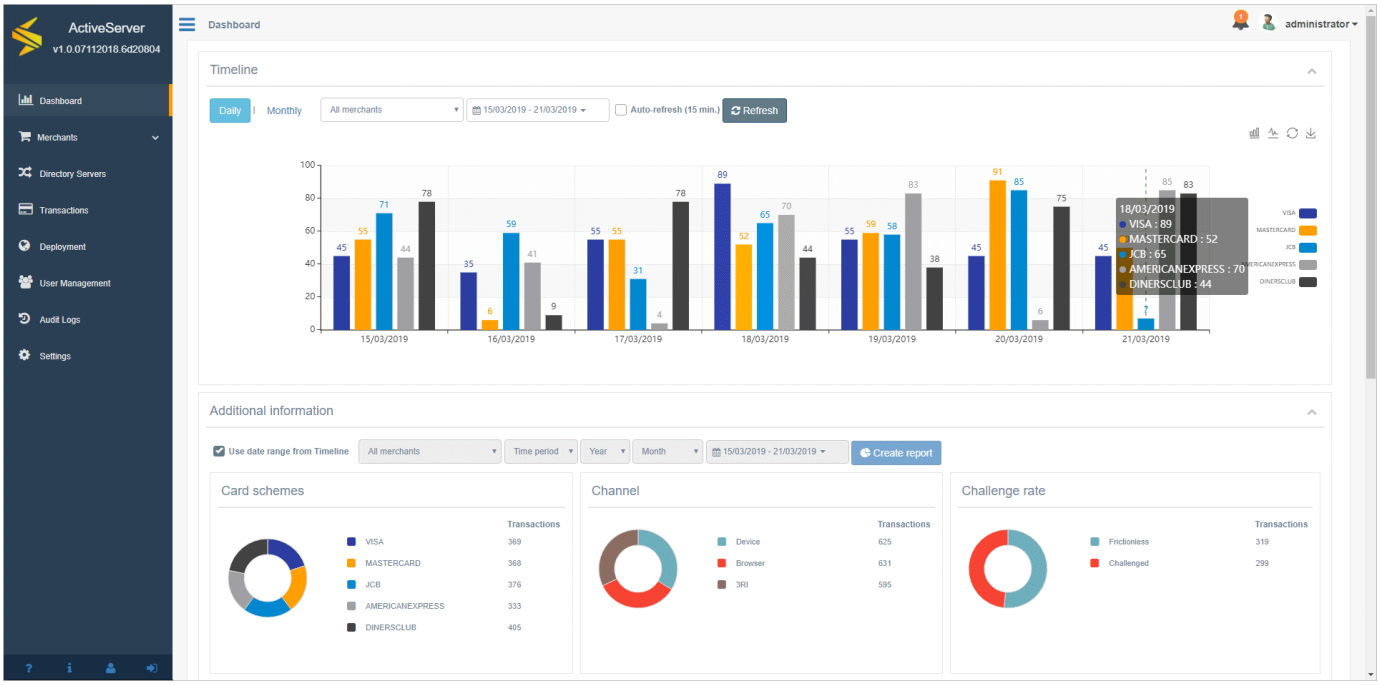
以下はユーザー・インターフェイスの概要を含む **ActiveServer**のすべての機能の概要です。各セクションの下には、タスクの実行に役立つ関連ガイドへのリンクがあります。

ダッシュボード

Dashboardには、認証の統計グラフが表示されます。これらの統計情報は、期間をカスタマイズして使用することができ、システム全体の統計を表示したり、加盟店ごとに分割したりすることができます。

ダッシュボードは加盟店の管理用に設計されたロール(**Business Admin**、**Merchant Admin**、**Merchant** など)のユーザーにのみ表示されます。

詳細については、*ガイド* > *管理UI* > ダッシュボードの使い方を参照してください。



加盟店

Merchantsには2つのセクションがあります： **Merchants**および **Acquirers**。

Merchantsページは、**ActiveServer**の加盟店エンティティを管理するのに使用されます。加盟店を *作成*、 *検索*、 *表示*、 *編集*、 *削除*できます。これは、3DSリクエスター **クライアント証明書**をダウンロードする場所でもあり、加盟店の暗号化キーをローテーションできる場所でもあります。

Acquirersページは、**ActiveServer**のアクワイアラー・エンティティを管理するのに使用されます。3DS2認証リクエスト用に加盟店プロフィールを割り当てる前に、アクワイアラーを *作成*、*検索*、*表示*、*編集*、*削除*できます。

加盟店ページは加盟店の管理用に設計されたロール(**Business Admin**、**Merchant Admin**、**Merchant**ロールなど)のユーザーにのみ表示されます。

詳細については、*ガイド* > *管理UI*以下を参照してください：

- 加盟店を検索
- 加盟店を管理
- アクワイアラを管理

ディレクトリ・サーバー

Directory Serversページは、**ActiveServer**のさまざまな国際ブランドディレクトリ・サーバーを管理するのに使用されます。
それぞれの国際ブランド用のタブがあります。証明書セクションから、国際ブランド固有の接続詳細を入力し、タイムアウト設定を調整し、SSL接続を管理できます。

詳細については、*ガイド* > *管理UI*以下を参照してください：

- DS設定の管理
- DS証明書の管理

取引

Transactionsページは、**ActiveServer**によって処理されるすべての取引のレコードへのアクセスに使用されます。取引はさまざまなフィールドでフィルタリングでき、取引詳細と3DSメッセージの両方を参照するためにアクセスできます。

このメニュー項目とページは加盟店の管理用に設計されたロール(**Business Admin**、**Merchant Admin**、**Merchant**ロールなど)を持つユーザーにのみ表示されます。

詳細については、*ガイド* > *管理UI* > 取引を見るを参照してください。

デプロイ

Deploymentページは、現在オンラインの**Node**（ノード）を管理できる場所です。また、インスタンスの**アクティベーションのステータス**を確認する場所でもあり、インスタンスが最初にアクティブ化される場所でもあります。

このメニュー項目とページはシステムアーキテクチャの管理用に設計されたロール(**System Admin**ロールなど)を持つユーザーにのみ表示されます。

詳細については、以下を参照してください：

- *ガイド* > *管理UI* > ノードの管理
- Guides > インスタンスのアクティブ化

ユーザー管理

User Managementページは、管理者インターフェイスへのユーザーアクセスを付与および管理できる場所です。ユーザーを *作成* 、*検索* 、*表示* 、*編集* 、*削除* できます。特に、ここはユーザーがさまざまなシステム機能にアクセスするための **ロール**が付与できる場所です。

このメニュー項目とページはシステム全体のユーザーの管理用に設計されたロール(**User Admin**ロールなど)を持つユーザーにのみ表示されます。

詳細については、 *ガイド* > *管理UI* 以下を参照してください：

- ユーザーの管理
- ロールと権限

監査ログ

Audit Logsページは、システムイベントおよび変更が参照用に記録される場所です。

このメニュー項目とページはシステムアーキテクチャの管理用に設計されたロール(**System Admin**ロールなど)を持つユーザーにのみ表示されます。

詳細については、*ガイド* > *管理UI* > ログを参照してください。

設定

Settingsページは、ユーザーがインスタンスの**システム**、**セキュリティ**、**3Dセキュア2**関連の設定を管理できる場所です。

このメニュー項目とページはシステムアーキテクチャの管理用に設計されたロール(**System Admin**ロールなど)を持つユーザーにのみ表示されます。

詳細については、*ガイド* > *管理UI* > システム設定の管理を参照してください。

システム情報

Aboutページは、インスタンスの技術仕様が表示される場所です。ここは、GPaymentsサポートチームに技術サポートを問い合わせる際にユーザーに役立つ情報です。

このメニュー項目とページはシステムアーキテクチャの管理用に設計されたロール(**System Admin**ロールなど)を持つユーザーにのみ表示されます。

詳細については、*ガイド* > *管理UI* > ActiveServerの情報を見るを参照してください。

通知

Notificationsセクションは、重要なシステム通知をユーザーに連絡する場所です。通知は管理インターフェイスの右上の📡アイコンの下に表示されます。

詳細については、*ガイド* > *管理UI* > 通知を参照してください。

ユーザープロフィール

User profileページは、現在のユーザーがアカウントに関する詳細を編集したり、パスワードを変更したりできる場所です。管理インターフェイスの左下の **Profile**アイコンを選択することでアクセスできます。

詳細については、*ガイド* > *管理UI* > ユーザープロフィールを参照してください。

ログファイル

ActiveServerは、毎日ログファイルを作成し、`as_home/logs`ディレクトリに保存します。ログファイルの名前は ***"as.yyyy-mm-dd.log"***形式です(例： 2019年11月23日に作成されたログファイルは ***as.2019-11-23.log***と命名されます)。

ログファイルには、**ActiveServer**コンソールウィンドウに表示されたものと同じメッセージ、警告、エラーが含まれます。

ActiveServerをデバッグ・モードで実行中の場合、ログファイルには取引に関する詳細な情報が含まれるため、サイズが非常に大きくなる可能性があります。必ずログ記録用に十分なディスク容量があることを確認してください。3ヶ月ごとに古いログファイルを削除（またはアーカイブ）することをお勧めします。

ログファイルの詳細度は、システムで設定できます。詳細については、*ガイド* > *管理UI* > システム設定の管理を参照してください。

インスタンスのアクティブ化

ActiveServerインスタンスをアクティブ化するには：

1.GPaymentsからライセンスを購入する

インスタンスをアクティブ化するためのMyAccount機能にアクセスするには、GPaymentsからライセンスを購入する必要があります。詳細については、sales@gpayments.co.jpにお問い合わせください。

2.インスタンスのセットアップ

クイックスタートに従って、ActiveServerインスタンスがセットアップされていることと、管理インターフェイスにアクセスできることを確認してください。

3.外部URLおよび認証API URLの設定

管理インターフェイスで、**Settings > 3D Secure 2**に移動し、**External URL**および**Auth API URL**の値を入力します。***Save***ボタンを選択します。

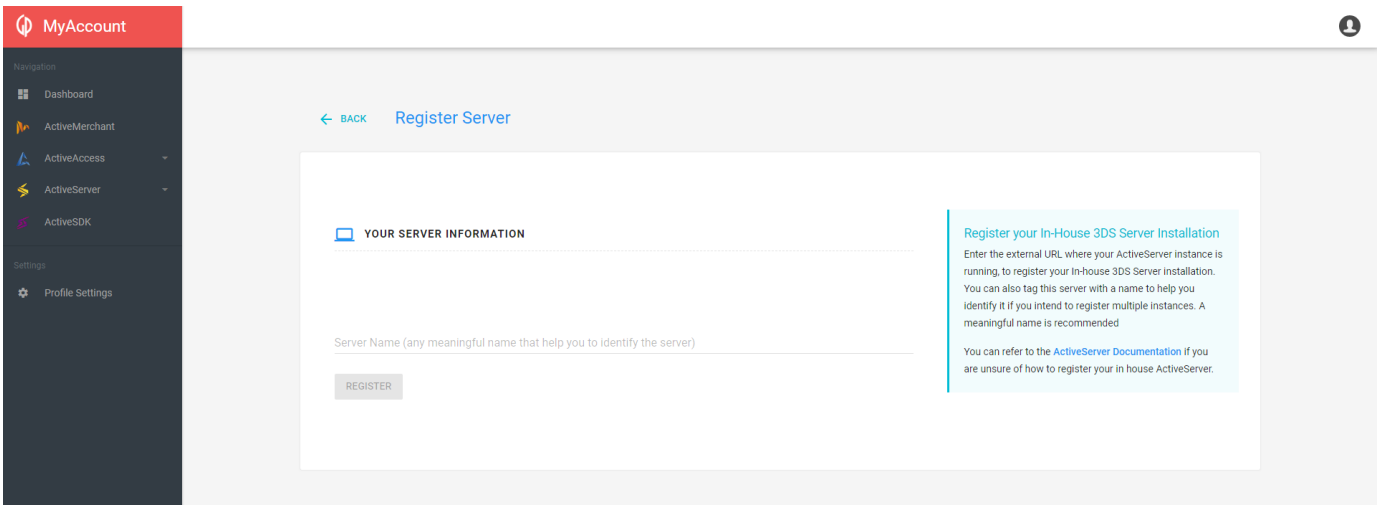
- **External URL** - ActiveServerインスタンスが実行中で、`as.server.https.port`でリスンするように設定した、パブリックにアクセスできるURL。
- **Auth API URL** - 認証APIにアクセスするためのURL。このURLドメイン名は、認証APIコールを行って加盟店を認証するのに使用されるクライアント証明書を発行するために使用されます。このURLが指定されない場合、デフォルトでは、アクティブ化中に設定した**External URL**を使用してクライアント証明書を発行します。

4.サーバーを登録し、アクティブ化の方法を選択

1.MyAccountにログインします。GPaymentsからライセンスを購入している場合は、すでにActiveServerセクションにアクセスできるはずです。

2.サイドメニューの**ActiveServer > My Instances**を選択します。

1. ***ADD NEW SERVER***を選択します。以下のような画面が表示されます。この画面には、**Server Name**の入力フィールドが表示されます。



1. ***REGISTER***を選択します。入力したサーバー情報と **Activation State**が表示されます。間違えてしまい、このインスタンスを削除する場合は、***REMOVE***を選択します。
2. ***ACTIVATE 3DS SERVER***を選択します。以下のアクティブ化方法からいずれかを選択するように求められます。

オプション1：セッションを使用したアクティブ化

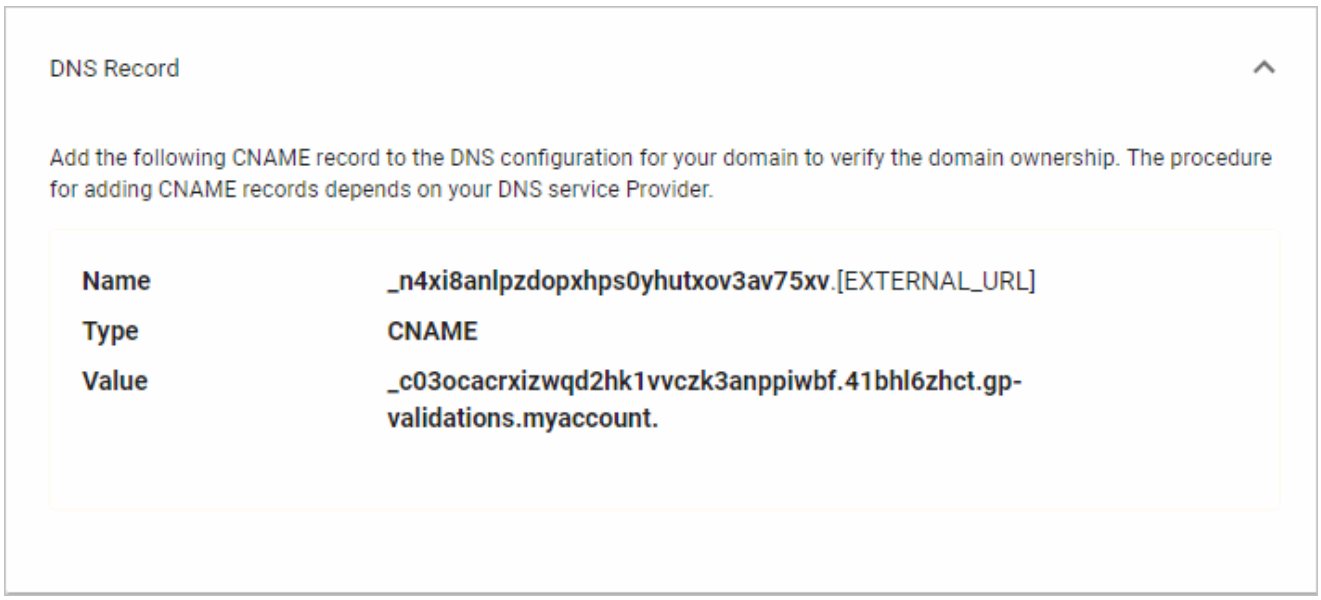
この方法を選択する場合、前のステップで指定した **External URL** がパブリックにアクセスできることを確認してください。

ライセンス・サーバーは、この**External URL**にリクエストを送信し、指定した**External URL**でインスタンスが実行中であることを確認し、インスタンスをアクティブ化します。

オプション2：DNSを使用したアクティブ化

このアクティブ化プロセスでは、GPaymentsのライセンス・サーバーによって生成された `CNAME` レコードを検証することで、ActiveServerインスタンスをアクティブ化します。

DNSレコードは以下のように表示されます：



DNSレコードを作成するには：

1.ドメインのDNSレコードに移動します。

2.DNS設定にレコードを追加し、レコードのタイプとしてCNAMEを選択します。

3.**Name**の値（上記のスクリーンショットの `_n4xi8anlpzdopxhps0yhutxov3av75xv` ）をコピーして、DNSレコードの**Label/Host/Name**に貼り付けます。これはドメイン・ホストによって異なります。

4.**Value**の値（上記のスクリーンショットの `_c03ocacrxizwqd2hk1vvczk3anppiwbf.41bhl6zhct.gp-validations.myaccount.` ）をコピーして、**Destination/Target/Value**に貼り付けます。これはドメイン・ホストによって異なります。

5.レコードを保存します。**CNAME**レコードの変更が有効になるには最大72時間かかる場合がありますが、通常はより短い時間で反映されます。

> **Note**
>
> ドメイン・ホストは通常、ドメイン名の購入元です（GoDaddy®、Enom®、Name.comなど）。

6.すべてのデータ要素を送信するか、送信したデータ要素をカスタマイズするかを選択することで、ライセンス・サーバーに送信されるデータ要素を選択します。

Transaction data (core)： 請求のために必要な情報であり、送信するために必須（または条件付き必須）です。

ID	名前	必須	グループ	コメント
ADE001	ディレクトリ・サーバー・タイプ	Y	コア	認証リクエストがProductionまたはGPayments TestLabsのディレクトリ・サーバーに送信されたかどうかを追跡するのに使用されます。
ADE002	3DSサーバー取引ID	Y	コア	3DSサーバーが取引に割り当てたID。請求の紛争が発生した場合に取引を相互参照するのに使用されます。
ADE003	SDK取引ID	C	コア	条件付き必須：SDK取引に対してのみ割り当てられ、値が存在する場合に指定する必要があります。請求の紛争が発生した場合に取引を相互参照するのに使用されます。
ADE004	ACS取引ID	Y	コア	ACSが取引に割り当てたID。請求の紛争が発生した場合に相互参照するのに使用されます。
ADE005	取引ステータス	Y	コア	取引ステータス（“Y”、“A”、“N”など）。これは、請求のための最終取引ステータスを決定するのに使用されます（すなわち、取引中にエラーが発生）。
ADE006	取引ステータスの理由	C	コア	条件付き必須：取引が失敗した理由であり、請求のために失敗した正確な理由を特定するのに役立ちます。値が存在（すなわち取引が失敗）する場合に指定する必要があります。
ADE007	取引開始時間	Y	コア	取引開始時間。請求サイクルを決定するときに必要です。
ADE008	取引終了時間	C	コア	条件付き必須：取引終了時間。取引が失敗したり早期に終了したりした場合はnullとなり、利用可能な場合は必須です。

Transaction data (extended)： 請求目的で条件付き必須の場合を除き、任意の情報です。この情報をオプトインすると、GPaymentsが匿名の業界の見識を、参加するクライアントと共有することを許可したことになります。

ID	名前	必須	グループ	コメント
ADE009	ペイメント・ネットワーク	N	拡張	取引に使用されるペイメント・ネットワーク（American Express、China UnionPay、Discover、JCB、Mastercard、Visaなど）。請求の仕組み上この情報が必要な場合を除き、クライアントによる提供は任意です。
ADE010	デバイス・チャネル	N	拡張	取引に使用されるデバイス（BRW、APP、3RIなど）。請求の仕組み上この情報が必要な場合を除き、クライアントによる提供は任意です。
ADE011	認証タイプ	N	拡張	取引に使用される認証タイプ（NPA（非決済）、PA（決済）など。請求の仕組み上この情報が必要な場合を除き、クライアントによる提供は任意です。
ADE012	加盟店ID	C	拡張	内部加盟店ID（アクワイアラーが割り当てたIDではありません）。請求の仕組み上この情報が必要な場合は、クライアントによる提供は条件付き必須です。これは、ライセンス・サーバーが（個別の加盟店IDの計算によって）決済代行会社の規模を判断するのに使用されます。
ADE013	加盟店アクワイアラーIDインデックス	C	拡張	加盟店のアクワイアラー加盟店IDのインデックス番号。請求の仕組み上この情報が必要な場合は、クライアントによる提供は条件付き必須です。これは、ライセンス・サーバーが（個別の加盟店IDの計算によって）決済代行会社の規模を判断するのに使用されます。

Tech support data (core)： GPaymentsがトラブルシューティングおよびプランニング目的で使用する情報です。インスタンス・サーバで条件付きで利用できない場合を除き、送信が必要です。

ID	名前	必須	グループ	コメント
AD001	ActiveServerバージョン	Y	コア	ActiveServerのバージョン（v1.0など）
AD002	OS名	C	コア	OSの名前（Ubuntuなど）
AD003	OSバージョン	C	コア	OSのバージョン（16.04.5 LTSなど）

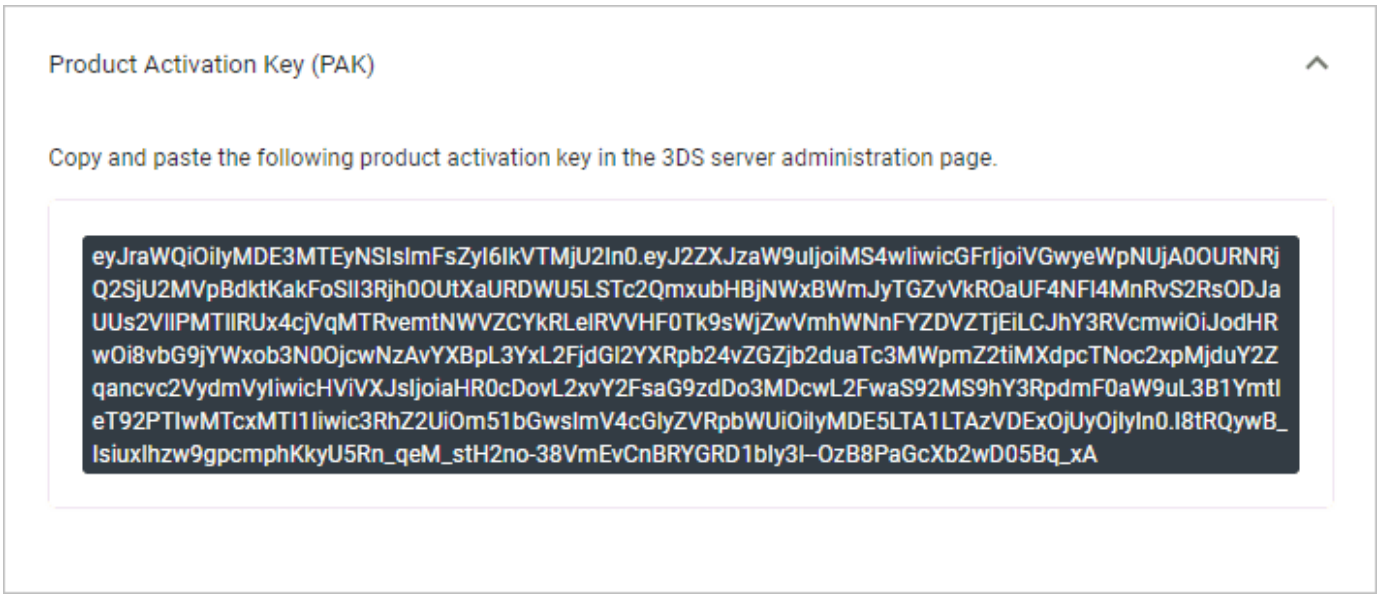
ID	名前	必須	グループ	コメント
AD004	データベース名	C	コア	データベースの名前（MySQLなど）
AD005	データベース・バージョン	C	コア	データベースのバージョン（5.7など）
AD006	Javaエディションおよびバージョン	C	コア	使用されているJavaのバージョンのエディション（OpenJDK 1.8.120など）
AD007	ノード数	C	コア	インスタンスのノードの数（2など）

7.指定した情報を確認し、インスタンスをアクティブ化します。変更が必要な場合は ***BACK***を、そうでない場合は ***FINISH***を選択します。

5.アクティブ化

以下のような製品アクティブ化キー（PAK）が表示されるはずです。

1.この後使用するため、この値をクリップボードにコピーします。



2.ActiveServerダッシュボードに戻り、**Deployment > Activation Status**に移動して、MyAccountから詳細情報を入力します。

- **MyAccount Login Name :** インスタンスをアクティブ化したアカウントに登録されている電子メールアドレス。
- **PAK :** クリップボードにコピーした製品アクティブ化キー。

4.***ACTIVATE*** ボタンを選択します。成功すると、**Activation Status**が*Waiting to restart*に変更されます。

5.変更を有効化し、アクティブ化プロセスを完了するには、インスタンスを **Restart**（再起動）します。

インスタンスをアップグレード

ActiveServerのアップグレードは、`as.jar`を新しいバージョンに置き換えるだけのシンプルなプロセスです。

既存の**ActiveServer**インスタンスを最新バージョンにアップグレードするには：

1.**ActiveServer**インスタンスノードを停止します。

2.**ActiveServer**ディレクトリを開き、ロールバックが必要な場合は、古い`as.jar`ファイルをバックアップ（単純にコピーするか、アーカイブで保存）します。

3.**ActiveServer**データベースをバックアップします（バックアップ・プロセスとデータベース固有の要件については、データベースのドキュメントを参照してください）。

4.新しい**ActiveServer**パッケージを一時ディレクトリに展開し、`as.jar`ファイルを**ActiveServer**ディレクトリにコピーします。

5.**ActiveServer**インスタンスノードを起動します。**ActiveServer**は、起動中、必要に応じてデータベースを自動的にアップグレードします。アップグレードプロセスが完了します。

前のバージョンの**ActiveServer**に*ロールバック*する必要がある場合：

1.**ActiveServer**インスタンスノードを停止します。

2.**ActiveServer**ディレクトリを開き、必要に応じて、古い`as.jar`ファイルをバックアップ（単純にコピーするか、アーカイブで保存）します。

3.**ActiveServer**ディレクトリに以前の`as.jar`をリストアします。

4.以前バックアップした**ActiveServer**データベースをリストアします。

5.**ActiveServer**を起動します。

ロールと権限

ロールと権限は、業務ロールに関連するさまざまなシステム機能への適切なアクセス権をユーザーに付与するために使用されます。ユーザーは複数のロールを持つことができます。ActiveServerには、以下の事前定義済みのユーザーロールがあります。

- **System admin** - 展開とライセンス認証、ディレクトリ・サーバー接続管理、システム設定管理、システム通知の監視を含む、インスタンスの技術的な維持管理を管理するためのロールです。
    - **ページ・アクセス：** *Directory servers*、*Deployment*、*Audit logs*、*Settings*、*About*、*Profile*、*System notifications*
- **User admin** - ロールの割り当てを含む、インスタンスのユーザーを管理するためのロールです。 このロールは、システム内のすべての加盟店を表示でき、単一スコープ・ユーザーへの加盟店の割り当てを可能にします。常にこのロールを持ったユーザーが1人は存在する必要があります。
    - **ページ・アクセス：** *Merchants*、*User Management*
- **Business admin** - ダッシュボード統計の表示、加盟店機能の管理、取引履歴の表示を含む、インスタンス上の**すべての**加盟店の事業プロセスを管理するためのロールです。
    - **ページ・アクセス：** *Dashboard*、*Merchants*、*Transactions*、*Profile*
- **Merchant admin** - ダッシュボード統計の表示、加盟店詳細の管理、取引履歴の表示を含む、インスタンス上の**単一の**加盟店の事業プロセスを管理するためのロールです。
    - **ページ・アクセス：** *Dashboard*、*Merchants*、*Transactions*、*Profile*
- **Merchant** - ダッシュボード統計の表示、加盟店詳細の表示、取引履歴の表示を含む、インスタンス上の**単一の**加盟店への読み取り専用アクセスが必要なユーザー用のロールです。
    - **ページ・アクセス：** *Dashboard*、*Merchants*、*Transactions*、*Profile*

権限スコープ

各ユーザーロールには、**User admin**ユーザーがシステム内のエンティティに対する適切なアクセスレベルを定義できるようにするために添付されたスコープのレベルがあります。

加盟店スコープ

Merchantsについては、スコープはユーザーが**すべての**加盟店とその情報（例： 統計情報、詳細、取引）にアクセスできるかどうか、または**単一の**加盟店の情報にアクセスできるかどうかを示します。

- ***All*スコープ** - ***Business admin***ロールには**すべての**加盟店での権限があります。これによって、ダッシュボード統計の表示時に**すべての加盟店**を選択したり、すべての加盟店を検索/編集/作成/削除したり、システム内のすべての加盟店の取引を表示したりできます。***User admin***には、単一スコープ・ユーザーに加盟店を割り当てるため、加盟店の詳細を表示するアクセス権があります。
- ***Single* スコープ** - ***Merchant admin***および***Merchant***ロールには、**単一の**加盟店のみでの権限があります。加盟店がプロファイルに割り当てられると、その加盟店のダッシュボード統計、加盟店詳細、および取引に対してのみ、アクセスできます。
- ***No* スコープ** - ***System admin***ロールには加盟店の管理に関する権限がないため、加盟店機能のページにアクセスできません。

この役割の分割によって、決済代行会社などのクライアントは単一システムで複数の加盟店を管理し、必要に応じて個別の加盟店を詳細に制御できるようになります。

> **Important**
>
> ユーザーに**All**と**Single**の両方のスコープを持つロールが割り当てられている場合、**All**スコープが優先されます。

加盟店の割り当て

ユーザーの加盟店に関するスコープ・レベル・アクセスが**Single**の場合、***User admin***はそれらをすでに作成済みの加盟店に割り当てて管理できます。

ユーザーがすでに加盟店を割り当てている場合、プロファイルでこれを上書きできますが、一度に複数の加盟店を持つことはできません。

権限リストの表

以下の表は、ユーザーロールに付与されている特定の権限の詳細を示しています。**スコープ**列は、必要に応じてスコープが付随している権限を示します。

User Note

このドキュメントを通じて、特定のユーザーロールで利用可能な機能を示す**User Note**ボックスが表示されます。

ページ	サブページ	権限	スコープ	System Admin	User Admin	Business Admin	Merchant Admin	Merchant
Dashboard		すべての加盟店統計の表示	すべての加盟店			✔		
		加盟店統計の表示	単一の加盟店				✔	✔
Merchants	Search	すべての加盟店詳細の表示	すべての加盟店		✔	✔		
		加盟店詳細の表示	単一の加盟店				✔	✔

ページ	サブページ	権限	スコープ	System Admin	User Admin	Business Admin	Merchant Admin	Merchant
		複数の加盟店の作成	すべての加盟店			✔		
		複数の加盟店の削除	すべての加盟店			✔		
	Merchant Settings	すべての加盟店詳細の表示	すべての加盟店		✔	✔		
		加盟店詳細の表示	単一の加盟店				✔	✔
		すべての加盟店詳細の編集	すべての加盟店			✔		
		加盟店詳細の編集	単一の加盟店				✔	
		すべての加盟店メモの表示	すべての加盟店			✔		
		すべての加盟店メモの編集	すべての加盟店			✔		

ページ	サブページ	権限	スコープ	System Admin	User Admin	Business Admin	Merchant Admin	Merchant
		すべての加盟店の有効ステータスの編集	すべての加盟店			✔		
		すべての加盟店証明書のダウンロード	すべての加盟店			✔		
		加盟店証明書のダウンロード	単一の加盟店				✔	✔
		すべての加盟店証明書の失効	すべての加盟店			✔		
		加盟店証明書の失効	単一の加盟店				✔	
		すべての加盟店暗号化キーのローテーション	すべての加盟店			✔		
		加盟店暗号化キーのローテーション	単一の加盟店				✔	
	Acquirer	アクワイアラーの表示				✔		
		アクワイアラーの作成				✔		

ページ	サブページ	権限	スコープ	System Admin	User Admin	Business Admin	Merchant Admin	Merchant
		アクワイアラーの編集				✔		
		アクワイアラーの削除				✔		
Directory Servers		ディレクトリ・サーバー設定の表示		✔				
		ディレクトリ・サーバー設定の編集		✔				
		ディレクトリ・サーバー証明書の表示		✔				
		ディレクトリ・サーバー証明書の編集		✔				
Transactions		すべての加盟店取引の表示	すべての加盟店			✔		
		加盟店取引の表示	単一の加盟店				✔	✔
Deployment	Nodes	展開情報の表示		✔				

ページ	サブページ	権限	スコープ	System Admin	User Admin	Business Admin	Merchant Admin	Merchant
		展開情報の編集		✔				
	Activation Status	アクティブ化詳細の表示		✔				
		製品アクティブ化情報の編集		✔				
User Management	Search	すべてのユーザー詳細の表示	すべてのユーザー		✔			
		ユーザーの追加			✔			
		ユーザーの削除			✔			
	Details	すべてのユーザー詳細の編集	すべてのユーザー		✔			
		すべてのユーザーロールの編集	すべてのユーザー		✔			
		すべてのユーザー・ステータスの編集	すべてのユーザー		✔			

ページ	サブページ	権限	スコープ	System Admin	User Admin	Business Admin	Merchant Admin	Merchant
Audit Logs		すべての監査ログの表示		✔				
Settings	System	システム設定の表示		✔				
		システム設定の編集		✔				
	Security	セキュリティ設定の表示		✔				
		セキュリティ設定の編集		✔				
	3D Secure 2	3Dセキュア2設定の表示		✔				
		3Dセキュア2設定の編集		✔				
About		詳細の表示		✔	✔	✔		
User profile	プロファイルの編集	ユーザー詳細の表示	単一のユーザー	✔	✔	✔	✔	✔
		ユーザー詳細の編集	単一のユーザー	✔	✔	✔	✔	✔
Notifications		システム通知の表示		✔				

ページ	サブページ	権限	スコープ	System Admin	User Admin	Business Admin	Merchant Admin	Merchant
		ユーザー通知の表示		✔	✔	✔	✔	✔
Reset Password		パスワードのリセット		✔	✔	✔	✔	✔

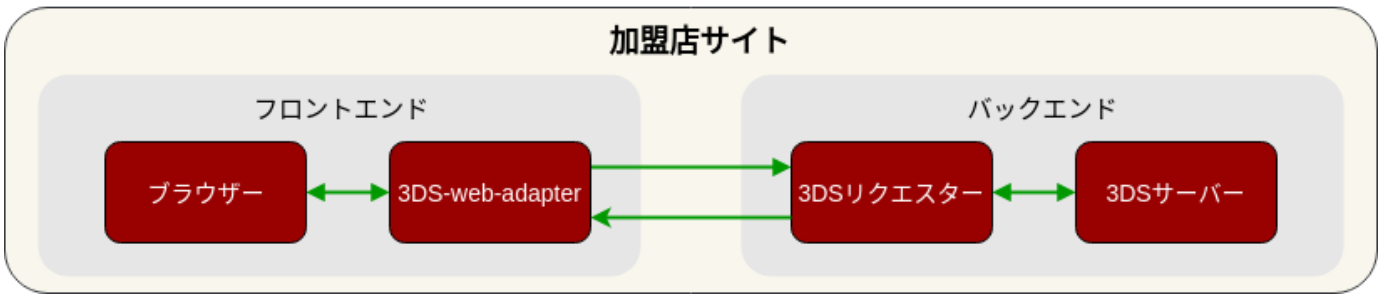
統合まとめ

加盟店または決済代行会社に3DS2認証を組み込むためには、eコマースサイト、eコマースサイトのバックエンドシステムに**ActiveServer's**認証APIを実装する必要があります。

API呼び出しは、アプリが動作中に特定のタスクを実行するために起動可能な処理です。 すべてのAPIリクエストは、動作の軽いデータ転送形式であるJSON形式で作成されます。 API文書の詳細は、APIドキュメントまとめを参照してください。

この章では、**ActiveServer**に接続できるように加盟店の**ウェブサーバー**を組み込んでテスト用の取引を実行する方法について、概要を説明します。加盟店の**App** の組み込みについては、**ActiveSDKドキュメント**を参照してください。

3DS2を利用するためには、加盟店サイトに2つの機能、すなわちフロントエンドの**3DS web adapter**とバックエンドの**3DSリクエスター**を実装する必要があります。次の図は、ブラウザー、3DS webアダプター、3DSリクエスター、3DSサーバーの間の関係を示します。



- **3DS web adapter** - 3DS webアダプターは加盟店サイトの3DS2コンポーネントであり、利用者のデバイスから3DSリクエスターに3Dセキュアデータを渡すために使用されます。3DS webアダプターとなり得るものの例としては、ブラウザーでの操作に対する応答を実行し3DS認証リクエストを3DSリクエスターに送信する `javascript (.js)` があります。
- **3DSリクエスター** - 3DSリクエスターはコントローラーであり、3DS webアダプターと3DSサーバーの間のブリッジとして使用されます。3DSリクエスターは3DS webアダプターからの3DS認証リクエストを受信し、それらのリクエストの形式を整えて3DSサーバーに送信します。また、3DSリクエスターは3DSサーバーからの認証結果を受信して3DS webアダプターに転送します。

取引の実行

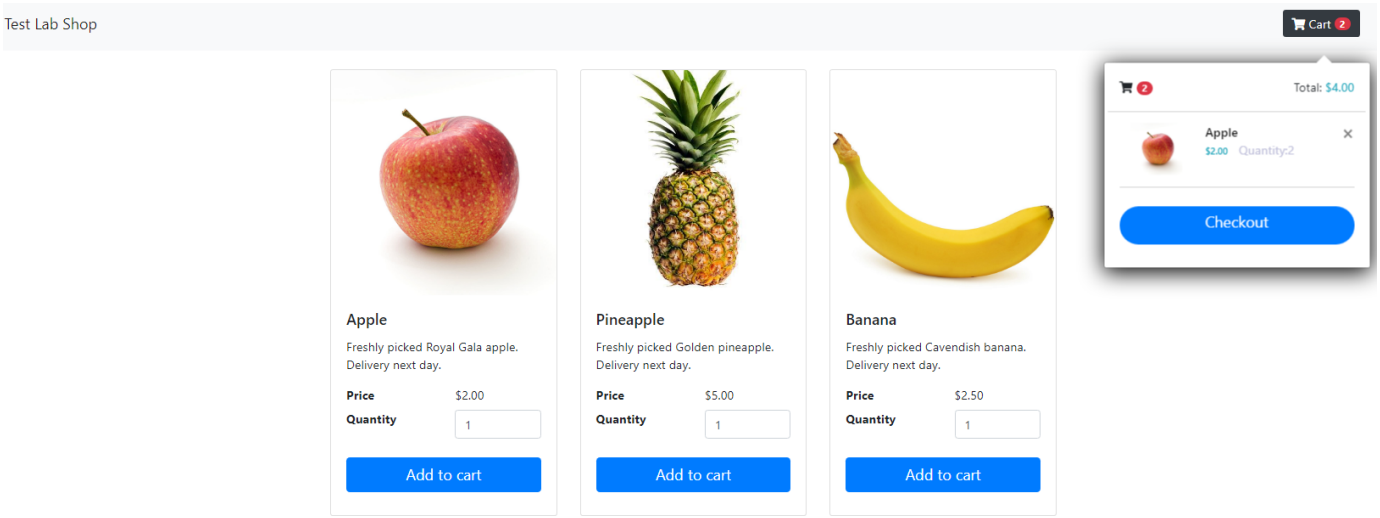
3DS2を使用した取引をシミュレートするため、このデモ用加盟店サイトを使用して、認証APIがどのように動作するかを確認できます。

> **Tip**
>
> この「組み込みガイド」のサンプルでは、このデモ用加盟店サイトを使用しますので、先に進む前に使用してみてください。

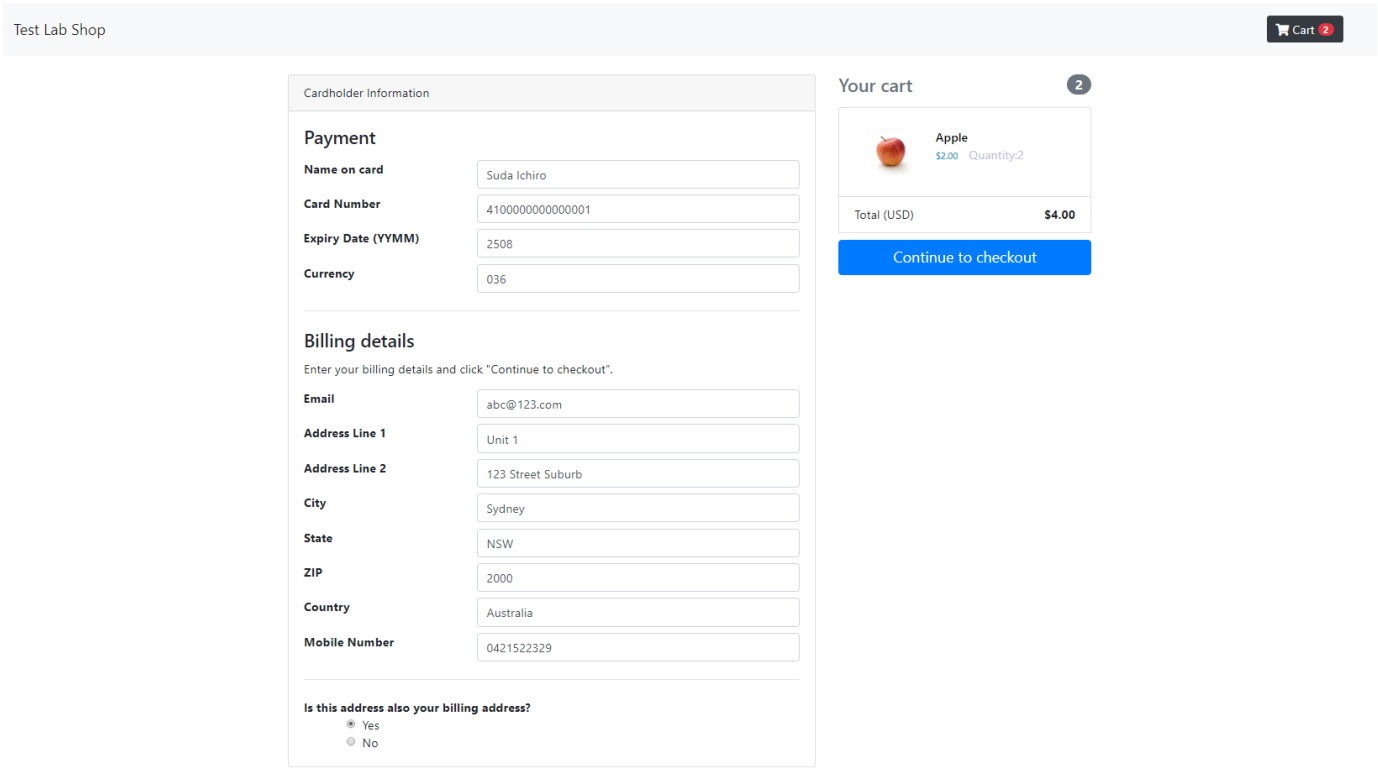
フリクションレス・フロー

フリクションレス取引を開始するには、デモ用加盟店サイトを開き、果物をカートに追加します。

画面右上の**Cart**アイコンを選択すると、カートの内容が表示されます。

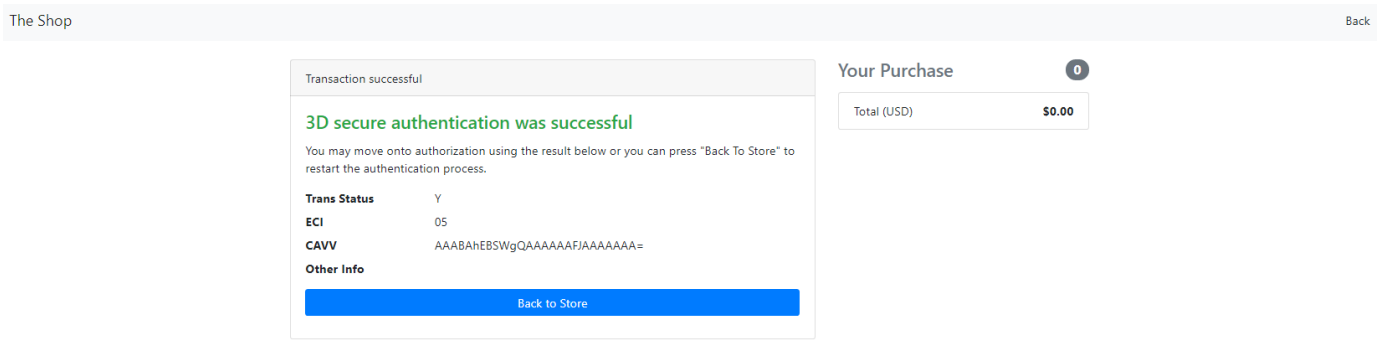


Checkoutボタンを選択して決済ページに移動します。



カード番号など、決済と請求書送付先に関するデフォルトの情報があらかじめセットされており、取引の実行にこれらの情報を使用できます。***Continue to checkout***ボタンを選択すると3DS2認証処理が開始されます。

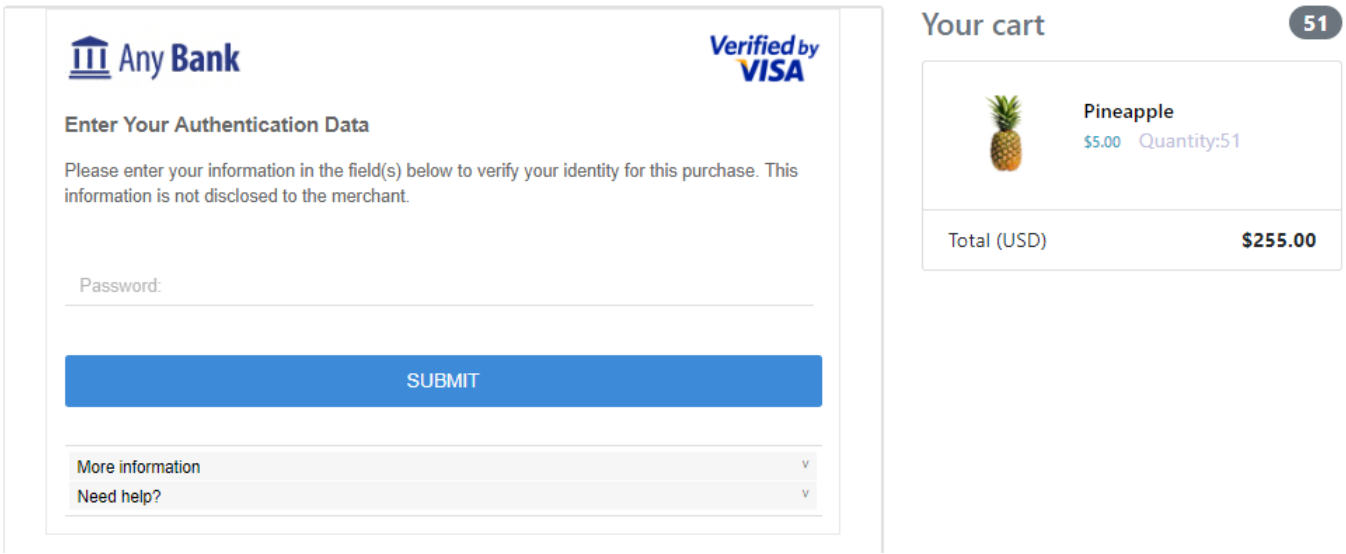
3DS web adapterはカード会員情報を収集して**3DSリクエスター**に送信します。**3DSリクエスター**はこの情報を使用してAPIリクエストを作成して**3DSサーバー**に送信し、**3DSサーバー**は3DS2メッセージングを開始します。**3DSリクエスター**は認証結果を待って**3DS web adapter**に結果を返し、結果が次のようなwebページに表示されます。



これで、**フリクションレス・フロー**を使用した取引は完了です。シミュレートした取引は低リスクと見なされたためチャレンジは要求されませんでした。

チャレンジ・フロー

チャレンジ・フローをテストするには、***Back to Store***ボタンを選択してチャレンジシナリオをテストしたい場合は、チェックアウトページでカード番号を**4100000000000050**に設定しましょう。このシミュレーションではACSは、取引がハイリスクであると判断しカード会員の追加のアクションが必要と判断しますのでチャレンジ・フローに移行します。次のようなチャレンジ画面が表示されます。このデモで使用するパスワードは**123456**です。



パスワードを入力すると正しく取引が処理されるはずです。実際の状況における、このチャレンジの方法としては、イシュアーのACSやカード会員に対応する登録済み認証方法に応じて**OTP**や**生体認証**などさまざまな方法が考えられます。

次のチャプター

下のフッターの**次**を選択し、**認証処理**に関する詳細を学習してください。

認証処理

3DSリクエスターは、認証中に次の3つの処理を実行します。

1.**認証の初期化**ー3DSリクエスターは**ActiveServer**にリクエストを送信して認証を初期化し、認証を行えるよう**ActiveServer**を準備します。

2.**認証の実行**ー**ActiveServer**は認証を実行します。3DS2には2つの主要な認証フローー**フリクションレス・フロー**と**チャレンジ・フロー**があります。これらについてはProcess Flowsの項で説明します。

3.**認証結果の取得**ー3DSリクエスターに認証結果が返されます。

処理 1: 認証の初期化

このステップで、フロントエンドの**3DS web adapter**はカード会員が入力した情報を取り込み、バックエンドの**3DSリクエスター**に渡します。次に**3DSリクエスター**は、3DS2から要求されたすべての情報を**ActiveServer**に渡し、認証処理が開始されます。

実装方法まとめのサンプルで、利用者が***Continue to checkout***を選択すると、**3DS web adapter**が3DSリクエスターに `認証の初期化` メッセージを送信します。 `認証の初期化` メッセージには、一意な**transaction ID (`transId`)**とカード会員の情報が含まれています。3DSリクエスターは `認証の初期化` メッセージを受信し、 `/api/v1/auth/brw/init/` を呼び出してActiveServer認証APIに適合する形式にし3DSサーバーに送信します。

> **Note**
>
> - `/auth` は、リクエストが認証リクエストであることを示します。
> - `/brw` は、リクエストの送信元がブラウザーであることを示します。3DSサーバーは、 `/app` を使用してモバイルアプリからのリクエストも受信します。
> - `/init` は、リクエストが認証初期化リクエストであることを示します。

ActiveServerは、 `/api/v1/auth/brw/init/` メッセージを受信するとブラウザー情報を収集し、認証処理を行える状態になります。

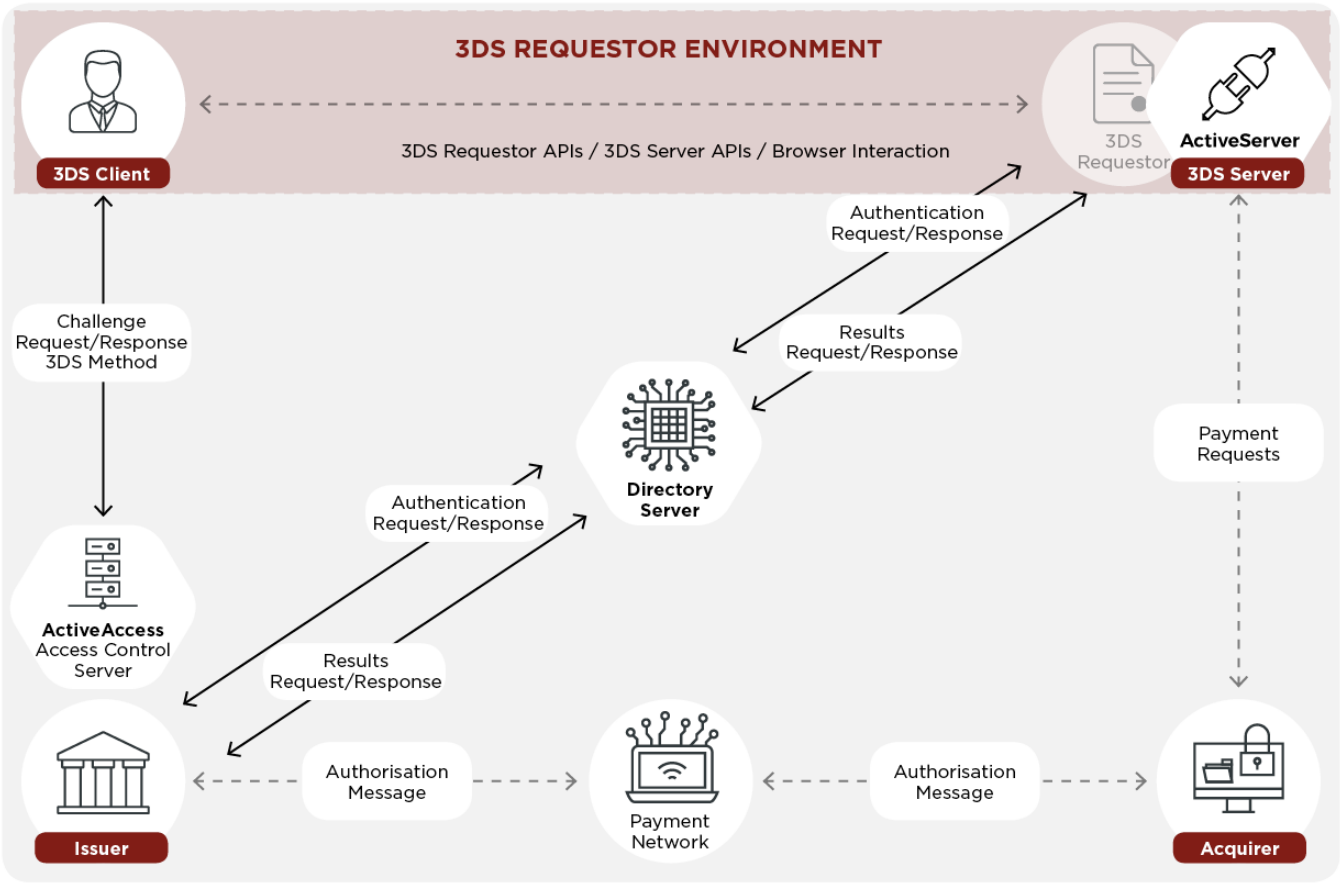
> **Note**
>
> 3DSサーバーとACSは自動的にブラウザー情報を収集します。この処理の概要は**3DSリクエスター**には含まれ**ません**。

処理 2: 認証の実行

ブラウザー情報の収集が完了すると、加盟店は `/api/v1/auth/brw` を呼び出して認証を実行できます。この処理が実行されると、**ActiveServer**が3DS2メッセージング処理を開始します。3DS2には2つの主要な認証フロー**フリクションレス・フロー**と**チャレンジ・フロー**があります。

- **フリクションレス・フロー**ーARes/ARes認証メッセージからなる3Dセキュア認証フローを開始します。 与えられた情報から取引が低リスクであるとACSが判断した場合は、直ちに認証が承認されます。
- **チャレンジ・フロー**ー取引が特定の許容限界値より高リスクであるとACSが判断した場合、または法律によってチャレンジが必須である場合は、カード会員がさらに操作を行うことが必要な、**フリクションレス・フロー**が**チャレンジ・フロー**に切り替わります。**チャレンジ・フロー**は**フリクションレス・フロー**でもあった**AReq/ARes**メッセージ、**CReq/CRes**チャレンジメッセージと**RReq/RRes**結果メッセージから構成されます。

チャレンジ・フローを次の図に示します。



> *点線は、クライアント/3DSリクエスターと信用承認機能の間の通信など、3DS2プロトコルの範囲外のメッセージングを示します。*

処理 3: 認証結果の取得

3DS2処理が完了すると、加盟店は `/api/v1/auth/brw/result` を呼び出して認証結果を取得します。認証結果（チャレンジのステータスによりAResまたはRRes）には、ECI、認証値（CAVVなど）、および3DSリクエスターへの最終取引ステータスなどの情報が含まれています。

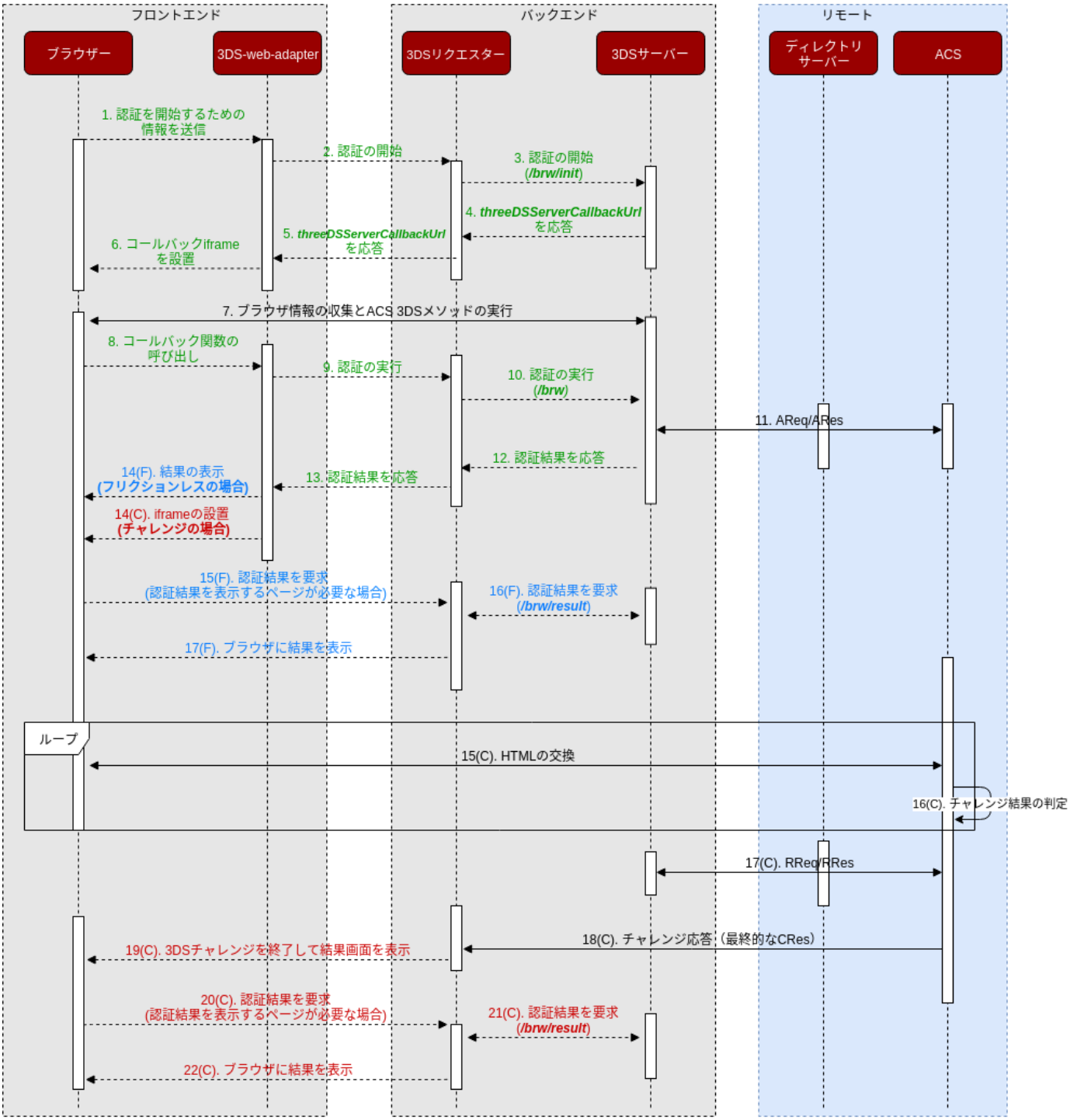
> **次のチャプター**
>
> 下のフッターの**次**を選択し、**認証シーケンス**の詳細をご覧ください。

認証シーケンス

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

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

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



- 点線は3DSリクエスターが実装する処理になります
- 実線は3DSリクエスター以外のコンポーネントが実装する処理になります
- 緑色の処理はチャレンジとフリクションレスどちらのフローでも行われる処理になります
- 青色の処理はフリクションレスフローのみで行われる処理で(F)と付きます
- 赤色の処理はチャレンジフローのみで行わる処理で(C)と付きます

1. 認証の初期化用の情報を送信

- カード番号やカード会員の氏名など、決済ページで得られたカード会員情報が、**3DS web adapter**に送信されます。これは加盟店のフロントエンドシステムをシミュレートした簡潔なJavaScriptのコードです。

2. **認証の初期化**
    - **3DS web adapter**は決済ページから収集した情報を使用して**3DSリクエスター**へのPOSTリクエストを行い、**3DSリクエスター**に認証の初期化を要求します。
3. **認証の初期化** (`/brw/init/{messageCategory}`)
    - **3DSリクエスター**はフロントエンドから情報を取得し、 `/brw/init/{messageCategory}` へのPOST API呼び出しを行って認証を初期化します。
    - ここで送信される重要なフィールドは `eventCallbackUrl` であり、**3DSサーバー**がこのURLへのコールバックを行ってブラウザーの情報収集完了を通知できるようステップ8を開始するために必要です。
4. **`threeDSServerCallbackUrl` を応答**
    - `/brw/init/{messageCategory}` からの正常な応答には `threeDSServerCallbackUrl` と `threeDSServerTransID` が含まれています。
5. **`threeDSServerCallbackUrl` を応答**
    - **3DSリクエスター**は**3DS web adapter**に `threeDSServerCallbackUrl` を返します。
6. **コールバック `iframe` を設置**
    - `src` 属性を `threeDSServerCallbackUrl` に設定した非表示の `iframe` を挿入します。これにより、**3DSサーバー**は**3DSリクエスター**に接続できる状態になります。**3DSサーバー**は、この `iframe` へのコールバックを行います。
7. **ブラウザ情報の収集**
    - **3DSサーバー**は `iframe` を使用してブラウザー情報を収集し、**ACS**が3DSメソッドデータを収集できるようにします。次に、**ACS**は、用意された `iframe` を使用して3DSメソッドデータを収集します。
8. **コールバック関数の呼び出し**
    - 認証の初期化中、3DSメソッドが終了またはスキップされたときにACSが**3DSリクエスター**に通知できるよう、**3DSリクエスター**は `eventCallBackUrl` を送信します。 ステップ７で設置した `iframe` からPOST要求がこの `eventCallBackUrl` に行われます。
    - **3DSリクエスター**は、この要求を受け `iframe` 内に必要な `callbackFn` 含めたパラメーターと一緒に `notify-3ds-events.html` を表示します。 `notify_3ds_events.html` は表示後 `3ds-web-adater` に定義された `callbackFn` を呼び出します。

9. **認証の実行**
    - `callbackFn` は `_on3DSMethodSkipped()` または `_on3DSMethodFinished()` のいずれかであり、どちらも `doAuth()` を呼び出します。**3DS web adapter**は `doAuth()` を呼び出し、認証を実行するよう3DSリクエスターに要求します。
    - `_on3DSMethodSkipped` はブラウザの情報がなんらかの理由によってACSが取得できなかったことを意味します。なので、もしこのコールバック関数が呼ばれた場合加盟店は認証を続行しない選択をすることもできます。
10. **認証の実行 (`/brw`)**
    - **3DSリクエスター**は `/brw` を呼び出して認証処理を開始します。
11. **AReq/ARes**
    - 認証リクエスト（AReq）は、ディレクトリ・サーバーを介して**3DSサーバー**からACSに送信されます。ACSからは、認証結果を含む認証応答（ARes）が**3DSサーバー**に送信されます。
12. **認証結果を応答**
    - `/brw` は、**3DSリクエスター**に `tranStatus` を返します。
13. **認証結果を応答**
    - 認証結果を3DS webアダプターに返します。

> **Info**
>
> 返された `transStatus` が"Y"の場合は ステップ 14(F) に、"C"の場合は ステップ 14(C) に進んでください。

フリクションレス・フロー specific

14(F). 結果の表示（フリクションレスの場合)

- 認証結果の `transStatus` が"Y"の場合は `authSuccess()` が呼び出され、ページを `/auth/result?transId` にリダイレクトします。

15(F). 認証結果を要求 (認証結果を表示するページが必要な場合)

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

16(F). **認証結果を要求(`/brw/result`)**

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

17(F). **ブラウザに結果を表示**

- 結果画面が開き、認証結果が表示されます。

チャレンジフローの場合

14(C). **`iframe` の設置（チャレンジの場合）。**

- 認証結果の `transStatus` が"C"である場合は `startChallenge()` が呼び出され、チャレンジ用の `iframe` を挿入します。

15(C). **HTMLの交換**

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

16(C). **チャレンジ結果の判定**

- ACSは、実行されたチャレンジが成功したかいなかを判定します。

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

- ACSは、ディレクトリ・サーバーを介して**3DSサーバー**に、認証結果を含む結果リクエスト（RReq）を送信します。**3DSサーバー**は、結果応答（RRes）を使用して受信確認通知を送信します。

18(C). **チャレンジ応答（最終的なCRes）**

- ACSは、最終的なチャレンジ応答（CRes）を**3DSリクエスター**に送信します。

19(C). **3DSチャレンジを終了して結果画面を表示**

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

20(C). **認証結果を要求(認証結果を表示するページが必要な場合)**

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

21(C). **認証結果を要求(`/brw/result`)**

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

22(C). **ブラウザに結果を表示**

- ブラウザーで結果画面が開き、認証結果が表示されます。

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

序章

実装ガイドの一連の文書では、シンプルな仮の加盟店決済サイトを使用して、バックエンド言語に**Java**を使用した3DS2フロー組み込み手順を最初から一通り説明します。

加盟店決済サイトのサンプルは、Springbootプロジェクトを使用して作成されています。参考までに、HTMLページの作成には以下のものが使用されています。

- Bootstrap 4.1.3
- Font Awesome 5.2.0
- JQuery 3.3.1

加盟店決済ページのサンプルには、次の**3つのページ**が含まれています。

- `index.html` - 購入する商品を選択するショッピングページ。
- `checkout.html` - カード会員情報と処理画面が含まれた決済ページ。
- `result.html` - 決済が成功したか否かを表示する決済結果ページ。

前提条件

このガイドを利用するための前提条件を以下に示します。

- Javaに関する必須知識
- webテクノロジーに関する必須知識（HTML、CSS、JavaScript）
- JDK 1.8
- 任意のIDE
- Apache Maven。インストールについてはhttps://maven.apache.org/install.htmlを参照してください。
- Gitクライアント
- アクティブ化され実行中のActiveServerインスタンス

サンプルコードの確認

3DSリクエスターのデモ用コードは次のGithubリポジトリ内にあります。

https://github.com/gpayments/3ds-requestor-springboot.git

サンプルコードを確認するには、ユーザーのローカル環境で次のコマンドを実行してください。

```
git clone https://github.com/gpayments/3ds-requestor-springboot.git
```

リポジトリのクローンを作成すれば、このチュートリアルに必要なすべてのデモ用コードを `3ds-requestor-springboot` ディレクトリ内で見つけることができます。

サンプル加盟店サイトの実行方法

上記のGPaymentsの*サンプルコード*には、次の2つのディレクトリが含まれています。

- **initial** - 3DS2認証フローを**実装していない**加盟店決済ページのサンプル。
- **final** - 3DS認証フローを**実装した**最終的な加盟店決済ページ。このガイドにしたがって実装することで終了的に得られるのが、この最終的なコードです。

3DS 2.0認証機能が有効な加盟店サイトのサンプルを実行するには、以下の手順に従ってください。

1. ActiveServer Administrationからテスト用の加盟店クライアント証明書**API クライアント証明書 (.p12 file)**をダウンロードします（*Administration UI -> Merchants -> Download Client Certificate*と操作）。 最初にActiveServerをアクティブ化しておく必要があることに注意してください。
2. 3DSリクエスターと3DSサーバーが相互に認証できるよう、ダウンロードした証明書（.p12ファイル）をファイルパス `/final/src/main/resources/certs` にコピーします。デモ用コードがp12ファイルをロードできるよう、証明書バンドルの名前を `client_certificate.p12` に変更します（ファイル名がハードコーディングされているため）。 アクティベーションや証明書のダウンロードに問題がある場合は、**GPayments**までご連絡ください。
3. **ターミナル**（Linuxの場合）または**コマンドプロンプト**（Windowsの場合）を開き、`/final` ディレクトリに移動します。

4. ルートディレクトリで次のコマンドラインを実行します。初回の実行時にはMavenが必要な依存関係をダウンロードするため、実行に数分かかる場合があります。

```
cd final
mvn spring-boot:run
```

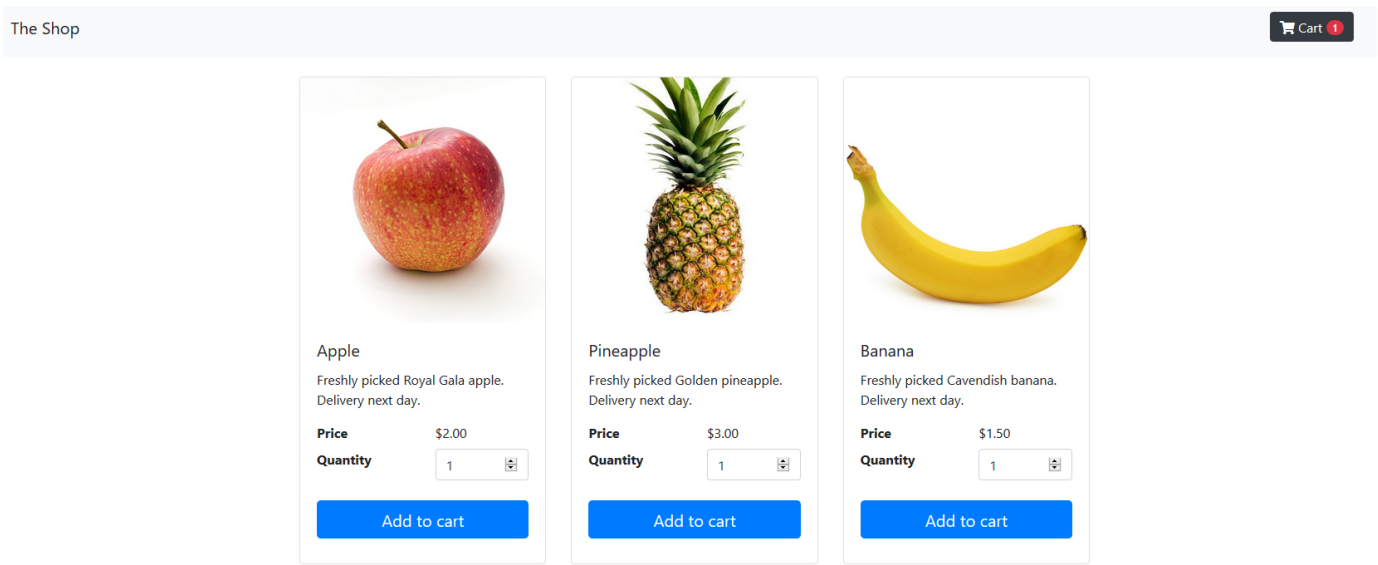
Note

上記のコマンドを実行する前に、Mavenがインストールされ使用可能な状態になっていることを確認してください。Mavenのインストールについては、https://maven.apache.org/install.htmlを参照してください。

Warning

Windowsでは、Javaネットワークアクセスに関するWindowsファイアウォールのセキュリティ警告が表示される場合がありますが、アクセスを許可して操作を進めてください。

http://localhost:8082にアクセスすると、サンプルのカートページを表示、確認できます。 いくつかの商品をカートに追加しデフォルトのカード会員情報を使用して決済する処理を試してみてください。



Note

ポート8082が使用中の場合はエラーになります。その場合は、次の行を[`/resources/application.properties`]ファイルに追加してポートを設定します。

```
server.port = # お好きなポート番号
```

Note

3DS 2.0認証機能のないデモオンラインショッピングサイトである初期状態のままで、プロジェクトを実行することもできます。 このような形で実行する場合は必ず、 `final` インスタンスを停止してください（すでに実行している場合）。そうでないとポート番号が競合してしまいます。

```
cd initial
mvn spring-boot:run
```

Warning

サンプルのリクエスターコードはデモ専用でありHTTP上で動作しています。実際の状況ではHTTPSを使用しましょう。

パッケージの実行状態により、3DS2機能を有効または無効にした状態で決済処理が実行されます。

次のチャプター

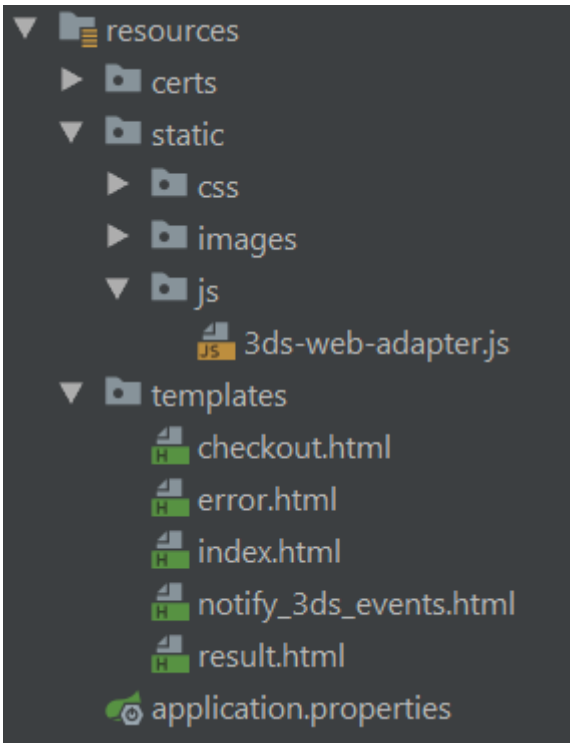
nextボタンを選択し、3DSリクエスター用の**フロントエンド実装**についてご覧ください。

フロントエンド実装

この章では、サンプル・コードを使用して、加盟店アプリのフロントエンドの実装方法について説明します。

> **Note**
>
> **サンプル・コード**に目を通し、実装してから、次に進んでください。

次の図に、フロントエンド用の主なファイルを示します。これらは `/final` プロジェクトの `/resource` ディレクトリにあります。必要に応じ**ディレクトリ構想**を確認してください。 `3ds-web-adapter.js` は**3DS-web-adapter**です。



処理 1: 認証の初期化

認証を初期化するためには、フロントエンドが以下のように動作する必要があります。

- **3DSリクエスター**に `認証の初期化` する為のメッセージを送信する（ステップ.1、およびステップ.2）。

- 応答メッセージを受信し、コールバック用の `iframe` をセットアップする（ステップ.5、およびステップ.6）。

ユーザーが決済ページの**"Continue to checkout"**ボタンをクリックすると、ブラウザーは、認証処理を開始するための情報を**3DS-web-adapter**に送信します(ステップ.1)。この処理は `checkout.html` の `checkout()` 関数で実行されます。

```
//checkout.html
function checkout() {
    ...
    $("#cardholderInfoCard").remove();
    $("#checkoutCard").removeClass("d-none");

    //デモ用に結果画面を表示するため2秒後に移動する
    setTimeout(function () { window.location.href = "/auth/result"}, 2000);
    threeDSAuth(transId, cardHolderInfo, purchaseInfo);
}
```

`3ds-web-adapter.js` で `threeDSAuth()` 関数が実行されます(ステップ.2)。

```
    //3ds-web-adapter.js
    function threeDSAuth(transId, cardHolderInfo, purchaseInfo) {

        console.log('init authentication');
        $.ajax({
            url: '/auth/init',
            type: 'POST',
            contentType: "application/json",
            data: JSON.stringify({
                threeDSRequestorTransID: transId,
                cardExpiryDate: purchaseInfo.expiryDate,
                purchaseAmount: purchaseInfo.purchaseAmount,
                purchaseCurrency: purchaseInfo.purchaseCurrency,
                acctNumber: purchaseInfo.acctNumber,
                cardHolderInfo: cardHolderInfo
                // ここで他の情報を３DSリクエスターに送ることもできる。acctInfo等
            }),
            success: function (data) {
                console.log('init auth returns:', data);
                $('<iframe id="threeds-container" width="0" height="0"
                style="visibility: hidden;"
                src="' + data.threeDSServerCallbackUrl + '"></iframe>')
                    .appendTo('.challengeContainer');
            },
            error: function () {
                alert('error');
            },
            dataType: 'json'

        });

    }
```

`threeDSAuth()` 関数は**3DSリクエスター**のURL `/auth/init` にPOSTリクエストを行います（**行6**）。オブジェクトはJSON形式でPOSTされます。**3DSリクエスター**がこのリクエストを処理しているかの確認については、ここを参照してください。

> **Note**
>
> 各自の実装方法によって `threeDSAuth()` を変更し、 `cardholderInfo` 、 `purchaseInfo` 、 `transId` でなく独自のオブジェクトを転送できます。データ構造については、API documentを参照してください。

`/auth/init` からの応答が正常に得られると(ステップ.5)、 `3ds-web-adapter.js` は非表示の `iframe` を決済ページに埋め込み(ステップ.6)、ACSと3DSサーバーが

`threeDSServerCallbackUrl` を使用してブラウザー情報を収集できるようになります(ステップ.7)（**行20〜23**）。

> **Info**
>
> シーケンス図のステップ.1〜ステップ.7がこの `処理 1: 認証の初期化` で行われます。

処理 2: 認証の実行

認証を実行するためには、フロントエンドが以下のように動作をする必要があります。

- `notify_3ds_events.html` を実行する。
- **3DSリクエスター**に `認証の実行` メッセージを送信する（ステップ.8、およびステップ.9）。
- 認証結果をフリクションレス・フローまたはチャレンジ・フローとして処理する（ステップ.13、ステップ.14(F)、またはステップ.14(C)）。

`notify_3ds_events.html` は、認証処理を開始するために使用されます(ステップ.8)。**3DSリクエスター**は、このファイルに対し、 `transId` 、 `callbackType` 、およびオプションの `param` を返します。バックエンドでの実行については、ここを参照してください。

```
<!--notify_3ds_events.html-->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8"/>
    <title>3DSecure 2.0 Authentication</title>
</head>
<body>

<form>
    <input type="hidden" id="notifyCallback" name="notifyCallback" data-th-
value="${callbackName}"/>
    <input type="hidden" id="transId" name="transId" data-th-value="${transId}"/
>
    <input type="hidden" id="param" name="param" data-th-value="$
{callbackParam}"/>
</form>

<script src="https://code.jquery.com/jquery-3.3.1.min.js"
        integrity="sha256-FgpCb/KJQlLNfOu91ta32o/NMZxltwRo8QtmkMRdAu8="
        crossorigin="anonymous"></script>
<script>

//チェックアウトページのiframe内で実行される。

var callbackFn = parent[$('#notifyCallback').val()];

//callbackFnは3DSリクエスターの3ds-notifyハンドラーメソッドから渡されます
if (typeof callbackFn === 'function') {
    callbackFn($('#transId').val(),$('#param').val());
}

</script>

</body>
</html>
```

`callbackName` により、 `3ds-web-adapter.js` で呼び出されるメソッドが異なることが分かります(ステップ.8)。 `callbackName` の値は `_onThreeDSMethodFinished` または `_onThreeDSMethodSkipped` のいずれかでなければなりません。 各メソッドについて以下に説明します。

- `_onThreeDSMethodFinished` - 3DSメソッドが完了したことを通知し、ブラウザー情報の収集が完了したことを示す `_doAuth()` を呼び出します。

- `_onThreeDSMethodSkipped` - 3DSメソッドをスキップしたことを通知し、ACSによるブラウザー情報の収集をスキップしたことを示す `_doAuth()` を呼び出します。

`3ds-web-adapter` の `_doAuth()` は、 `/auth` へのPOSTリクエストを行い認証を実行します(ステップ.9)。**3DSリクエスター**がこのリクエストを処理しているかの確認については、ここを参照してください。

```
    //3ds-web-adapter.js
    function _doAuth(transId) {
        ...
        $.post("/auth", {id: transId}).done(function (data) {
            console.log('auth returns:', data);

            if (!data) {
                alert('error');
                return;
            }

            switch (data.transStatus) {
                case "Y":
                    authSuccess(data, transId);
                    break;
                case "C":
                    startChallenge(data.challengeUrl);
                    break;
                ...
            }
    }
```

`_doAuth()` は `/auth` へリクエストをPOSTし（**行4**）、結果の `data` を取得します。この後に実行されるフローは、返された `transStatus` により異なります（**行14と17**）。

`transStatus` が `Y` の場合は認証に成功したことを意味します。**3DS-web-adapter**はフリクションレス・フローに切り替わり、次の処理である `authSuccess()` 関数を実行します(ステップ.14(F))。-> 認証結果の取得。

`transStatus` が `C` の場合、 `3ds-web-adapter.js` は `startChallenge()` を呼び出し、チャレンジ画面を表示するため `src` 属性が `challengeUrl` にセットされた `iframe` をに挿入します(ステップ.14(C))。

```
    function startChallenge(url) {
    //remove the spinner
    $(".spinner").remove();

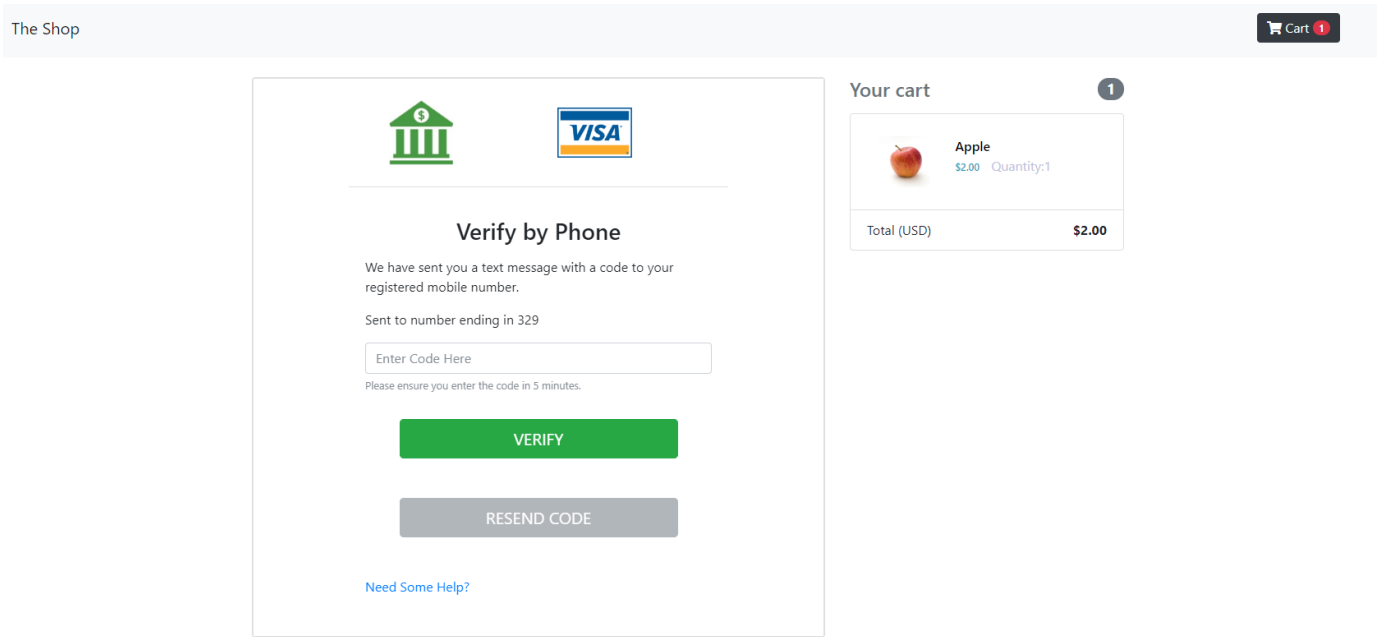
    //create the iframe
    $('<iframe src="'+url+'" class="h-100 w-100 border-0" id="challengeWindow"
name="challengeWindow"></iframe>')
     .appendTo('.challengeContainer');

    }
```

Note

チャレンジシナリオをテストしたい場合は、20個のパイナップルをカートに追加するなど、$100以上の商品の購入操作を試みてください。

`iframe` クラスが `h-100` および `w-100` にセットされたことが分かります。これらはそれぞれ、`height: 100%!important` および `width: 100%!important` というcss型を実装するためのブートストラップ・デフォルト・クラスです。これが必要なのは、ACSから与えられた内容に応じて `iframe` のサイズを調整しなければならないためです。チャレンジ画面は次のスクリーンショットのようになるはずです。



> **Info**
>
> 実際の状況でACSが実行するのは、購入金額の確認だけで済むような簡単な処理ではなく、取得したカード会員情報に応じたリスクに基づく複雑な認証処理になります。同様に、認証メソッドはイシュアーにより決定され実行されます。

処理 3: 認証結果の取得

認証結果を取得するためには、フロントエンドが以下のように動作する必要があります。

- **3DSリクエスター**に `認証結果の取得` の為のメッセージを送信する（ステップ.15(F)、またはステップ.20(C)）。
- 結果を画面に表示する（ステップ.17(F)、またはステップ.22(C)）。

認証結果が用意できると、**3DSリクエスター**は `notify_3ds_events.html` の `_onAuthResult` を使用してフロントエンドに通知します。次に、**3ds-web-adapter**は次の図のURL `/auth/result` を使用して `認証結果の取得` リクエストを**3DSリクエスター**に送信します。**3DSリクエスター**が `notify_3ds_events.html` メッセージを処理しているかの確認については、ここを参照してください。

```
function _onAuthResultReady(transId) {
    //redirect to result page
    window.location.href = "/auth/result?txid=" + transId;
```

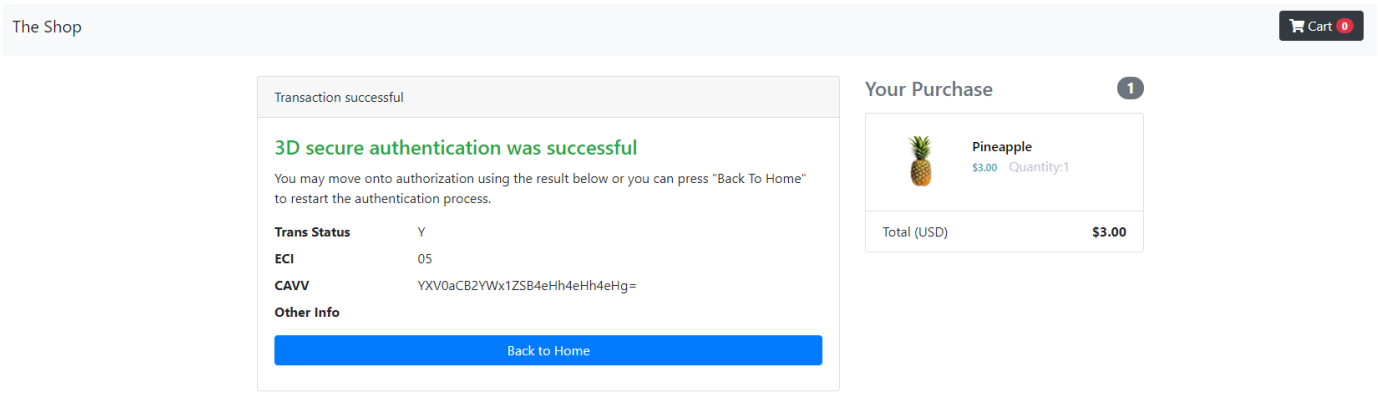
最後に、ブラウザーは認証結果の詳細を表示する `result.html` ページにリダイレクトします。

```
 <div class="col-sm-8">
     <div class="card">
         <div class="card-header">Transaction successful</div>
         <div class="card-body">
             <h4 class="card-title text-success">3D secure authentication was
successful</h4>
             <p class="card-text">You may move onto authorization using the
result below
                  or you can press "Back To Home" to restart authentication
process.</p>

             <dl class="row">
                 <dt class="col-sm-3">Trans Status</dt>
                 <dd class="col-sm-9" data-th-text="${result.transStatus}">Y</
dd>
                 <dt class="col-sm-3">ECI</dt>
                 <dd class="col-sm-9" data-th-text="${result.eci}">eci value</
dd>
                 <dt class="col-sm-3">CAVV</dt>
                 <dd class="col-sm-9" data-th-text="$
{result.authenticationValue}">cavv value</dd>
                 <dt class="col-sm-3">Other Info</dt>
                 <dd class="col-sm-9"></dd>
             </dl>

             <a href="/" class="btn btn-primary">Back To Home</a>
         </div>
     </div>
 </div>
```

新しい結果画面は、次のスクリーンショットのようになります。



Success

お疲れ様でした。これでフロントエンドの導入は完了です。通常、この処理の後は、取引を完了するための、取引ステータス、ECI、CAVVを使用してオーソリの実行に進みます。

次のチャプター

nextボタンを選択し、3DSリクエスター用の**バックエンド導入**についてご覧ください。

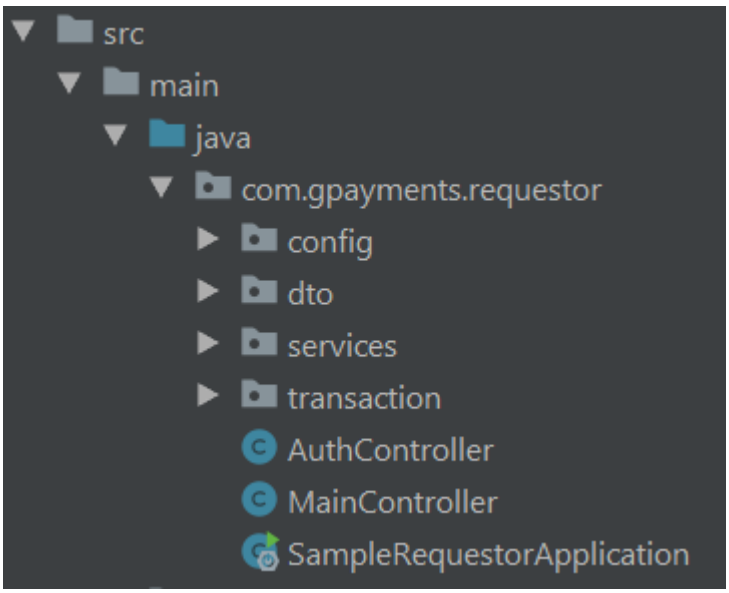
バックエンド実装

この章では、サンプルコードを使用して、加盟店アプリのバックエンドへの実装方法について説明します。

> **Note**
>
> **サンプルコード**に目を通し、実装してから、次に進んでください。

バックエンドには**3DSリクエスター**を実装する必要があります。次の図は、**3DSリクエスター**に含まれる主なファイルを示します。これらはサンプルコードの `/final` ディレクトリの中にあります。必要に応じ**ディレクトリ構想**を確認してください。

`AuthController` は3DS認証用のリクエストを処理するコントローラークラスで、ステップ.2、ステップ.3のように**3DS-web-adapter**からのリクエストを処理して**ActiveServer**に転送します。



処理 1: 認証の初期化

認証を初期化するためには、**3DSリクエスター**が以下のように動作する必要があります。

- **3DS-web-adapter**からの `認証の初期化` リクエストを処理する（ステップ.2、およびステップ.3）。

- 応答メッセージを受信して**3DS-web-adapter**に返送する（ステップ.4、およびステップ.5）。

`AuthController` で、**3DSリクエスター**は、`@PostMapping("/auth/init")` を使用して `initAuth` ハンドラーメソッドを実行し `認証の初期化` リクエストを処理します。フロンドエンドがこのリクエストをどうやって送信しているかの説明は、ここを参照してください。

```
public class AuthController {

  @PostMapping("/auth/init")
  public InitAuthResponseBRW initAuth(@RequestBody InitAuthRequestBRW request) {
    String initBrwUrl = THREE_DS_SERVER_URL + "/api/v1/auth/brw/init/pa";
    // 認証の初期化を/brw/init/{messageCategory} (ステップ. 3)にPOST要求する
    RequestEntity<InitAuthRequestBRW> req =
        new RequestEntity<>(request, HttpMethod.POST, URI.create(initBrwUrl));

    try {
      ResponseEntity<InitAuthResponseBRW> resp =
          restTemplate.exchange(req, InitAuthResponseBRW.class);

      InitAuthResponseBRW initRespBody = resp.getBody();
      logger.info("initAuthResponseBRW {}", initRespBody);

      // set InitAuthResponseBRW for future use
      transactionInfo.setInitAuthResponseBRW(initRespBody);
      return initRespBody;

    } catch (HttpClientErrorException | HttpServerErrorException ex) {

      logger.error("initAuthReq failed, {}, {}", ex.getStatusCode(),
ex.getResponseBodyAsString());

      throw ex;
    }
  }
}
```

`initAuth` ハンドラーメソッドは、まずURL `initBrwUrl = THREE_DS_SERVER_URL + ../brw/init/{messageCategory}` を初期化します（**行5**）。リクエストは、このURLにPOST送信されます(ステップ.3)。

> **Note**
>
> - 文字列 `THREE_DS_SERVER_URL` は、デフォルトの設定で与えられるURLです。 `THREE_DS_SERVER_URL` は、 `Settings > 3D Secure 2` の中にある `AUTH_API_URL` です。詳細は、ここを参照してください。
> - `{messageCategory}` は `pa` （決済認証）または `npa` （非決済認証）のいずれかであり、この例では `pa` を使用しています。

次に、**行11と12**で `initAuth` コントローラーはリクエストをPOST送信して応答を待ちます。リクエストと応答のデータ構造はそれぞれ `InitAuthRequestBRW` 、 `InitAuthResponseBRW` です。

- `InitAuthRequestBRW` - `/brw/init/{messageCategory}` へのAPI呼び出しを行うのに必要なすべての情報を保持しています(ステップ.3)。
- `InitAuthResponseBRW` - `/brw/init/{messageCategory}` へのAPI呼び出しに対する応答に関する情報を保持しています(ステップ.4)。

`InitAuthRequestBRW` と `InitAuthResponseBRW` のデータ構造については、API Documentを参照してください。

> **Note**
>
> **3DSリクエスター**を実装する際には、次の2つの事項が重要です。

> **Tip**
>
> **3DSリクエスター**はSSLを使用して**ActiveServer**に接続します。API呼び出しを行うには、 `RESTTemplate` に**クライアント証明書**を添付する必要があります。
>
> - `/resources/certs` ディレクトリ内のクライアント証明書（ `.p12` ファイル）と `cacerts.jks` トラストストアファイルを確認してください。 `.p12` ファイルの取得についてはIntroductionを参照してください。
> - SSL設定は `RestClientConfig` クラスに実装されています。このクラスは `requestor/config` ディレクトリにあります。

Tip

次の例のコードでは、**3DSリクエスター**は `InitAuthRequestBRW` リクエストを挿入してから**3DSサーバー**に送信しています。

```
//AuthController.java
public InitAuthResponseBRW initAuth(@RequestBody InitAuthRequestBRW request)
{
...
    fillInitAuthRequestBRW(request, THREE_DS_REQUESTOR_URL + "/3ds-notify");
...
}
```

次に、 `AuthController.java` の中にある `fillInitAuthRequestBRW` メソッドについて説明します。このメソッドでは `InitAuthRequestBRW` にデモ用のデフォルトデータを挿入していますが、実際の状況では、データベースからのデータ、あるいはフロントエンドから送信されたカード会員情報をロードするように、このメソッドの部分を入れ替えることができます。

- 文字列 `THREE_DS_REQUESTOR_URL` は、デフォルト設定で与えられるURLであり、 `http://localhost:8082` になります。このURLを、ホストされる3DSリクエスターのURLに更新することができます。
- ここでは、 `eventCallBackUrl` を `THREE_DS_REQUESTOR_URL/3ds-notify` に設定しています。こうすることで、**3DSサーバー**は、ブラウザー情報収集(ステップ.7)の完了時に通知を行えます。

処理 2: 認証の実行

認証を実行するためには、**3DSリクエスター**が以下のように動作する必要があります。

- ステップ.7の後に**ActiveServer**からの `/3ds-notify` メッセージを処理する。
- **3DS-web-adapter**からの `認証の実行` リクエストを処理する（ステップ.9、およびステップ.10）。
- 認証結果を受信して**3DS-web-adapter**に結果を返す（ステップ.12、およびステップ.13）。

ブラウザー情報収集(ステップ.7)が完了したら、**ActiveServer**は、 `THREE_DS_REQUESTOR_URL/3ds-notify` に設定されている `eventCallBackUrl` に `iframe` を通して通知します。**3DSリクエスター**は、 `MainController.java` にてこの通知を処理します。

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

    String callbackName;
    // check the callbackType and initialise callbackName
    if ("3DSMethodFinished".equals(callbackType)) {

        callbackName = "_on3DSMethodFinished";

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

        callbackName = "_on3DSMethodSkipped";

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

    //オブジェクトの情報をnotify_3ds_eventsに送る。
    model.addAttribute("transId", transId);
    model.addAttribute("callbackName", callbackName);
    model.addAttribute("callbackParam", param);

    return "notify_3ds_events";
}
```

このハンドラーメソッドは、パラメーター `transId` と `callbackType` 、およびオプションの `param` を取り込みます。 `callbackType` は、 `3DSMethodFinished` または `3DSMethodSkipped` のいずれかです。ハンドラーメソッドは文字列 `notify_3ds_events` を返します。これは、名前が `notify_3ds_events.html` であるHTMLページを返すことを意味します。フロントエンドでの `notify_3ds_events.html` の実装については、ここを参照してください。

`AuthController` で、**3DSリクエスター**は、 `@PostMapping("/auth")` を使用して `auth` ハンドラーメソッドを実行し `認証の実行` リクエストを処理します(ステップ.9)。このメソッドは、 `threeDSRequestorTransID` と `threeDSServerTransID` を使用して **`/brw`** へのPOST APIリクエストを行います(ステップ.10)。**ActiveServer**は、 `AReq` を初期化して送信することで3DS処理を開始し(ステップ.11)、 `ARes` を受信します。

```
//AuthController.java
@PostMapping("/auth")
public AuthResponseBRW auth(@RequestParam("id") String transId) {

    MerchantTransaction transaction = transMgr.findTransaction(transId);

    //create authentication request.
    AuthRequestBRW authRequest = new AuthRequestBRW();
    authRequest.setThreeDSRequestorTransID(transaction.getId());

authRequest.setThreeDSServerTransID(transaction.getInitAuthResponseBRW().getThreeD

    String brwUrl = THREE_DS_SERVER_URL + "/api/v1/auth/brw";
    AuthResponseBRW response = restTemplate.postForObject(brwUrl, authRequest,
AuthResponseBRW.class);

    logger.info("authResponseBRW {}", response);

    return response;
}
```

`auth` ハンドラーメソッドはフロントエンドに 応答 します。 応答 に含まれている `transStatus` の値は `Y`, `N`, `U`, `R`, `A` または `C` であり、この応答結果に応じて以後のフローを加盟店が決定できます。ステップ毎の実装ガイドにおいては主に `Y` と `C` それぞれフリクションレス、チャレンジのフローについて解説しています。フロントエンドでの `transStatus` の処理については、ここを参照してください。また、 `AuthRequestBRW` と `AuthResponseBRW` のデータ構造については、API documentを参照してください。

処理 3: 認証結果の取得

認証結果を取得するためには、**3DSリクエスター**が以下のように動作する必要があります。

- **3DS-web-adapter**からの 認証結果取得 リクエストを処理する（ステップ.15(F)、またはステップ.20(C)）。
- **ActiveServer**にリクエストを送信して結果を取得し、フロントエンドに結果を返す（ステップ.16(F)、ステップ.17(F)、またはステップ.21(C)、ステップ.22(C)）。

認証結果の準備ができると、**ActiveServer**は `eventCallBackUrl` を使用して通知を送信します。**3DSリクエスター**は `_onAuthResult` メッセージを `notify_3ds_events.html` に返して通知を処理します。フロントエンドでの `_onAuthResult` メッセージの実装については、ここを参照してください。

```
//MainController.java
@PostMapping("/3ds-notify")
public String notifyResult(
    ...
    if ("3DSMethodFinished".equals(callbackType)) {
        callbackName = "_on3DSMethodFinished";
    } else if ("3DSMethodSkipped".equals(callbackType)) {
        callbackName = "_on3DSMethodSkipped";
    } else if ("AuthResultReady".equals(callbackType)) {
        callbackName = "_onAuthResult";
    }
    ...
```

3DSリクエスターは、`@GetMapping("/auth/result")` を使用して `result` ハンドラーメソッドを実行し 認証結果を取得 します。このハンドラーメソッドは、`/brw/result` を呼び出して **ActiveServer**に認証結果を要求し、フロントエンドに `result` を返します。フロントエンドは `result.html` ページを使用して結果を表示します。フロントエンドでの実装については、ここを参照してください。

```
@GetMapping("/auth/result")
public String result(
        @RequestParam("txid") String transId,
        Model model) {

    MerchantTransaction transaction = transMgr.findTransaction(transId);
    String resultUrl = AuthController.THREE_DS_SERVER_URL + "/api/v1/auth/brw/
result?threeDSServerTransID=" +

transaction.getInitAuthResponseBRW().getThreeDSServerTransID();
    AuthResponseBRW response = restTemplate.getForObject(resultUrl,
AuthResponseBRW.class);
    model.addAttribute("result", response);
    return "result";
}
```

Success

お疲れ様でした。これでバックエンドの導入は完了です。

次のチャプター

「次へ」ボタンを選択し、ステップ毎の実装ガイドに従って加盟店のウェブサイトに**3DSリクエスター**を統合してください。

ステップ毎の実装ガイド

この章では、認証処理を実行して、序章でダウンロードした簡単なショッピングサイトのプロジェクトに3DS2フローを組み込む方法を説明します。この章の説明は、同じ機能を既存の決済処理に追加する場合のガイドとしても使用できます。

このチュートリアルを通して、Intellij IDEAを使用します。

完成したコードについてさらに詳しく学習するには、`/final` ディレクトリにあるGPaymentsのサンプルコードの、3DS2が組み込まれた最終的な加盟店決済ページにアクセスしてください。デモ用の3DSリクエスターのコードの概要は、最終的なデモコードの章を参照してください。

> **Info**
>
> 本書の目的は、決済ページがJava以外の別のプログラミング言語で実装されていても、決済ページに3DSリクエスターを実装できるように、3DSリクエスターの基本的な流れを理解するのを助けることにあります。

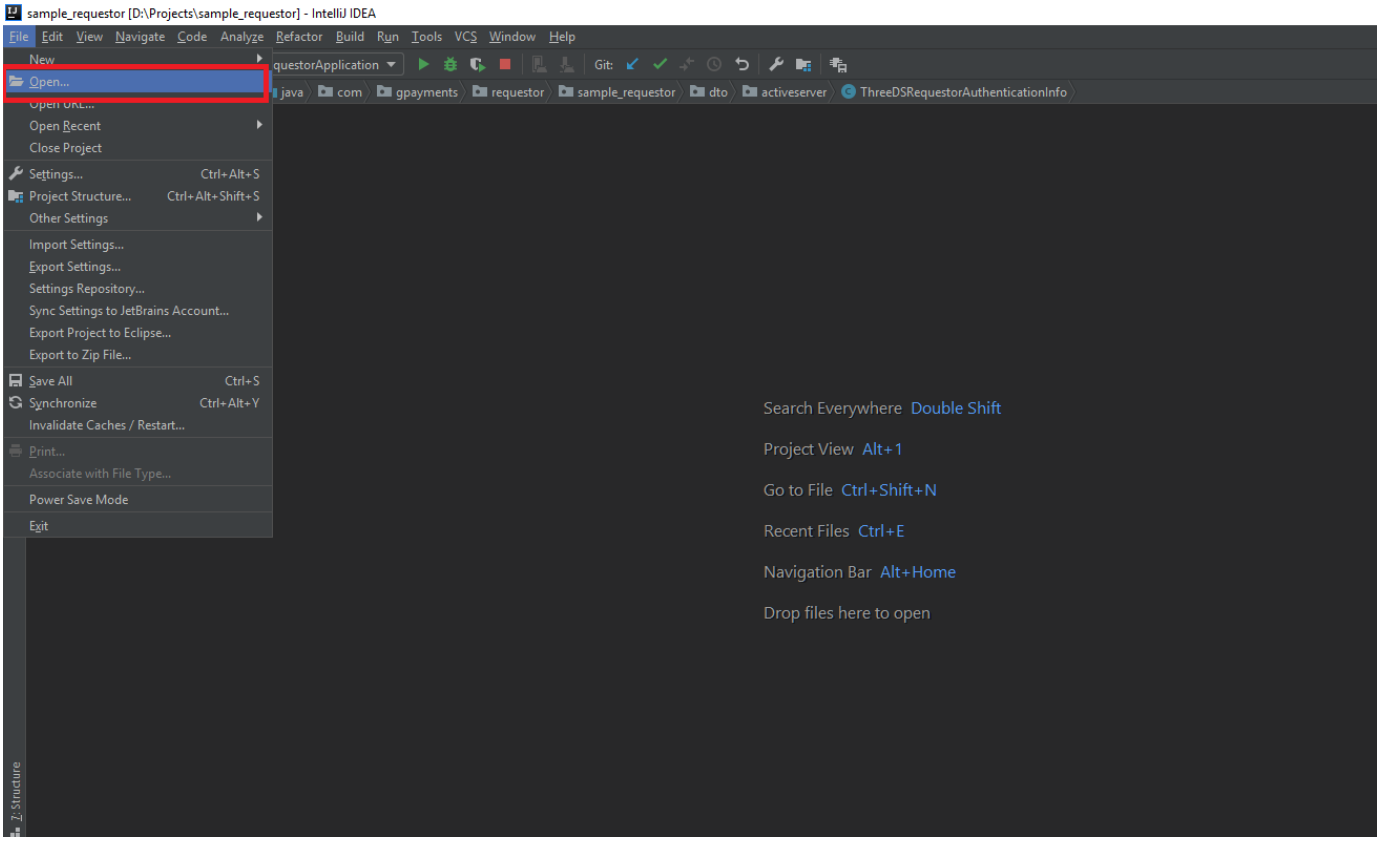
処理 1: 認証の初期化

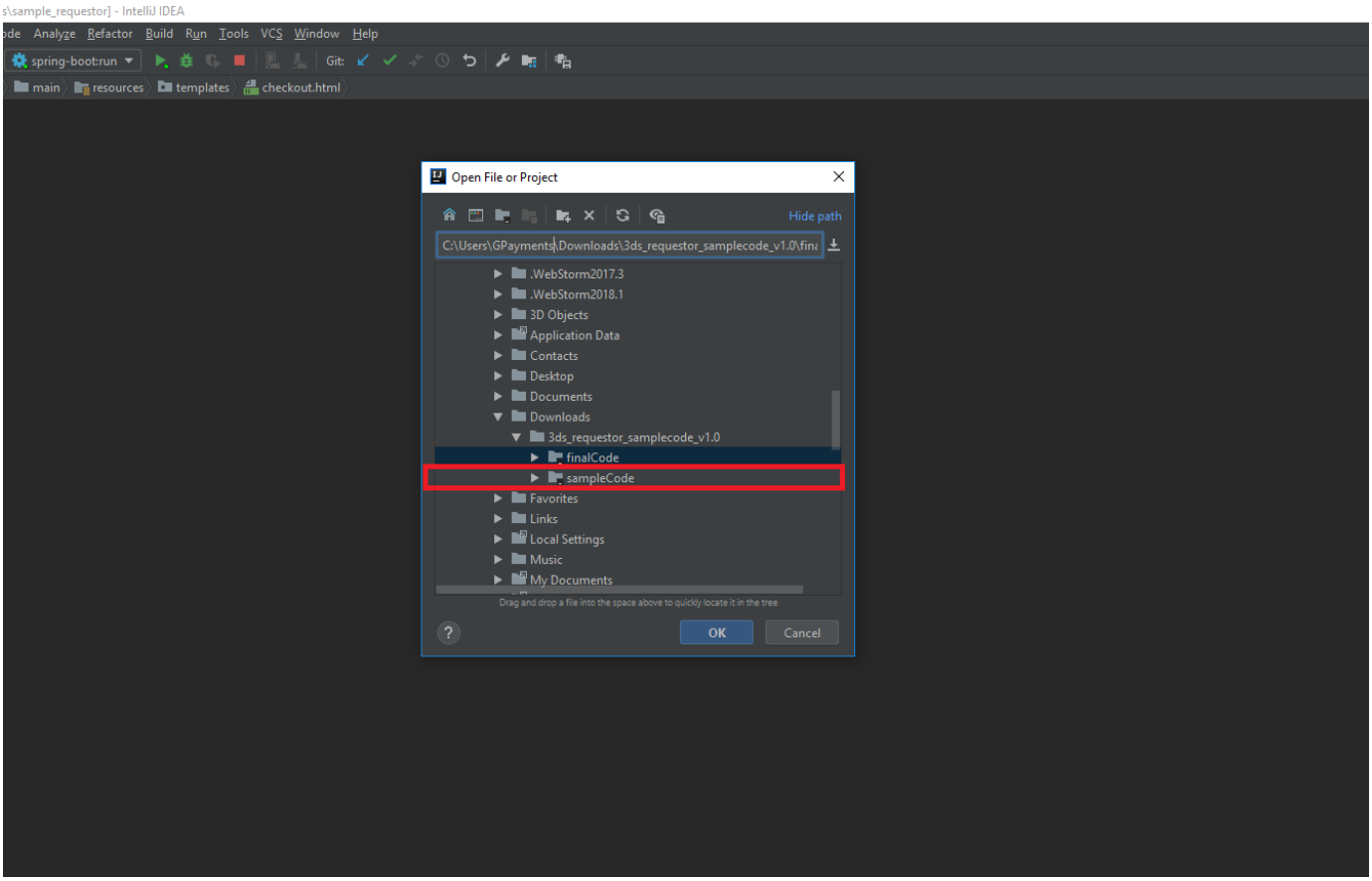
まず、認証を初期化する手順、すなわち `/brw/init/{messageCategory}` を呼び出す手順を説明します。

`messageCategory` は `pa` （決済認証）または `npa` （非決済認証）のいずれかです。

> **Tip**
>
> 各APIのエンドポイントは認証APIというドキュメント内のAPIに対応する参照先にリンクしており、APIの使用方法に関する詳細情報にアクセスできます。

GPaymentsのサンプル・コードには、`/initial` というディレクトリが含まれています。自分のIDEを使用してこのディレクトリを**開き**ます（**File > Open > Browse to initial folder > OK**と操作します）。





また、認証処理の実装に役立つ `3ds-web-adapter.js` も用意されています。

`/final` プロジェクトから最初のプロジェクトの**/resources/static/js**ディレクトリに `3ds-web-adapter.js` を**コピーし貼り付け**ます。必要に応じ `js` ディレクトリを作成します。このファイルの場所がよく分からない場合は、ディレクトリ構想を確認してください。

次に示す強調表示の行を**追加**し、 `3ds-web-adapter.js` ファイルを `checkout.html` ページにインポートします。

```
....
<script src="https://code.jquery.com/jquery-3.3.1.min.js"
        integrity="sha256-FgpCb/KJQlLNfOu91ta32o/NMZxltwRo8QtmkMRdAu8="
        crossorigin="anonymous"></script>
<script src="https://cdnjs.cloudflare.com/ajax/libs/popper.js/1.14.3/umd/
popper.min.js"
        integrity="sha384-ZMP7rVo3mIykV+2+9J3UJ46jBk0WLaUAdn689aCwoqbBJiSnjAK/
l8WvCWPIPm49"
        crossorigin="anonymous"></script>
<script src="https://stackpath.bootstrapcdn.com/bootstrap/4.1.3/js/
bootstrap.min.js"
        integrity="sha384-ChfqqxuZUCnJSK3+MXmPNIyE6ZbWh2IMqE241rYiqJxyMiZ6OW/
JmZQ5stwEULTy"
        crossorigin="anonymous"></script>
<script src="/js/3ds-web-adapter.js"></script>
.....
```

アダプターを使用できるよう `checkout.html` を編集します。 `checkout.html` の `checkout()` 関数を確認してください。次のコードのようになっているはずです。この関数は、JQueryを使用してカード会員入力フィールドの値をHTML文書から取得し、新しい変数 `transId` 、 `cardHolderInfo` 、 `purchaseInfo` を初期化します。

```
//checkout.html
function checkout() {
    var transId = $('#transId').val();

    // NOTE: Some attributes are set to default values for demo purpose
    var cardHolderInfo = {};

    if($('#billAddrLine1').val()) {
        cardHolderInfo.billAddrLine1 = $('#billAddrLine1').val();
    }

    ....

    if($('#cardExpiry').val()) {
        purchaseInfo.expiryDate = $('#expiryDate').val();
    }
    //remove cardholder information class, checkout button and show spinner
effect
    $("#checkoutButton").remove();

    $("#cardholderInfoCard").remove();
    $("#checkoutCard").removeClass("d-none");

    //move to result screen 2 seconds after for demo purposes
    setTimeout(function () {
        window.location.href = "/auth/result"
    }, 2000);
}
```

`checkout.html` の `checkout()` に、認証処理の開始時に `threeDSAuth()` メソッド(ステップ.1)を呼び出すための強調表示の行を**追加**します。

```
function checkout() {
    ...
    $("#cardholderInfoCard").remove();
    $("#checkoutCard").removeClass("d-none");

    //move to result screen 2 seconds after
    setTimeout(function () { window.location.href = "/auth/result"}, 2000);
    threeDSAuth(transId, cardHolderInfo, purchaseInfo);
}
```

次に示す強調表示の行を**コメントアウトまたは削除**してください。この行はデモの目的でのみここに挿入されているもので、結果画面に移動する前に2秒の遅延時間を確保します。

```
function checkout() {
    ...
    $("#cardholderInfoCard").remove();
    $("#checkoutCard").removeClass("d-none");

    //setTimeout(function () { window.location.href = "/auth/result"}, 2000);
    threeDSAuth(transId, cardHolderInfo, purchaseInfo);
}
```

Note

次に示すコードは、`3ds-web-adapter.js` で定義されている `threeDSAuth()` 関数です。このコードから分かるように、`threeDSAuth()` はJSON形式でページから `/auth/init` にオブジェクトを送信し（**行6**）、たとえば `threeDSRequestorTransID` は `transId` により初期化されています。

```
//3ds-web-adapter.js
function threeDSAuth(transId, cardHolderInfo, purchaseInfo) {

    var postData = {...}

    ....

    console.log('init authentication');
    $.ajax({
        url: '/auth/init',
        type: 'POST',
        contentType: "application/json",
        data: JSON.stringify(postData),
        success: function (data) {
            console.log('init auth returns:', data);
            $('<iframe id="threeds-container" width="0" height="0"
            style="visibility: hidden;"
            src="' + data.threeDSServerCallbackUrl + '"></iframe>')
                .appendTo('.challengeContainer');
        },
        error: function () {
            alert('error');
        },
        dataType: 'json'

    });

}
```

ユーザーの必要に応じ `threeDSAuth()` を変更したり、`cardholderInfo`、`purchaseInfo`、`transId` でなくユーザー独自のオブジェクトを転送できます。送信できるデータ要素の型は、最終的なコードの**/dto/activeserver**ディレクトリの中の `InitAuthRequestBRW` に示されているほか、APIドキュメントにも示されています。

`3ds-web-adapter.js` は `/auth/init` へのPOSTリクエストを行いますので(ステップ.2)、このリクエストを処理するための新しいコントローラークラスが必要です。

クライアントサイドからのリクエストを処理するためには、`/auth/init` リクエストを処理するコントローラーを作成する必要があります。**/java/sample_requestor**ディレクトリ内に新しいコントローラークラス `AuthController.java` を**作成**してください。さらに、`/auth/init` リクエス

トを処理し認証APIエンドポイント `/api/v1/{messageCategory}` を呼び出す `initAuth` メソッドを**コピーし貼り付け**ます。

```
public class AuthController {
  @PostMapping("/auth/init")
  public InitAuthResponseBRW initAuth(@RequestBody InitAuthRequestBRW request) {
    String initBrwUrl = THREE_DS_SERVER_URL + "/api/v1/auth/brw/init/
{messageCategory}";
    // 認証の初期化 by making  POST request to /brw/init/{messageCategory} (ステッ
プ. 3)
    RequestEntity<InitAuthRequestBRW> req =
        new RequestEntity<>(request, HttpMethod.POST, URI.create(initBrwUrl));

    try {
      ResponseEntity<InitAuthResponseBRW> resp =
          restTemplate.exchange(req, InitAuthResponseBRW.class);

      InitAuthResponseBRW initRespBody = resp.getBody();
      logger.info("initAuthResponseBRW {}", initRespBody);

      // set InitAuthResponseBRW for future use
      transactionInfo.setInitAuthResponseBRW(initRespBody);
      return initRespBody;

    } catch (HttpClientErrorException | HttpServerErrorException ex) {

      logger.error("initAuthReq failed, {}, {}", ex.getStatusCode(),
ex.getResponseBodyAsString());

      throw ex;
    }
  }
}
```

決済認証を初期化したい場合は `{messageCategory}` を `pa` に、非決済認証を初期化したい場合は `{messageCategory}` を `npa` に**置換**します。この例では `pa` を使用しています。

```
//AuthController.java
String initBrwUrl = THREE_DS_SERVER_URL + "/api/v1/auth/brw/init/pa";
// 認証の初期化 by making  POST request to /brw/init/{messageCategory} (ステップ.
3)
InitAuthResponseBRW initAuthResponseBRW = restTemplate.postForObject(initBrwUrl,
    request, InitAuthResponseBRW.class);

....
```

文字列 `THREE_DS_SERVER_URL` を、デフォルト設定で与えられるURLに**置換**します。
`THREE_DS_SERVER_URL` は、 `Settings > 3D Secure 2` の中にある `AUTH_API_URL` です。詳細は、ここを参照してください。

```
//AuthController.java
private final String THREE_DS_SERVER_URL = "https://api.as.testlab.
3dsecure.cloud:7443";
....
```

文字列 `THREE_DS_REQUESTOR_URL` は、デフォルト設定で与えられるURLに**置換**するか、ホスト対象の3DSリクエスターのURLに更新します。

```
//AuthController.java
private final String THREE_DS_REQUESTOR_URL = "http://localhost:8082";
....
```

`activeserver` ディレクトリを `/java/sample_requestor/dto` ディレクトリに**コピーし貼り付け**ます。このディレクトリには、API呼び出しを行うのに必要なすべてのクラスが含まれています。認証を初期化するために最も重要なのは、以下のクラスです。

- `InitAuthRequestBRW` - **`/brw/init/{messageCategory}`** へのAPI呼び出しを行うのに必要なすべてのデータを保持しています(ステップ.3)。
- `InitAuthResponseBRW` - **`/brw/init/{messageCategory}`** へのAPI呼び出しに対する応答に関する情報を保持しています(ステップ.4)。

次に、 `AuthController.java` の中にある `fillInitAuthRequestBRW` メソッドについて説明します。このメソッドでは `InitAuthRequestBRW` にデモ用のデフォルトデータを挿入していますが、実際の状況では、データベースからのデータ、あるいは `checkout.html` から送信されたカード会員情報をロードするように、このメソッドの部分を入れ替えることができます。

```
//AuthController.java
 /**
 * This method is to fill in the InitAuthRequestBRW with demo data, you need to
fill the information from your database
 * @param initAuthRequestBRW
 * @param eventCallBackUrl
 */
private void fillInitAuthRequestBRW(InitAuthRequestBRW initAuthRequestBRW,
String eventCallBackUrl) {

    initAuthRequestBRW.setAcctID("personal account");

    // Fill AcctInfo with default data.
    AcctInfo acctInfo = new AcctInfo();
    acctInfo.setChAccAgeInd("03");
    acctInfo.setChAccChange("20160712");
    acctInfo.setChAccChangeInd("04");
    acctInfo.setChAccDate("20140328");
    acctInfo.setChAccPwChange("20170328");
    acctInfo.setChAccPwChangeInd("02");
    acctInfo.setNbPurchaseAccount("11");
    acctInfo.setPaymentAccAge("20160917");
    acctInfo.setPaymentAccInd("02");
    acctInfo.setProvisionAttemptsDay("3");
    acctInfo.setShipAddressUsage("20160714");
    acctInfo.setShipAddressUsageInd("02");
    acctInfo.setShipNameIndicator("02");
    acctInfo.setSuspiciousAccActivity("02");
    acctInfo.setTxnActivityDay("1");
    acctInfo.setTxnActivityYear("21");
    initAuthRequestBRW.setAcctInfo(acctInfo);

    initAuthRequestBRW.setAcctType("03");
    initAuthRequestBRW.setAuthenticationInd("01");//01 = Payment transaction

    // fills ThreeDSRequestorAuthenticationInfo
    ThreeDSRequestorAuthenticationInfo threeDSRequestorAuthenticationInfo = new
ThreeDSRequestorAuthenticationInfo();
    threeDSRequestorAuthenticationInfo.setThreeDSReqAuthData("login GP");
    threeDSRequestorAuthenticationInfo.setThreeDSReqAuthMethod("02");

threeDSRequestorAuthenticationInfo.setThreeDSReqAuthTimestamp("201711071307");

initAuthRequestBRW.setAuthenticationInfo(threeDSRequestorAuthenticationInfo);

    // fills MerchantRiskIndicator, optional but strongly recommended for the
accuracy of risk based authentication
    MerchantRiskIndicator merchantRiskIndicator = new MerchantRiskIndicator();
    merchantRiskIndicator.setDeliveryEmailAddress("test@123.com");
    merchantRiskIndicator.setDeliveryTimeframe("02");
```

```
    merchantRiskIndicator.setGiftCardAmount("123");
    merchantRiskIndicator.setGiftCardCount("02");
    merchantRiskIndicator.setGiftCardCurr("840");
    merchantRiskIndicator.setPreOrderDate("20170519");
    merchantRiskIndicator.setPreOrderPurchaseInd("02");
    merchantRiskIndicator.setReorderItemsInd("01");
    merchantRiskIndicator.setShipIndicator("01");

    initAuthRequestBRW.setMerchantRiskIndicator(merchantRiskIndicator);

    /**
     * Options for threeDSRequestorChallengeInd - Indicates whether a challenge
is requested for this transaction.
     * Values accepted:
     *  01 = No preference
     *  02 = No challenge requested
     *  03 = Challenge requested: 3DSリクエスター Preference
     *  04 = Challenge requested: Mandate
     *  05–79 = Reserved for EMVCo future use (values invalid until defined by
EMVCo)
     *  80-99 = Reserved for DS use
     */
    initAuthRequestBRW.setChallengeInd("01");
    initAuthRequestBRW.setEventCallbackUrl(eventCallBackUrl); //Set this to
your url
    initAuthRequestBRW.setMerchantId("123456789012345");

initAuthRequestBRW.setPurchaseDate(LocalDateTime.now().format(DateTimeFormatter.of
    initAuthRequestBRW.setPurchaseInstalData("24");
    initAuthRequestBRW.setRecurringExpiry("20180131");
    initAuthRequestBRW.setRecurringFrequency("6");
    initAuthRequestBRW.setTransType("03");
}
```

`MerchantTransaction.java` 、および対応する `getter` 、 `setter` メソッドに、新しい変数 `initAuthResponseBRW` を**追加**します。

```
//MerchantTransaction.java
import com.gpayments.requestor.requestor.dto.activeserver.InitAuthResponseBRW;

public class MerchantTransaction {

    private InitAuthResponseBRW initAuthResponseBRW;

    ...

    //getter
    public InitAuthResponseBRW getInitAuthResponseBRW() {
        return initAuthResponseBRW;
    }

    //setter
    public void setInitAuthResponseBRW(InitAuthResponseBRW initAuthResponseBRW)
{
        this.initAuthResponseBRW = initAuthResponseBRW;
    }
```

ActiveServerへのAPI呼び出しを行うには、RESTTemplateにクライアント証明書を添付する必要があります。

GPaymentsから与えられたクライアント証明書（.p12ファイル）を**/resources/certs**ディレクトリに**コピーし貼り付け**ます。ディレクトリがない場合は `/certs` ディレクトリを作成します。証明書ファイルをまだ受け取っていない場合は、**お問合わせ**ください。ActiveServerが社内ソフトウェアである場合は、管理者UIからクライアント証明書をダウンロードできます。クライアント証明書のダウンロードに関する詳細は、加盟店のセキュリティーを参照してください。

最終コードに含まれている `cacerts.jks` トラストストアファイルを**/resources/certs**ディレクトリに**コピーし貼り付け**ます。証明書が必要なのは、相互SSL認証を可能にするために3DSリクエスターと3DSサーバーの相互認証が必要だからです。

`final` パッケージに含まれている設定クラス `RestClientConfig.java` を**/java/sample_requestor/config**ディレクトリに**コピーし貼り付け**ます。次のクラスは、クライアント証明書をHttpClientにロードする方法の例です。このクラスは、GPaymentsから与えられたトラストストアを使用してユーザーがActiveServerからダウンロードしたキーストアファイルをロードします。

```
//RestClientConfig.java
@Configuration
public class RestClientConfig {

  private static final String KEYSTORE_PASSWORD = "123456";
  private static final String CA_CERTS_FILE_NAME = "certs/cacerts.jks";
  private static final String CLIENT_CERTS_FILE_NAME = "certs/
client_certificate.p12";

  @Bean
  public RestTemplate restTemplate()
      throws IOException, UnrecoverableKeyException, CertificateException,
NoSuchAlgorithmException,
      KeyStoreException, KeyManagementException {
    SSLContext sslContext =
        SSLContextBuilder.create()
            .loadKeyMaterial(
                new ClassPathResource(CLIENT_CERTS_FILE_NAME).getURL(),
                KEYSTORE_PASSWORD.toCharArray(),
                KEYSTORE_PASSWORD.toCharArray())
            .loadTrustMaterial(
                new ClassPathResource(CA_CERTS_FILE_NAME).getURL(),
KEYSTORE_PASSWORD.toCharArray())
            .build();

    CloseableHttpClient client =
        HttpClients.custom()
            .setSSLContext(sslContext)
            .build();
    HttpComponentsClientHttpRequestFactory httpRequestFactory =
        new HttpComponentsClientHttpRequestFactory(client);

    return new RestTemplate(httpRequestFactory);
  }
}
```

`KEYSTORE_PASSWORD` を、デフォルト設定で与えられるパスワードに**置換**します。ActiveServer が社内ソフトウェアである場合、このパスワードは、管理者ダッシュボードの加盟店ページからクライアント証明書をダウンロードするときに指定したパスワードです。クライアント証明書のダウンロードに関する詳細は、加盟店のセキュリティーを参照してください。

```
//RestClientConfig.java
private String KEYSTORE_PASSWORD = "123456";
```

> **Info**
>
> これは、クライアント証明書を使用してRESTFul APIリクエストを行う、`Java`に固有な方法です。別の言語を使用して、3DSリクエスターが3DSサーバーと相互認証するためのサーバーサイドコードを実装する場合も、同様な手順に従う必要があります。

次のような依存関係を`pom.xml`に**追加**し、`SSLContextBuilder`クラスに必要なApache HttpClientを取得します。

```
<dependencies>
    ...
    <dependency>
        <groupId>org.apache.httpcomponents</groupId>
        <artifactId>httpclient</artifactId>
        <version>4.5.6</version>
    </dependency>
</dependencies>
```

必要なすべてのクラスがインポートされ、エラーがないことを確認してください。

もう一度**アプリを実行**してhttp://localhost:8082からアクセスし、決済処理を実行します。`/api/v1/auth/brw/init/{messageCategory}`に対応する正しい応答が得られ、次のようにログの`threeDSServerCallbackUrl`が有効になっているはずです。

```
{
    "threeDSServerCallbackUrl":"https://admin.as.testlab.3dsecure.cloud/api/v1/
auth/brw/callback?transId=cbc559c0-96bd-4078-be3c-
ea0cc80686f0&t=ZIIgV2LqMak5ONQ1w3l3akfpZlMPftFlOE5Garat6lHiJ3T2Kq2vULuLhIQ7l5UKzpn
    "threeDSServerTransID":"ade87add-b50c-42da-bbea-cdcbd85dcbf3"
}
```

`3ds-notify`の部分で決済処理が停止しますが、これは、この部分がまだ実装されていないためであり、この部分については次の章で対応します。

Tip

変更内容がプロパゲートされるためには、プロジェクトを再実行する必要があります。`CTRL+C` を使用して現在実行中のアプリを終了し、コマンドラインに次のように入力してプロジェクトを再実行します。

```
mvn spring-boot:run
```

3DSサーバーはコールバックURLを使用して、ACSは3DSメソッドを使用して、リスクに基づく認証に必要なブラウザー情報を収集します。

Note

`/auth/init` からの応答が正常に得られると(ステップ.5)、`3ds-web-adapter.js` は非表示の `iframe` を決済ページに埋め込み(ステップ.6)、ACSと3DSサーバーは `threeDSServerCallbackUrl` を使用してブラウザー情報を収集できるようになります(ステップ.7)。

```
//3ds-web-adapter.js
success: function(data) {
    $('<iframe id="threeds-container" width="0" height="0"
style="visibility: hidden;" src="'+ data.threeDSServerCallbackUrl+'"></
iframe>')
    .appendTo('.challengeContainer');
},
```

Info

シーケンス図のステップ.1～ステップ.7は、この処理が終了すると実行されます。

処理 2: 認証の実行

前のステップで、`3ds-web-adapter.js` は、コールバック `src` が `threeDSServerCallbackUrl` にセットされた `iframe` を挿入しています。このため、3DSサーバーはブラウザーへのコールバックを行って(ステップ.7)、必要なブラウザー情報を収集できます。

また、`eventCallBackUrl` を `THREE_DS_REQUESTOR_URL/3ds-notify` にセットして **`/brw/init/{messageCategory}`** を呼び出しています**(line.2)**。これにより、3DSサーバーは、ブラウザー情報の収集が完了したときに通知を行えます。

```
//AuthController.java
fillInitAuthRequestBRW(request, THREE_DS_REQUESTOR_URL + "/3ds-notify");
.....
```

したがって、このAPI呼び出しを受信するためのハンドラーメソッドを作成しておく必要があります。

以下のコードを `MainController.java` に**追加**します。 このハンドラーメソッドは、パラメーター `transId` と `callbackType` 、およびオプションの `param` を取り込みます。 `callbackType` は、 `3DSMethodFinished` または `3DSMethodSkipped` のいずれかです。ハンドラーメソッドは文字列 `notify_3ds_events` を返します。Springbootに慣れていない方向けに説明すると、これは、**/resources/templates**ディレクトリから名前の一致するHTMLページを返すことを意味します。

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

    String callbackName;
    // check the callbackType and initialise callbackName
    if ("3DSMethodFinished".equals(callbackType)) {

        callbackName = "_on3DSMethodFinished";

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

        callbackName = "_on3DSMethodSkipped";

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

    //Pass on the object to the page
    model.addAttribute("transId", transId);
    model.addAttribute("callbackName", callbackName);
    model.addAttribute("callbackParam", param);

    return "notify_3ds_events";
}
```

`/3ds-notify` への呼び出しにより返されたページである `callbackFn` を呼び出すための `notify_3ds_events.html` という新しいHTMLファイルを、**/resources/templates**ディレクトリ内に**作成**します。

```
<!--notify_3ds_events.html-->
<!DOCTYPE html>
<html lang="en">
<head>
    <meta charset="UTF-8"/>
    <title>3DSecure 2.0 Authentication</title>
</head>
<body>

<form>
    <input type="hidden" id="notifyCallback" name="notifyCallback" data-th-
value="${callbackName}"/>
    <input type="hidden" id="transId" name="transId" data-th-value="${transId}"/
>
    <input type="hidden" id="param" name="param" data-th-value="$
{callbackParam}"/>
</form>

<script src="https://code.jquery.com/jquery-3.3.1.min.js"
        integrity="sha256-FgpCb/KJQlLNfOu91ta32o/NMZxltwRo8QtmkMRdAu8="
        crossorigin="anonymous"></script>
<script>

//notify parent checkout page to proceed with rest procedures.

var callbackFn = parent[$('#notifyCallback').val()];

//callbackFn is defined in 3ds-notify handler method
if (typeof callbackFn === 'function') {
    callbackFn($('#transId').val(),$('#param').val());
}

</script>

</body>
</html>
```

3DSサーバーから与えられる `callbackName` によって、 `3ds-web-adapter.js` で呼び出されるメソッドが異なることが分かります(ステップ.8)。 各メソッドについて以下に説明します。

- `_onThreeDSMethodFinished` - 3DSメソッドが完了したことを通知し、ブラウザー情報の収集が完了したことを示す `_doAuth()` を呼び出します。

- `_onThreeDSMethodSkipped` - 3DSメソッドをスキップしたことを通知し、ACSによるブラウザー情報の収集をスキップしたことを示す `_doAuth()` を呼び出します。

`3ds-web-adapter` の `_doAuth()` は、 `/auth` へのPOSTリクエストを行い認証を実行します(ステップ.9)。

このリクエストを処理する新しいハンドラーメソッドを `AuthController.java` に**追加**します。このメソッドは、 `threeDSRequestorTransID` と `threeDSServerTransID` を使用して **`/brw`** へのPOST APIリクエストを行うはずです(ステップ.10)。続いて3DSサーバーは、AReqを作成、送信し、AResを受信するとこれを処理して3DS処理を開始します(ステップ.11)。

```
//AuthController.java
@PostMapping("/auth")
public AuthResponseBRW auth(@RequestParam("id") String transId) {

    MerchantTransaction transaction = transMgr.findTransaction(transId);

    //create authentication request.
    AuthRequestBRW authRequest = new AuthRequestBRW();
    authRequest.setThreeDSRequestorTransID(transaction.getId());

authRequest.setThreeDSServerTransID(transaction.getInitAuthResponseBRW().getThreeD

    String brwUrl = THREE_DS_SERVER_URL + "/api/v1/auth/brw";
    AuthResponseBRW response = restTemplate.postForObject(brwUrl, authRequest,
AuthResponseBRW.class);

    logger.info("authResponseBRW {}", response);

    return response;
}
```

アプリを再実行し、デフォルト値を使用して決済処理を実行すると、 `/api/v1/auth/brw` に対応する正しい応答が得られ(ステップ.12)、transStatusが `Y` にセットされているはずです。

```
{
    "authenticationValue":"YXV0aCB2YWx1ZSB4eHh4eHh4eHg=",
    "eci":"06",
    "threeDSServerTransID":"52056514-7504-40c7-886f-2a0452d8edbd",
    "transStatus":"Y"
}
```

チャレンジシナリオをテストしたい場合は、ここのセクションを参照ください

`/brw` へのAPI呼び出しに対する応答では、 `challengeUrl` がセットされ、 `transStatus` が `C` にセットされます。

```
{
    "acsChallengeMandated":"Y",
    "authenticationType":"01",
    "challengeUrl":"https://admin.as.testlab.3dsecure.cloud/api/v1/auth/brw/
challenge/init?txid=78cd5068-96ea-48c4-b013-e6843fa8b2e4",
    "threeDSServerTransID":"c8a32fb7-9556-4242-896b-562ee8ca25df",
    "transStatus":"C",
}
```

チャレンジウィンドウで要求されるワンタイムパスコードは**123456**です。エラーが発生しますが、これは `callbackType` 、 `AuthResultReady` の処理機能がまだ実装されていないためであり、この部分は次の章で実装します。

Note

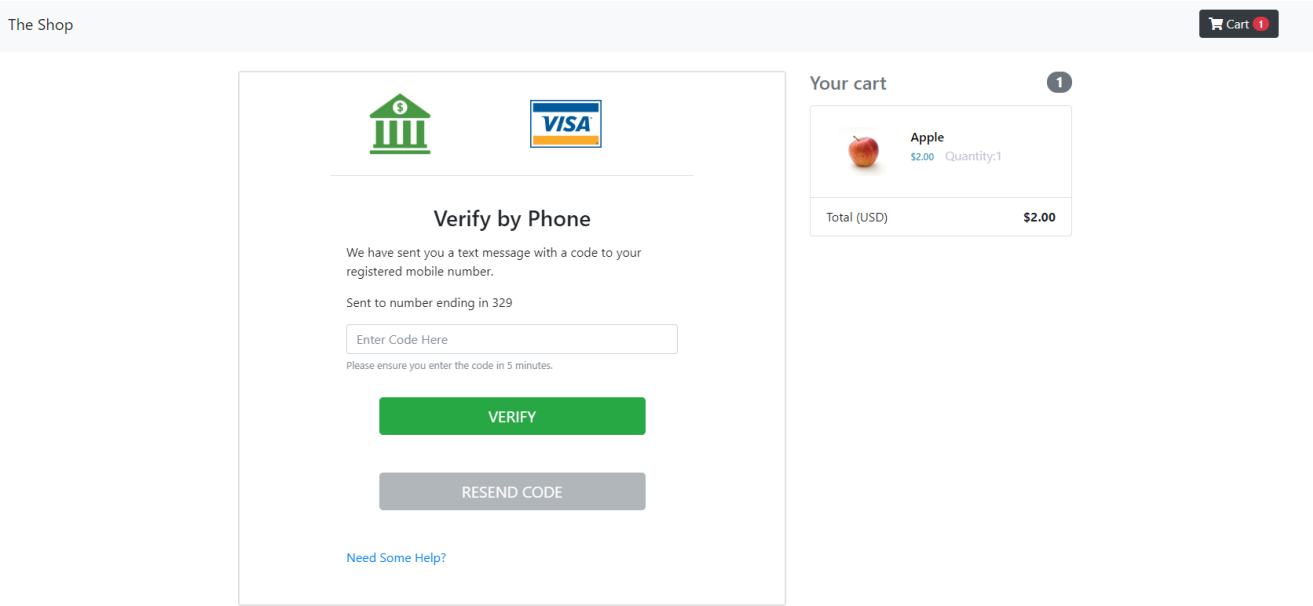
`transStatus` が `C` の場合、`3ds-web-adapter.js` は `startChallenge(url)` を呼び出し、`src` が `challengeUrl` にセットされた `iframe` をチャレンジウィンドウに挿入します(ステップ.14(C))。

```
function startChallenge(url) {
//remove the spinner
$(".spinner").remove();

//create the iframe
$('<iframe src="'+url+'" class="h-100 w-100 border-0" id="challengeWindow"
name="challengeWindow"></iframe>')
.appendTo('.challengeContainer');

}
```

`iframe` クラスが `h-100` および `w-100` にセットされたことが分かります。これらはそれぞれ、`height: 100%!important` および `width: 100%!important` というcss型を実装するためのブートストラップ・デフォルト・クラスです。これが必要な理由は、ACSから与えられた内容に応じて `iframe` のサイズを調整する必要があるためです。チャレンジ画面は次のスクリーンショットのようになるはずです。



Info

実際の状況でACSが実行するのは、購入金額の確認だけで済むような簡単な処理ではなく、取得したカード会員情報に応じたリスクに基づく複雑な認証処理になります。同様に、認証メソッドはイシュアーにより決定され実行されます。

処理 3: 認証結果の取得

前のステップで説明したように、ここでカード会員に表示できるように認証結果の更新を要求する必要があります。フリクションレス・フローの場合も同様に更新を要求する必要があります（次の警告の項を参照）。

> **認証結果を別に取得しなければいけない理由**
>
> フリクションレス・フローの場合、利用可能な認証結果がステップ.12ですでに取得できているのに、なぜもう一度結果を要求する必要があるのか、不思議に思われるかもしれません。
>
> これが必要なのは、ステップ.13で認証結果がページに転送されるためです。`3ds-web-adapter` が結果を3DSリクエスターに送り返すことも可能ですが、データが安全でないと見なされます。サーバー側では常に、元の情報源である3DSサーバーから戻される結果をサーバー自身の機能により取得できなければなりません。このステップで結果の受信を要求する必要があるのは、このためです。

チャレンジ・フローの場合は、3DSメソッドのステータスの通知の場合と同様に、認証結果は `3ds-notify` のエンドポイントを介して3DSサーバーから通知されます。

`MainController.java` で"AuthResultReady" `callbackType` を処理できるよう、次に示す強調表示の新しい `else if` ステートメントを**追加**します。

```
//MainController.java
if ("3DSMethodFinished".equals(callbackType)) {
    callbackName = "_on3DSMethodFinished";
} else if ("3DSMethodSkipped".equals(callbackType)) {
    callbackName = "_on3DSMethodSkipped";
} else if ("AuthResultReady".equals(callbackType)) {
    callbackName = "_onAuthResult";
}
```

`notify-3ds-events` から `_onAuthResultReady` が呼び出され、ウィンドウを `/auth/result?txid=+transId` にリダイレクトします。 認証結果を取得したい場合は、 **`/brw/result`** を呼び出して、ハンドラー `/auth/result` の中で結果の受信を要求できます。

```
@GetMapping("/auth/result")
public String result(
        @RequestParam("txid") String transId,
        Model model) {

    MerchantTransaction transaction = transMgr.findTransaction(transId);
    String resultUrl = AuthController.THREE_DS_SERVER_URL + "/api/v1/auth/brw/
result?threeDSServerTransID=" +

transaction.getInitAuthResponseBRW().getThreeDSServerTransID();
    AuthResponseBRW response = restTemplate.getForObject(resultUrl,
AuthResponseBRW.class);
    model.addAttribute("result", response);
    return "result";
}
```

`result.html` の中の強調表示になっているコードを**Delete or comment out**します。

```
<div class="col-sm-8">
    <div class="card">
        <div class="card-header">Transaction successful</div>
        <div class="card-body">
            <!--<h4 class="card-title text-success">Thank you for your purchase!
</h4>-->
            <!--<p class="card-text">Want to continue shopping? Press "Back to
Home".</p>-->

            <a href="/" class="btn btn-primary btn-block">Back to Home</a>

        </div>
    </div>
</div>
```

認証結果とその他の説明情報が表示されるよう、 `result.html` の代わりに、次に示す強調表示のコードを**追加**します。

```
 <div class="col-sm-8">
     <div class="card">
         <div class="card-header">Transaction successful</div>
         <div class="card-body">
             <h4 class="card-title text-success">3D secure authentication was
successful</h4>
             <p class="card-text">You may move onto authorization using the
result below
                  or you can press "Back To Home" to restart authentication
process.</p>

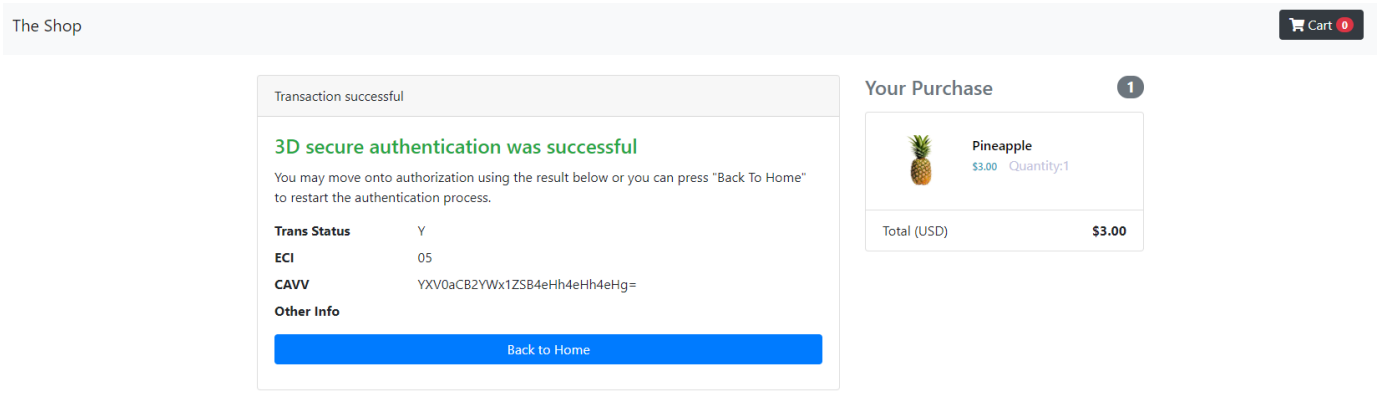
             <dl class="row">
                 <dt class="col-sm-3">Trans Status</dt>
                 <dd class="col-sm-9" data-th-text="${result.transStatus}">Y</
dd>
                 <dt class="col-sm-3">ECI</dt>
                 <dd class="col-sm-9" data-th-text="${result.eci}">eci value</
dd>
                 <dt class="col-sm-3">CAVV</dt>
                 <dd class="col-sm-9" data-th-text="$
{result.authenticationValue}">cavv value</dd>
                 <dt class="col-sm-3">Other Info</dt>
                 <dd class="col-sm-9"></dd>
             </dl>

             <a href="/" class="btn btn-primary">Back To Home</a>
         </div>
     </div>
 </div>
```

Success

お疲れ様でした。ActiveServerが組み込まれた加盟店決済ページの作成作業は、これで完了です。 アプリをもう一度実行してみてください。取引ステータス、ECI、CAVVが含まれた結果ページが正しく表示されるはずです。

新しい結果画面は、次のスクリーンショットのようになります。



> **Tip**
>
> 通常、この処理の後は、取引を完了するための、取引ステータス、ECI、CAVVを使用した信用認証の実行に進みます。

最終的なデモコード

最終的なデモ用コードには、 `/final` ディレクトリにあるGPaymentsのサンプル・コードからアクセスできます。このコードは、追加設定を行わなくてもデフォルト設定のままでActiveServerを使用して動作しますので、操作は簡単で、このガイドに従ってアプリを実行するだけです。

コード解説

3DSリクエスターのデモ用コードには、以下のようにいくつかのエンドポイントとコンポーネントが実装されています。

- `/auth/init` - このエンドポイントは `3ds-web-adapter.js` から呼び出されて認証初期化リクエストを受信し、3DS2認証処理を開始します。このエンドポイントは、3DSリクエスター内でActiveServerの認証初期化APIである **`/api/v1/auth/brw/init/{messageCategory}`** を呼び出します。
- `/auth` - このエンドポイントは `3ds-web-adapter.js` から呼び出されて認証リクエストを受信し、ブラウザー情報の収集後に3DS2認証処理を実行し、オプションのその他の3DSメソッドを実行します。このエンドポイントは、3DSリクエスター内でActiveServerの認証APIである **`/api/v1/auth/brw`** を呼び出します。
- `/3ds-notify` - 3DSメソッドの実行ステータス、認証結果、チャレンジ結果についてActiveServerが `iframe` を通して3DSリクエスターに通知できるようにします。

- `3ds-web-adapter.js` - 3DSリクエスター処理のスキャフォールディング用に作成された簡単なJavaScriptライブラリ。
- `com.gpayments.requestor.dto.activeserver` パッケージ内のクラス - ActiveServer API DTO（データ転送オブジェクト）。

加盟店のwebサイトに組み込まれた3DSリクエスターのデモ用コードは、加盟店サイトが決済ページに組み込むための `3ds-web-adapter` を提供します。加盟店webサイトは、 `3ds-web-adapter.js` で `threeDSAuth()` を呼び出してカード会員情報をセットし、認証処理を開始します。また、加盟店webサイトは、3DSリクエスターが関連するコールバックURLと実行すべき3DSメソッドをセットできるようにするための `iframe` も提供する必要があります。

認証処理が終わると、3DSリクエスターからイベントコールバックURL `/3ds-notify` を使用して認証結果が送信されます。この時点で、決済代行会社または加盟店は認証処理に進むことができます。

ライトボックス・チャレンジ画面

チュートリアルではチャレンジ画面がインラインになっていましたが、webサイトのデザインによってはポップアップ内にチャレンジウィンドウを表示した方が良いかも知れません。 UI自体はACSに用意されていますので、この機能の実装について心配する必要はありません。

`checkout.html` を**開き**、チャレンジウィンドウを定義しているコード部分を見つけてください。

```
 <!--checkout.html-->
<div class="card d-none" id="checkoutCard">
    <div class="challengeContainer border">
        <div class="spinner row h-100 justify-content-center align-items-
center">
            <div class="col">
                <div class="sk-fading-circle">
                    <div class="sk-circle1 sk-circle"></div>
                    <div class="sk-circle2 sk-circle"></div>
                    <div class="sk-circle3 sk-circle"></div>
                    <div class="sk-circle4 sk-circle"></div>
                    <div class="sk-circle5 sk-circle"></div>
                    <div class="sk-circle6 sk-circle"></div>
                    <div class="sk-circle7 sk-circle"></div>
                    <div class="sk-circle8 sk-circle"></div>
                    <div class="sk-circle9 sk-circle"></div>
                    <div class="sk-circle10 sk-circle"></div>
                    <div class="sk-circle11 sk-circle"></div>
                    <div class="sk-circle12 sk-circle"></div>
                </div>
                <div class="text-center"><img class="w-25" src="images/visa-
logo.png"/></div>
            </div>
        </div>
    </div>
</div>
```

行2と3をコメントアウトして、インライン・チャレンジ・カードを無効化します。

```
 <!--checkout.html-->
<!--<div class="card d-none" id="checkoutCard">-->
    <!--<div class="challengeContainer border">-->
        <div class="spinner row h-100 justify-content-center align-items-
center">
            <div class="col">
                <div class="sk-fading-circle">
                    <div class="sk-circle1 sk-circle"></div>
                    <div class="sk-circle2 sk-circle"></div>
                    <div class="sk-circle3 sk-circle"></div>
                    <div class="sk-circle4 sk-circle"></div>
                    <div class="sk-circle5 sk-circle"></div>
                    <div class="sk-circle6 sk-circle"></div>
                    <div class="sk-circle7 sk-circle"></div>
                    <div class="sk-circle8 sk-circle"></div>
                    <div class="sk-circle9 sk-circle"></div>
                    <div class="sk-circle10 sk-circle"></div>
                    <div class="sk-circle11 sk-circle"></div>
                    <div class="sk-circle12 sk-circle"></div>
                </div>
                <div class="text-center"><img class="w-25" src="images/visa-
logo.png"/></div>
            </div>
        </div>
    </div>
</div>
<!--Cardholder information -->
<div class="card" id="cardholderInfoCard">
```

4, 5, 6, 7, 8, 30 と31を追加し、モーダル内にチャレンジウィンドウを作成します。

```
<!--checkout.html-->
<!--<div class="card d-none" id="checkoutCard">-->
<!--<div class="challengeContainer border">-->
<div class="modal fade" id="authBox" data-backdrop="static" data-
keyboard="false" tabindex="-1"
             role="dialog" aria-labelledby="exampleModalLabel" aria-
hidden="true">
    <div class="modal-dialog h-100 d-flex flex-column justify-content-center
my-0" role="document">
        <div class="modal-content">
            <div class="modal-body challengeContainer">
                <div class="spinner row h-100 justify-content-center align-
items-center">
                    <div class="col">
                        <div class="sk-fading-circle">
                            <div class="sk-circle1 sk-circle"></div>
                            <div class="sk-circle2 sk-circle"></div>
                            <div class="sk-circle3 sk-circle"></div>
                            <div class="sk-circle4 sk-circle"></div>
                            <div class="sk-circle5 sk-circle"></div>
                            <div class="sk-circle6 sk-circle"></div>
                            <div class="sk-circle7 sk-circle"></div>
                            <div class="sk-circle8 sk-circle"></div>
                            <div class="sk-circle9 sk-circle"></div>
                            <div class="sk-circle10 sk-circle"></div>
                            <div class="sk-circle11 sk-circle"></div>
                            <div class="sk-circle12 sk-circle"></div>
                        </div>
                        <div class="text-center"><img class="w-25" src="images/
visa-logo.png"/></div>
                    </div>
                </div>
            </div>
        </div>
    </div>
</div>
<!--Cardholder information -->
<div class="card" id="cardholderInfoCard">
```

次に、`checkout.html` の最後まで下にスクロールして `checkout()` 関数の中の次のコードを探します。

```
//checkout.html
function checkout() {
    ....


    //remove cardholder information class, checkout button and show spinner
effect
    $("#checkoutButton").remove();

    $("#cardholderInfoCard").remove();
    $("#checkoutCard").removeClass("d-none");

    threeDSAuth(transId, cardHolderInfo, purchaseInfo);
}
```

カード会員情報クラスを削除しカード内にスピナーを表示するため、**行8と9コメントアウト**します。

```
//checkout.html
function checkout() {
    ....

    //remove cardholder information class, checkout button and show spinner
effect
    $("#checkoutButton").remove();

    //$("#cardholderInfoCard").remove();
    //$("#checkoutCard").removeClass("d-none");

    threeDSAuth(transId, cardHolderInfo, purchaseInfo);
}
```

これに代わるポップアップ・チャレンジウィンドウ用のモーダルボックスを表示するため、**行11を追加**します。

```
//checkout.html
function checkout() {
    ....

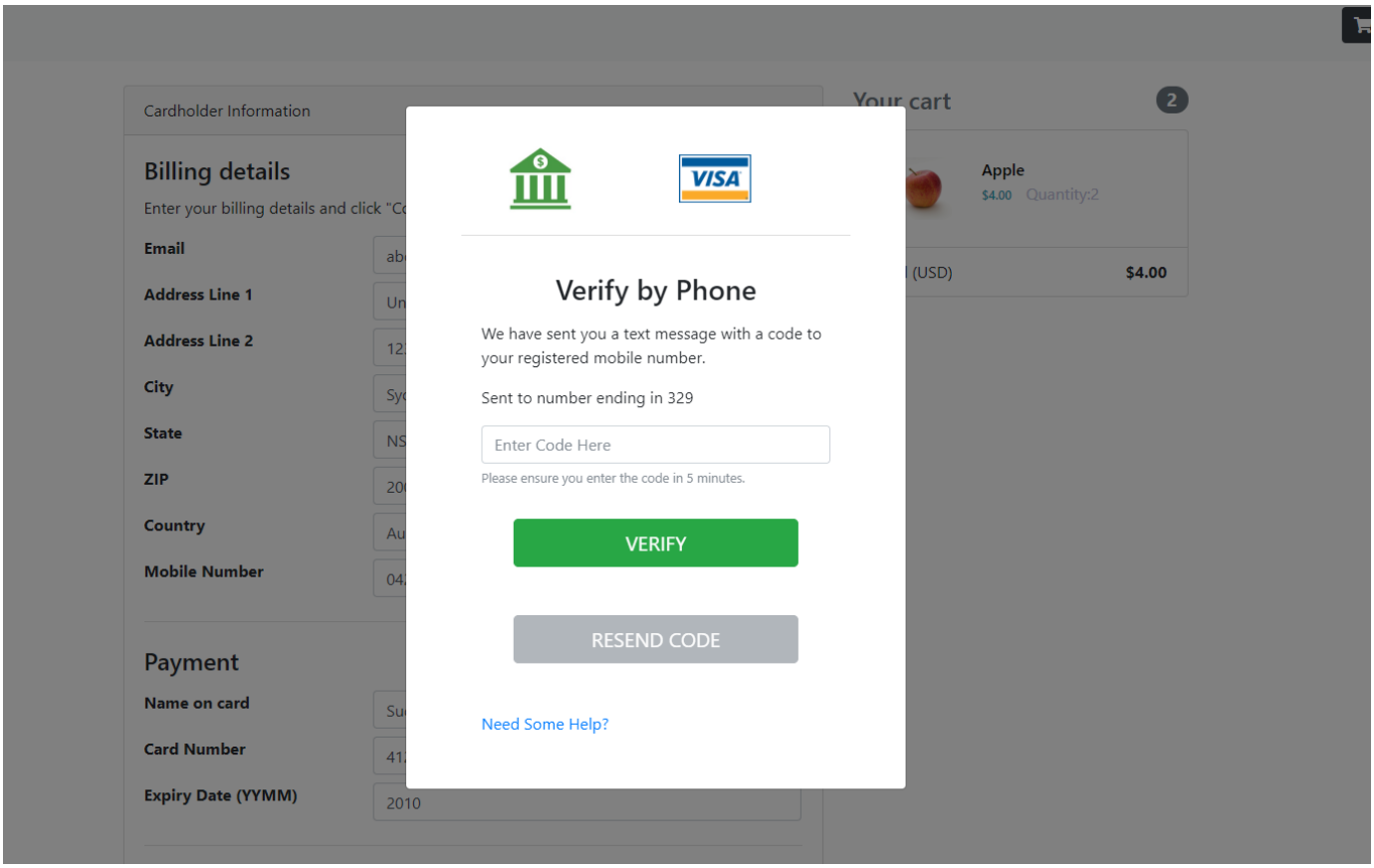
    //remove cardholder information class, checkout button and show spinner
effect
    $("#checkoutButton").remove();

    //$("#cardholderInfoCard").remove();
    //$("#checkoutCard").removeClass("d-none");

    $('#authBox').modal();

    threeDSAuth(transId, cardHolderInfo, purchaseInfo);
}
```

コードを再度実行すると、チャレンジ画面がポップアップウィンドウに変わったことが分かります。



デフォルト設定

最終的なデモ用コードでは、以下の値はデフォルト値に設定されています。この部分は、段階を追ったチュートリアルの実行中に参照できます。

- THREE_DS_SERVER_URL=https://api.as.testlab.3dsecure.cloud:7443
- THREE_DS_REQUESTOR_URL=http://localhost:8082
- KEYSTORE_PASSWORD=123456

> **Note**
>
> - `THREE_DS_SERVER_URL` は、 `Settings > 3D Secure 2` の中にある `AUTH_API_URL` です。詳細は、ここを参照してください。
> - `KEYSTORE_PASSWORD` は、ActiveServer Administrationから証明書ファイル（.p12ファイル）をダウンロードするときに設定したパスワードです。証明書ファイルのダウンロードについては、ここを参照してください。

お問い合わせ

本書の内容についての質問は、弊社担当部門がお受けします。techsupport@gpayments.co.jpまでEメールにてお問い合わせください。

ディレクトリ構想

下記は最終的な加盟店サイトのディレクトリ構想です。ステップ毎の実装ガイドの最後にはこのようなディレクトリ構想になっているはずです。

```
// Finalデモコードのディレクトリ構想
|- com.gpayments.requestor
|   |- config
|   |   |- RestClientConfig.java
|   |   |- JacksonConfig.java
|   |- dto
|   |   |- activeserver
|   |   |    |- AcctInfo.java
|   |   |    |- AuthRequestBRW.java
|   |   |    |- AuthResponseBRW.java
|   |   |    |- CardholderInformation.java
|   |   |    |- InitAuthRequestBRW.java
|   |   |    |- InitAuthResponseBRW.java
|   |   |    |- MerchantRiskIndicator.java
|   |   |    |- PhoneNumber.java
|   |   |    |- ThreeDSRequestorAuthenticationInfo.java
|   |   |
|   |   |- CardholderInfo.java
|   |   |- MyCart.java
|   |   |- Item.java
|   |
|   |- services
|   |   |-CardHolderService.java
|   |   |-CartService.java
|   |   |-ShopService.java
|   |
|   |- transaction
|   |     |- MerchantTransaction.java
|   |     |- TransactionManager.java
|   |
|   |- AuthController.java
|   |- MainController.java
|   |- SampleRequestorApplication.java
|
|- resources
    |-static
    |   |-css
    |   |   |-spinner.css
    |   |   |-style.css
    |   |
    |   |-images
    |   |   |-apple.jpg
    |   |   |-banana.jpg
    |   |   |-pineapple.jpg
    |   |   |-visa-logo.jpg
    |   |
    |   |-js
    |   |   |-3ds-web-adapter.js
    |
    | -certs
```

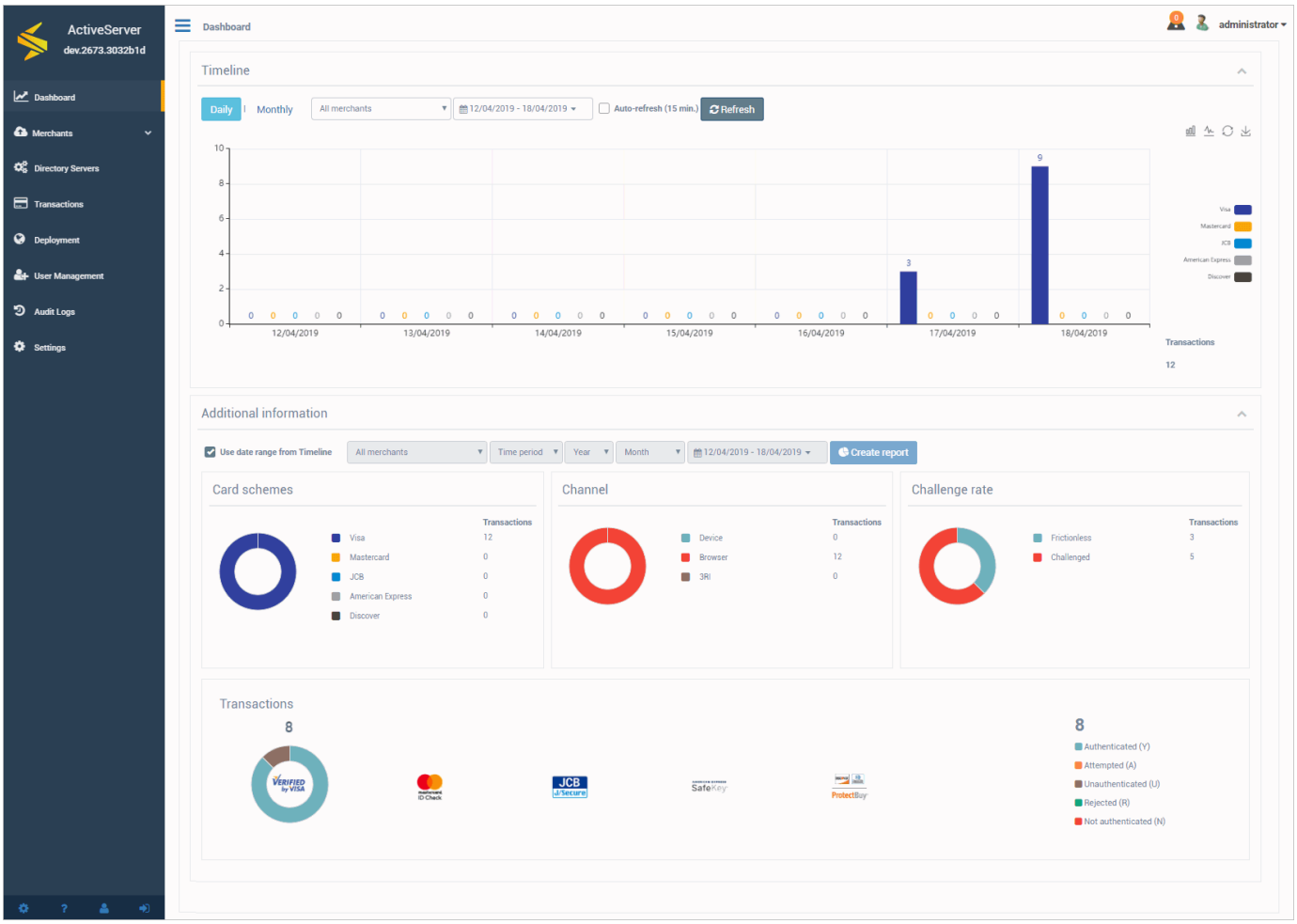
```
|   |- client_certificate.p12
|   |- cacerts.jks
|
|-templates
|   |- checkout.html
|   |- index.html
|   |- notify_3ds_events.html
|   |- result.html
|   |- error.html
|
|-application.properties
```

ダッシュボード使い方

Dashboardには2つのセクションがあります：

- タイムライン
- 追加の情報

> **Tip**
>
> 各ダッシュボード・セクションは、セクションの見出しをクリックすることで折りたたんだり展開したりできます。

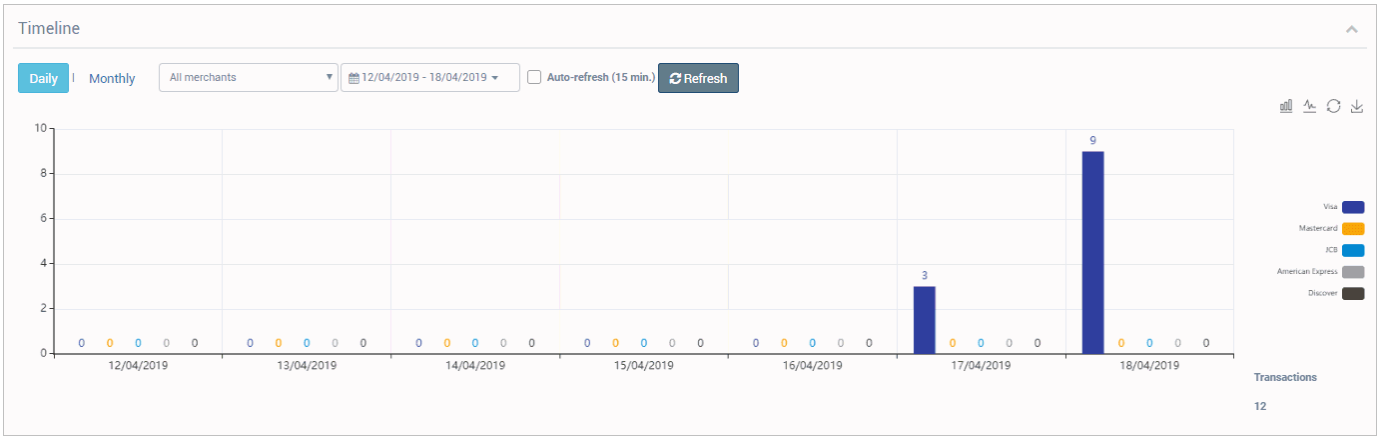


タイムライン

Timelineには、指定された期間中に実行された取引の数の履歴的でグフラフィカルな内訳が表示されており、これは国際ブランド別に分割されています。各国際ブランドビューは、グラフの右側のアイコンをクリックすることでオフにできます。グラフは、折れ線グラフと棒グラフの両方の形式で利用できます。これは、グラフの右上の関連するボタンを使用して切り替えることができます。

インターフェイスは、**Auto-refresh**オプションを選択することで、15分ごとに更新するように設定できます。***Refresh***ボタンは、画面上のデータを最新の収集データに更新するのに使用できます。

データは、**Daily**または**Monthly**ビューで表示できます。これは列の値が何を表しているかを示します。



日毎

Dailyビューを選択すると、グラフの各列は**DD/MM/YYYY**形式で1カレンダー日を表します。**Daily**ビューには、すぐに確認できるように、事前に選択可能な日付範囲が含まれています。

- `Last 7 days` - 当日を含む過去7カレンダー日。
- `Last 30 days` - 当日を含む過去30カレンダー日。
- `Current month` - 当日を含む、現在のカレンダー月のすべての日。
- `Last month` - 前のカレンダー月のすべての日。

日付範囲ピッカーを使用して、**カスタム日付範囲**を選択することもできます。このカスタム日付範囲は、最低1日、最大31日とすることができます。

月毎

Monthlyビューを選択すると、グラフの各列は**MM/YYYY**形式で1カレンダー月を表します。**Monthly**ビューには、すぐに確認できるように、事前に選択可能な日付範囲が含まれています。

- `Current month` - 当日を含む現在のカレンダー月。
- `Current year` - 当日を含む現在のカレンダー年。
- `Last month` - 前のカレンダー月。
- `Last year` - 前のカレンダー年。

日付範囲ピッカーを使用して、**カスタム日付範囲**を選択することもできます。このカスタム日付範囲は、最低1か月、最大12か月とすることができます。

追加の情報

ダッシュボードの**Additional information**セクションでは、**ActiveServer**で実行される各取引の最終ステータスのより詳細な情報が表示されます。この情報は、**Timeline**セクションに表示される情報の補足としたり、システムの過去の日付範囲の取引統計を表示したりするのに使用できます。

期間

Use date range from Timelineを選択すると、日付範囲および加盟店選択オプションがグレーアウトし、**Timeline**セクションで選択されたオプションが使用され、両方のセクションで同期されたビューが表示されます。

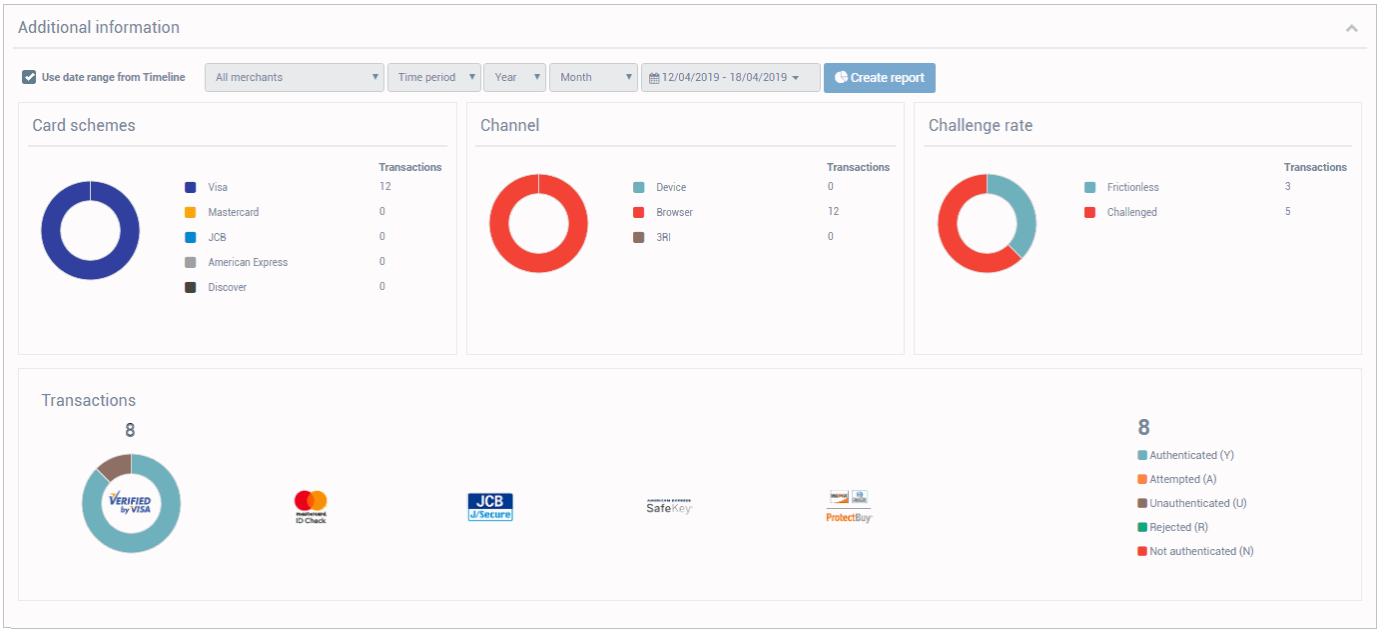
Use date range from Timelineが選択されていない場合、ユーザーは表示する必要がある加盟店統計を指定できます。これは以下の期間で表示されます。

- `Year` - カレンダー年の期間。
- `Month` - 特定の年のカレンダー月の期間。
- `Custom` - カスタム期間の特定の日付範囲。プリセットオプションから選択されます：
  - `Current month` - 当日を含む現在のカレンダー月。
  - `Current year` - 当日を含む現在のカレンダー年。
  - `Last month` - 前のカレンダー月。

- `Last year` - 前のカレンダー年。
- `Custom` - カスタム期間

グラフ

各グラフは、取引のデータの異なるサブセクションを表示します。



国際ブランド

Card schemesセクションでは、`Transaction`列の国際ブランドあたりの合計取引数の数値的な内訳が表示されます。

グラフにポインタを合わせると、国際ブランドあたりの合計取引数の割合的な内訳が表示されます。

チャネル

Channelセクションでは、使用されるチャネル値の合計取引数の数値的な内訳が表示されます。加盟店が取引のほとんどを実行するプラットフォームに関する情報が示されます。 `Browser`エントリは、Webベースのチェックアウトプロセスを使用してユーザーが認証されたことを示します。 `Device`エントリは、チェックアウトプロセス中にネイティブ・モバイルアプリを使用してユーザーが認証されたことを示します。 `3RI`エントリは、加盟店が`3DS Requestor Initiated`認証を実行したことを示します（例： アカウントの有効性の確認や取引の分離など）。

取引の合計数は、**Transactions**列に表示されます。 グラフにポインタを合わせると、使用されるチャネルの割合的な内訳が表示されます。

チャレンジの割合

Challenge rateセクションでは、取引のチャレンジステータスに基づいて、合計取引数の数値的な内訳が表示されます。 `Frictionless` エントリは、リスクベース認証（RBA）を使用してカード会員がイシュアーのACSによって認証でき、ステップアップ・チャレンジが不要だったことを示します。 `Challenged` エントリは、ACSがカード会員に自分自身を認証するようリクエストしたことを示します。 これは、時間経過に対してフリクションレス・フロー機能の進捗状況を監視するのに役立ちます。

取引

Transactionsセクションでは、取引と認証ステータスの数値的な内訳が表示されます。右側の列には、最終ステータスレスポンスに分割された合計取引数が表示されます。個々のグラフには、国際ブランドあたりの取引ステータスの割合的な内訳が表示されます。認証ステータスは以下のように説明されます。 **Transactions**セクションでは、取引と認証ステータスの数値的な内訳が表示されます。 右側の列には、最終ステータスレスポンスに分割された合計認証数が表示されます。 個々のグラフには、国際ブランドあたりの取引ステータスの割合的な内訳が表示されます。認証ステータスは以下のように説明されます。

- **Authenticated (Y)** - 認証確認が成功しました。
- **Attempted (A)** - 試行の処理が実行されました。認証/検証されませんでしたが、認証/検証が試行されたことの証明が提供されます。
- **Unauthenticated (U)** - 認証/アカウント確認を実行できませんでした。技術的またはその他の問題。
- **Rejected (R)** - 認証/アカウント確認が拒否されました。イシュアーは取引/検証を拒否し、オーソリゼーションを試行しないように要求しています。
- **Not Authenticated (N)** - 認証/アカウントが確認されませんでした。取引が拒否されました。

加盟店を検索

加盟店の管理

加盟店が認証APIを介して認証リクエストを行えるようにするには、加盟店エンティティを作成し、3DSリクエスターのクライアント証明書をダウンロードする必要があります。 認証リクエストに含まれていて、頻繁に変更されない詳細は、API機能の単純化のためにデータベースに格納されています。 加盟店エンティティの**作成**、**表示**、**編集**、**削除**プロセスは以下のとおりです。

加盟店の作成

加盟店を作成するには、まず管理インターフェイスの**Merchants**ページに移動し、***New***ボタンを選択します。

New merchant画面で、以下のフィールドを使用し、新しい加盟店を作成します。

> **ユーザーアクセス**
>
> ユーザーが加盟店を作成するには、**Business admin**ロールが必要です。

詳細

Detailsは認証リクエストに使用される一般的な加盟店の詳細です。項目の説明は以下です。

- **Merchant name** - アクワイアラーによって割り当てられた加盟店の名前。これはオーソリゼーションメッセージ・リクエストで使用されるものと同じである必要があります。*最大40文字*
- **Merchant ID** - アクワイアラーによって割り当てられた加盟店の識別子。これはオーソリゼーションメッセージ・リクエストで使用されるものと同じである必要があります。*最大35文字*
- **Country** - 加盟店の運営元の国。認証リクエストの一環として、ActiveServerはこのエントリーを使用しこれを**Merchant Country Code**に変換します。これはオーソリゼーションメッセージ・リクエストで使用される値と一致している必要があります。
- **Default currency** - 認証リクエストで使用されるデフォルト通貨。この値は、`purchaseCurrency`を指定することで、ブラウザベースの初期APIコールで上書きできます。

- **3DS Requestor URL** - 3DSリクエスターWebサイトまたは顧客ケアサイトの完全修飾URL。このデータ要素は、問題が発生した場合に受信した3-Dセキュア・システムに追加情報を提供します。これには、連絡先情報を含める必要があります。
- **Status** - 加盟店が**enabled**（有効）か**disabled**（無効）かを示すステータス。 加盟店を無効にすると、その特定の加盟店に対して認証APIリクエストが許可されなくなります。
- **Notes** - 管理者ユーザーが加盟店のメモにアクセスして編集できるようにするためのセクション。

> **ユーザーアクセス**
>
> ユーザーが**Status**および**Notes**フィールドを表示および編集するには、**Business admin**ロールが必要です。

国際ブランド

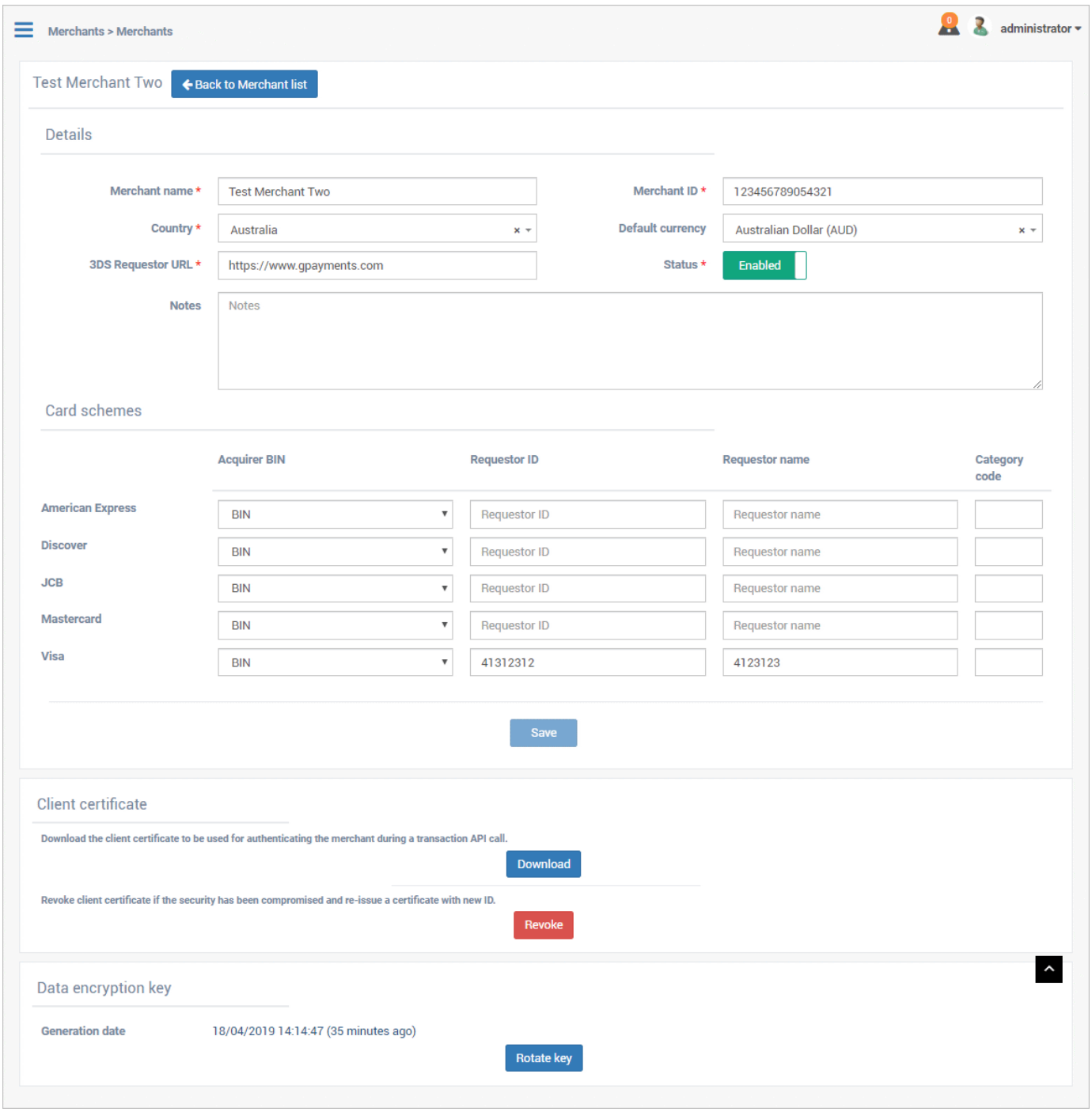
以下は認証リクエストに使用される国際ブランド固有の詳細です。

- **Acquirer BIN** - AReqメッセージを受信するDSによって割り当てられたアクワイアリング機関の識別コード。*最大11文字*
- **Requestor ID** - DSが割り当てた3DSリクエスター識別子。各DSが、3DS2加盟店の登録が完了した後に各3DSリクエスターに個別に一意の識別子を提供します。*最大35文字*
- **Requestor name** - DSが割り当てた3DSリクエスター名。各DSが、3DS2加盟店の登録が完了した後に各3DSリクエスターに個別に一意の名前を提供します。*最大40文字*
- **Category code** - 加盟店の事業、製品、またはサービスの種類を説明するDS固有のコード。*最大4文字*

> **Warning**
>
> 上記のすべての国際ブランド固有の詳細は、認証リクエストの提供に必須です。これらのいずれかが存在しない場合、認証リクエストが失敗します。

加盟店詳細の表示

加盟店詳細を表示するには、管理インターフェイスの**Merchants**ページで加盟店を検索し、加盟店リストから加盟店を選択します。このページでは、加盟店のセキュリティも管理されます。



加盟店のセキュリティ

このページでは、加盟店の**クライアント証明書**および**暗号化の為の鍵**も管理されます。

クライアント証明書

3DSリクエスターのクライアント証明書は、加盟店がSSL認証の認証APIリクエストに含めるために必要です。

- **Download** - ユーザーがパスワードの指定後に**.p12**形式で3DSリクエスターのクライアント証明書をダウンロードすることを許可します。
- **Revoke** - セキュリティ違反または証明書の喪失が発生した場合に現在の3DSリクエスター・クライアント証明書を無効化し、加盟店にダウンロードおよび提供可能な新しい証明書を再発行します。

> **Warning**
>
> クライアント証明書を**失効**させると、過去に発行された証明書は無効化され、代替証明書がインストールされるまで、加盟店はAPIリクエストを送信できなくなります。

CA証明書

- **Download** - 認証APIにリクエストを送信する際に必要なサーバーCA証明書をダウンロードします。この機能の詳細については、APIドキュメント概要を参照してください。

> **バージョン1.0.5**
>
> CA証明書のダウンロードはバージョン1.0.5でリリースされました。

> **ユーザーアクセス**
>
> ユーザーが証明書をダウンロードするには、**Business admin**、

Merchant adminまたは**Merchant**ロールが必要です。
ユーザーが証明書を失効させるには、**Business admin**または**Merchant admin**ロールが必要です。

データ暗号化キー

データベースに保存する前にすべての認証に対して**ActiveServer**がリクエストとレスポンスの暗号化に使用するキーがすべての加盟店に割り当てられています。このキーは、取引の検索時に取引に使用されるアカウント番号を復号化するのにも使用されます。

- **Rotate key** - 内部または外部ポリシーで暗号化キーのローテーションが要求される場合など、必要に応じて、使用中の現在のデータ暗号化キーを変更するのに使用されます。以前の鍵は破壊されるわけではなく以前の取引の暗号化/非暗号化のために使用されます。新しい鍵はローテーション後の取引において使用されます。

ユーザーアクセス

ユーザーがキーをローテーションするには、**Business admin**または**Merchant admin**ロールが必要です。

加盟店詳細の編集

加盟店を編集するには、プロファイルを表示したり、利用可能な項目を編集したりします。

利用可能な加盟店プロフィール詳細は、ユーザーロールに固有です。

- **Status** - **Business admin**ロールのユーザーのみが有効化されたステータスを利用できます。
- **Notes** - **Business admin**ロールのユーザーのみがメモセクションを利用できます。

ユーザーアクセス

加盟店詳細を表示するには、**Business admin**、**Merchant admin**または**Merchant**ロールが必要です。
ユーザーが加盟店詳細を編集するには、**Business admin**または**Merchant admin**ロールが必要です。

加盟店の削除

加盟店を削除するには、まず管理インターフェイスの**Merchants**ページに移動し、加盟店を検索して、検索結果の表の加盟店名に隣接する**deleteチェックボックス**を選択します。***Delete***ボタンを選択し、ダイアログボックスで確認します。

ユーザーアクセス

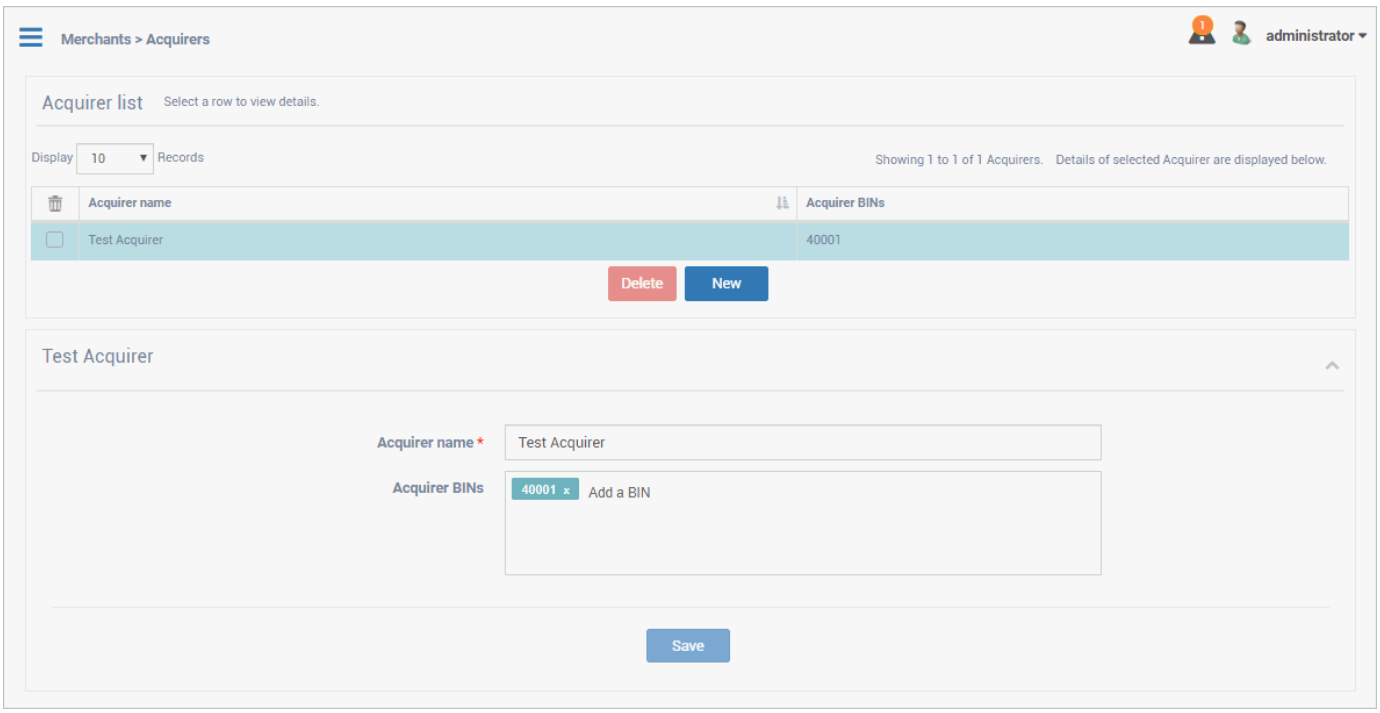
ユーザーが加盟店を削除するには、**Business admin**ロールが必要です。

アクワイアラの管理

すべてのアクワイアラーのリストは、管理インターフェイスの**Merchants > Acquirers**メニューからアクセスできます。 リストには**アクワイアラーの名前**と関連するすべての**BIN番号**が表示されます。

ユーザーアクセス

ユーザーがアクワイアラーを作成、表示、編集、削除するには、**Business admin**ロールが必要です。



アクワイアラーの作成

アクワイアラーを**作成**するには、***New***ボタンを選択し、フィールドに入力します。

- **Acquirer name** - ActiveServerでアクワイアラーを識別するのに使用される名前。認証メッセージには使用されません。
- **Acquirer BINs** - アクワイアラーに割り当てることができるBIN。1つ以上指定可能。このフィールドは認証メッセージで送信され、決済システム、DSへ加盟店が登録時に使用されたものと同じである必要があります。

Saveボタンを選択すると、アクワイアラーが作成されます。

アクワイアラー詳細の表示と編集

アクワイアラー詳細を表示および編集するには、リストからアクワイアラーを選択します。必要に応じてアクワイアラー詳細を変更し、***Save***ボタンを選択します。

アクワイアラーの削除

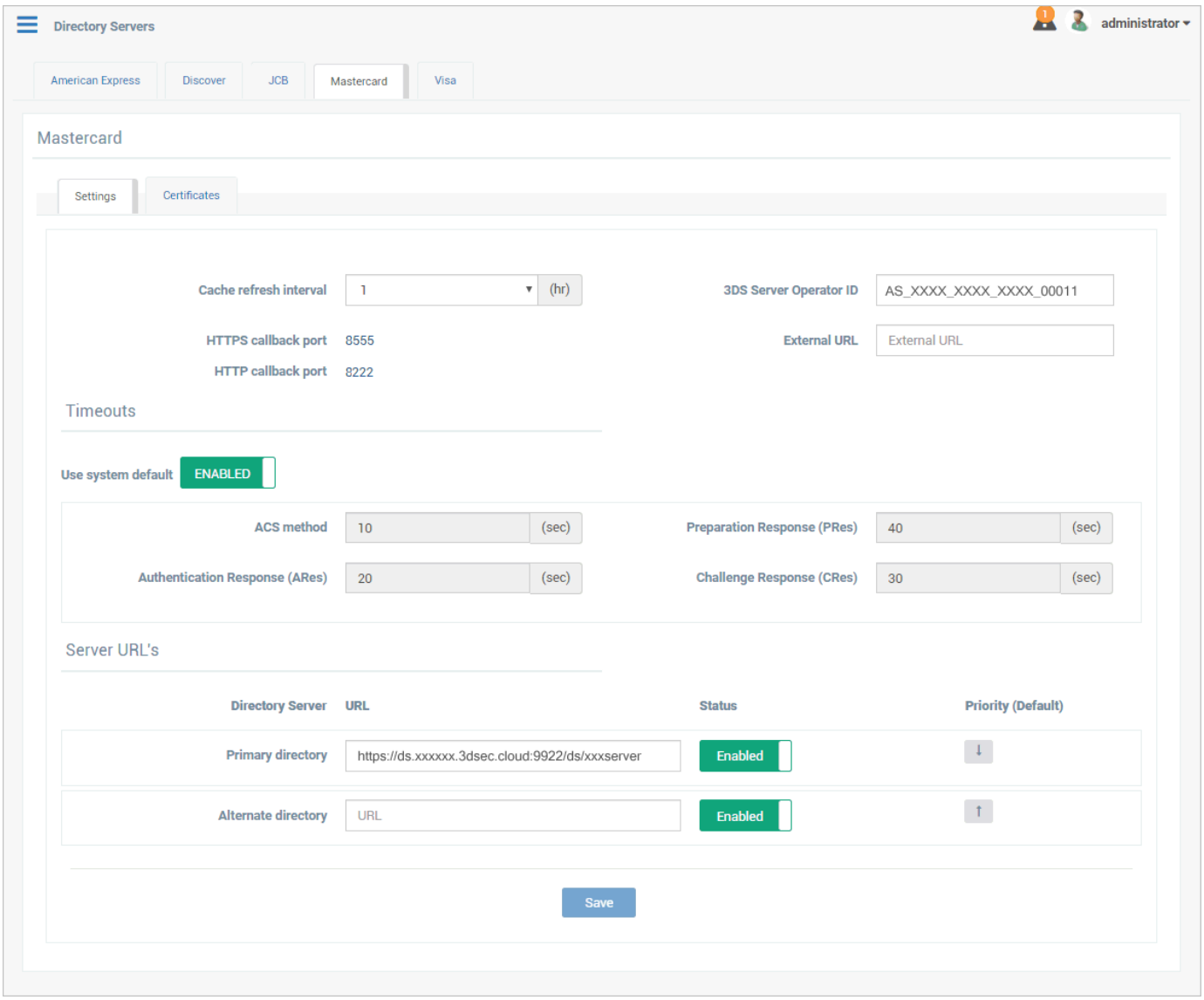
アクワイアラーを削除するには、アクワイアラーのリストの隣にある**delete**チェックボックスを選択します。***Delete***ボタンを選択し、ダイアログボックスで確認します。

DS設定の管理

ActiveServerでサポートされるすべての国際ブランドは**Settings**タブの**Directory Servers**ページで管理できます。

ユーザーアクセス

ユーザーがDS設定を管理するには、**System admin**ロールが必要です。



国際ブランド設定を編集するには：

詳細を表示するには、ページ上部の適切な**国際ブランド**を選択します。

設定

Settingsセクションでは以下の設定を編集できます：

- **Cache refresh interval** - 3DSサーバーは、最低24時間に1回、最大1時間に1回、各登録済みDSの呼び出しを実行して、キャッシュをリフレッシュする必要があります。キャッシュには、利用可能なACSでサポートされるプロトコルバージョン番号、DS、および3DSメソッド・コールに使用される任意のURLに関する情報が含まれます。（単位： 時間）
- **3DS Server Operator ID** - DSが割り当てた3DSサーバー識別子。各DSは各3DSサーバーに個別に一意のIDを提供できます。これは通常、国際ブランド・コンプライアンス・プロセス中、あるいは終了時に提供されます。このフィールドがAReqおよびPReqメッセージに存在する要件はDS固有です。
- **HTTPS callback port** - ASが認証中にHTTPS通信をリスンするポート。
- **External URL** - DSが認証中にコールバックするURL。

タイムアウト

Timeoutsセクションでは以下の設定を編集できます：

- **Use system default** - デフォルト3DS2メッセージ・タイムアウト設定の使用の**Enables**（有効化）と**Disables**（無効化）を切り替えます。
- **Enabled** - 3Dセキュア 2設定で定義されているタイムアウト設定を使用します。
- **Disabled** - ユーザーがDSごとに個別にタイムアウト設定を編集できます。
    - **ACS Method**
    - **Preparation response (PRes)**
    - **Authentication response (ARes)**
    - **Challenge response (CRes)**

サーバーURLリスト

Server URLsセクションでは、国際ブランドDSの本番環境のアドレスのDS URLを入力できます。 初回試行では**Primary directory**が使用されます。初回試行で接続が成功しなかった場合、**ActiveServer**は**Alternate directory** URLを使用して再試行します。

Statusボタンを使用すると、DSへのすべての接続にDS URLを使用するかどうかを切り替えることができます。**Enabled**は接続を許可し、**Disabled**は接続試行でURLを無視します。

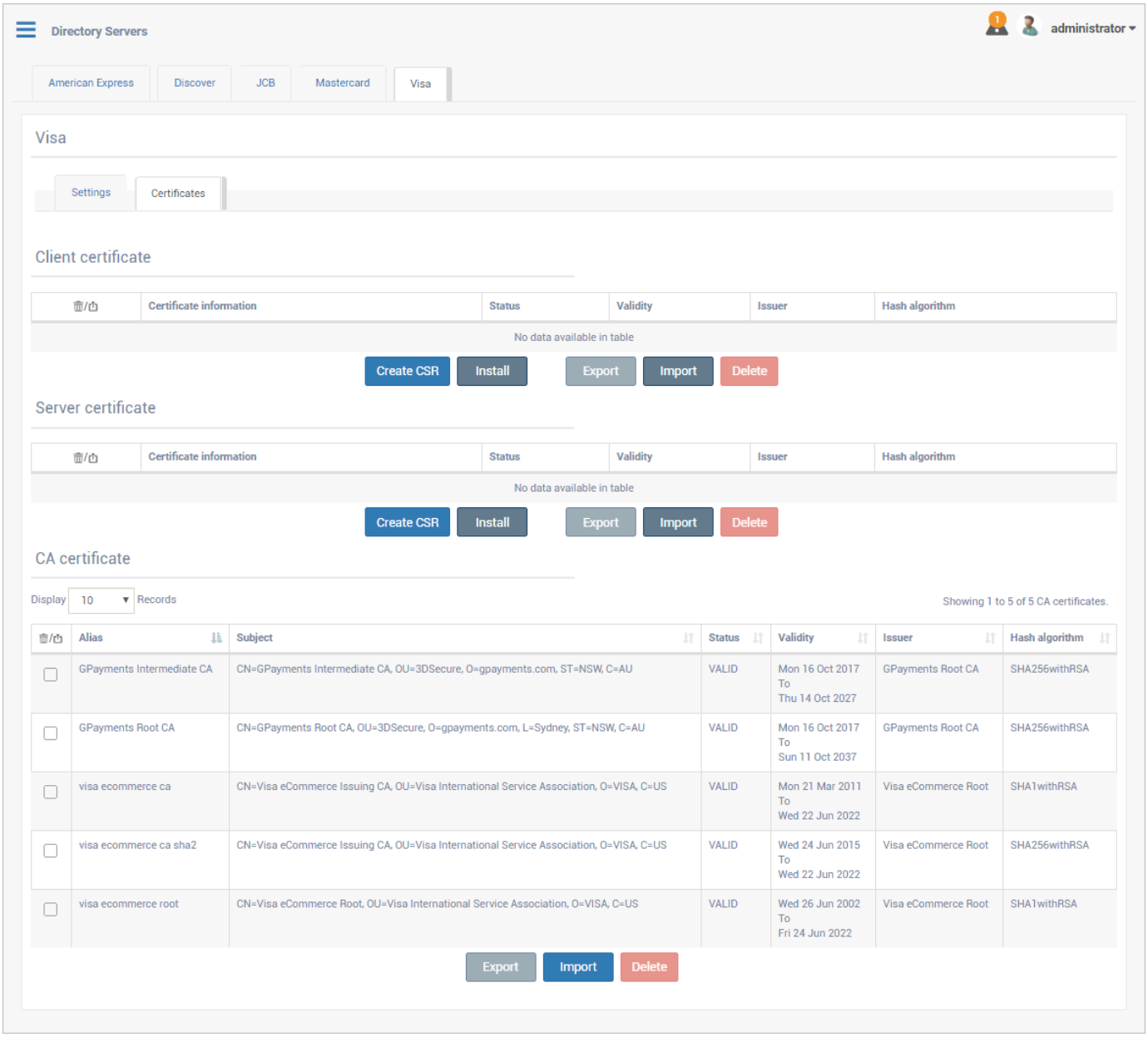
Priority (Default)矢印を使用すると、**Primary directory**と**Alternate directory**の間でDS URLを移動させることができ、URLプロパティを切り替えます。

DS証明書の管理

ActiveServerでサポートされるすべての国際ブランドは、**Certificates**タブの**Directory Servers**ページで管理できます。

ユーザーアクセス

ユーザーがDS証明書を管理するには、**System admin**ロールが必要です。



国際ブランド証明書を管理するには：

詳細を表示するには、ページ上部の適切な**国際ブランド**タブを選択します。

クライアント証明書

証明書は、通常、証明書署名リクエスト（CSR）を提供した後に国際ブランドからダウンロードできます。

以下のプロセスを実行できます：

CSRの作成

CSRの生成を支援するため、**ActiveServer**は***Create CSR***ボタンからこの機能を提供しています。ただし、ご希望であれば、*Java keytool*のような別の方法を使用して、手動でこのプロセスを実行することもできます。

証明書の内容は、国際ブランドの要件に応じて入力する必要があります。以下のオプションが利用可能です。

- **Key size** - リクエストのキーのサイズ（ビット単位）
- **Common Name** – 証明書に使用されるホスト名。通常、完全修飾ドメイン名が使用されます。
- **Organization** – 企業または組織の法的な名前
- **Organization Unit** – グループの部署または部門の名前
- **City** – 企業がある市区町村
- **Province** – 企業がある都道府県
- **Two letter country code** – 国の2文字の略称
- **Hash algorithm** - CSRの署名に使用されるハッシュアルゴリズム

CSRの作成では、生の証明書コンテンツが作成され、`.p10`形式で***Download certificate***のボタンが提供されます。

インストール

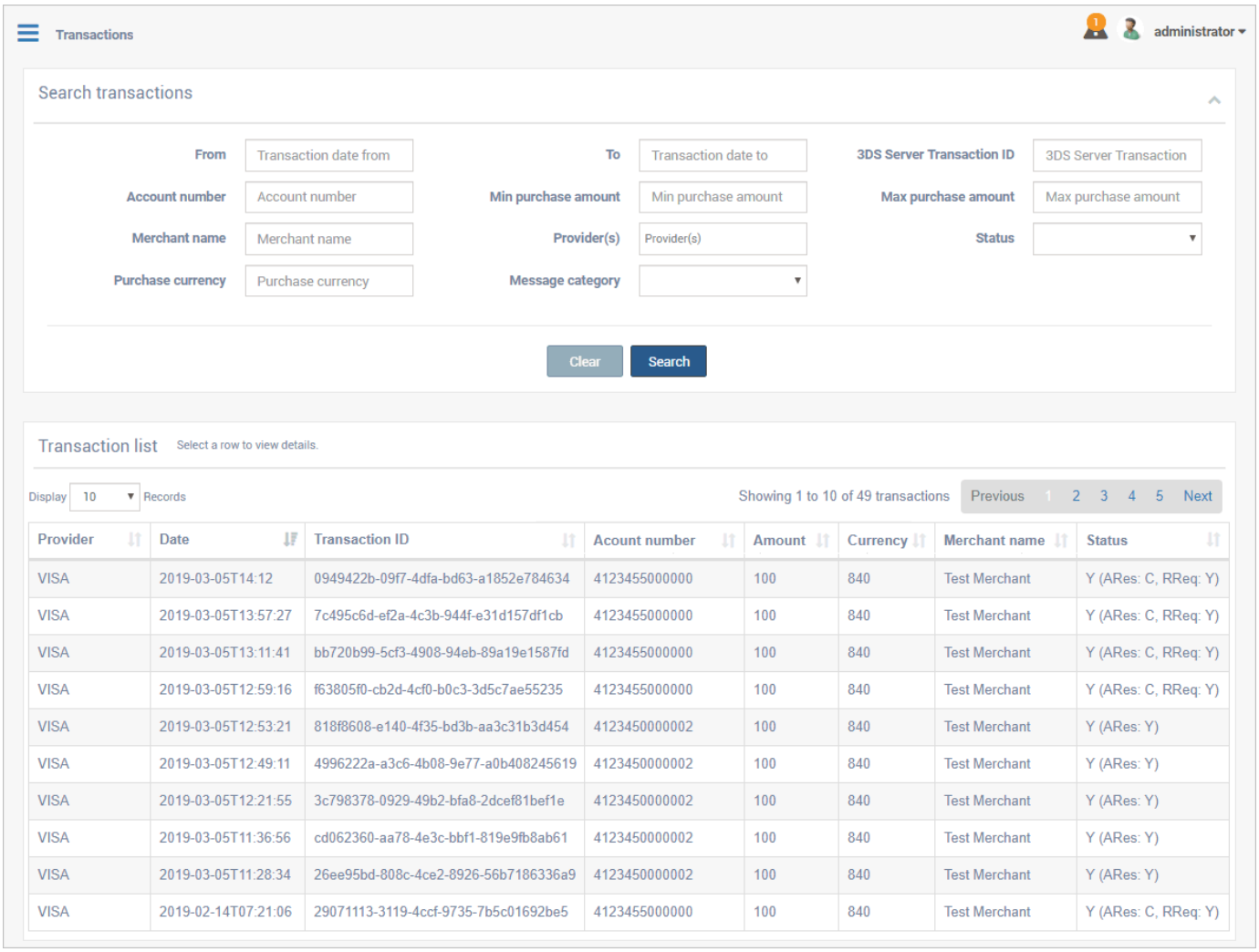
署名済みの証明書は、***Install***ボタンを使用することでインストールできます。

> **Warning**
>
> 一度に**1つ**のクライアント証明書のみを持つことができ、別の証明書を**インストール**または**インポート**すると、現在の証明書が上書きされます。

取引を見る

Transactionsは、左側のメニューからアクセスでき、3つのセクションがあります。

- 取引を検索
- 取引のリスト
- 取引の詳細



取引を検索

このセクションでは、特定の取引について、データベースから検索できます。取引に関する情報を入力することで、取引をフィルタリングできます。フィールドには以下が含まれます：

- **From** - 指定した日付以降の取引のみが含まれます

- **To** - 指定した日付以前の取引のみが含まれます
- **3DS Server Transaction ID** - 指定した取引IDの取引のみが含まれます
- **Account number** - カード会員アカウント番号による取引のみが含まれます（PANまたはトークンによって表される場合があります）
- **Minimum purchase amount** - 指定した金額を超える取引のみが含まれます
- **Maximum purchase amount** - 指定した金額未満の取引のみが含まれます
- **Merchant name** - 指定した加盟店によって処理された取引のみが含まれます。加盟店名のすべてまたは一部を使用できます。
- **Provider(s)** - 指定した１つまたは複数の国際ブランドによって処理された取引のみが含まれます
- **Status** - 指定した結果ステータスの取引のみが含まれます（例： `Transaction Successful` の場合は"Y"など）
- **Purchase currency** - 指定した通貨で実行された取引のみが含まれます。これは通貨コードで定義されます
- **Message category** - PA（決済）またはNPA（非決済）のいずれかの取引のみが含まれます

目的のフィルタを設定し、**Search**をクリックすると、**Transaction List**の下に結果が表示されます。**Clear**をクリックすると、フィールドがリセットされます。

取引のリスト

取引リストには、すべての取引、または上記の検索パラメータのいずれかを選択した場合はフィルタリングされたリストが表示されます。

表示される取引の詳細は以下のとおりです：

- **Provider** - 取引に使用された国際ブランド
- **Date** - 取引が処理された日時
- **3DS Server Transaction ID** - 特定の取引の3DSサーバー取引ID
- **Account number** - 取引に使用されたアカウント番号。PANまたはトークンで表される場合があります。
- **Amount** - 取引の購入金額
- **Currency** - 取引が処理された通貨コード

- **Merchant name** - 取引を処理している加盟店の名前
- **Status** - 取引に対して返された認証結果。以下のフォーマットが使用されます：
  - **Frictionless (i.e. no challenge)** - "*最終ステータス* (ARes: *AResステータス*)"、例： "Y (ARes: Y)"
  - **Challenge** - "*最終ステータス* (ARes: *AResステータス*", RReq: *RReqステータス*)、例： "N (ARes: C, RReq: N)"

取引行を選択すると、Transaction detailsセクションの下に詳細が表示されます。

取引の詳細

取引詳細には、取引リストで選択した取引の詳細が表示されます。

表示される取引の詳細は以下のとおりです：

- **3DS Server Transaction ID** - 単一の取引を特定するために3DSサーバーによって割り当てられた、世界で1つだけの取引識別子です
- **Purchase date** - 購入の日時

- **Merchant ID** - アクワイアラーが割り当てた加盟店識別子。これは、3DSリクエスターに代わって送信されるオーソリゼーションリクエストで使用されるものと同じ値である必要があります
- **Merchant name** - アクワイアラーまたは決済システムによって割り当てられた加盟店名
- **Account number** - 決済取引のオーソリゼーションリクエストに使用されるアカウント番号。PANまたはトークンで表される場合があります
- **Amount** - 取引の購入金額
- **ThreeDS Requestor name** - DSが割り当てた3DSリクエスター名。各DSが、各3DSリクエスターに個別に一意の名前を提供します。
- **ThreeDS Requestor ID** - DSが割り当てた3DSリクエスター識別子。各DSが、各3DSリクエスターに個別に一意の識別子を提供します。
- **ARes/CRes/RReq status** - 取引が認証済みの取引またはアカウント確認のどちらに該当するかを示します。考えられる値は以下のとおりです：
  - **Authenticated (Y)** - 認証確認が成功しました
  - **Not Authenticated (N)** - 認証/アカウントが確認されませんでした。取引が拒否されました
  - **Unauthenticated (U)** - 認証/アカウント確認を実行できませんでした。技術的またはその他の問題
  - **Attempted (A)** - 試行の処理が実行されました。認証/検証されませんでしたが、認証/検証が試行されたことの証明が提供されます
  - **Challenge Required** - CReq/CResを使用した追加認証が必要です
  - **Rejected (R)** - 認証/アカウント確認が拒否されました。イシュアーは取引/検証を拒否し、オーソリゼーションを試行しないように要求しています
- **Status reason** - Transaction Statusフィールドが指定された値を持つ理由に関する情報を提供します。決済チャネルについては、Transaction StatusフィールドがN、U、またはRの場合に必須です。非決済チャネルについては、DSで定義されている場合に条件付き必須です。可能な値は以下のとおりです：
  - **01** - カード認証が失敗しました
  - **02** - 不明なデバイス
  - **03** - サポートされていないデバイス
  - **04** - 認証頻度制限を超えました
  - **05** - 期限切れのカード

- **06** - カード番号が無効です
- **07** - 取引が無効です
- **08** - カードレコードがありません
- **09** - セキュリティ障害
- **10** - 盗難されたカード
- **11** - 詐欺の疑い
- **12** - カード会員に取引が許可されていません
- **13** - カード会員がサービスに登録されていません
- **14** - ACSで取引がタイムアウトしました
- **15** - 信用：低
- **16** - 信用：中
- **17** - 信用：高
- **18** - 信用：最高
- **19** - ACS最大チャレンジ数を超えました
- **20** - 非決済取引はサポートされていません
- **21** - 3RI取引はサポートされていません
- **22–79** - EMVCoが今後使用するため予約されています（EMVCoによって定義されるまで値は無効です）
- **80–99** - DSが使用するため予約されています

- **Acquirer BIN** - アクワイアラーのアクワイアリング機関の識別コード
- **Device channel** - 取引の発生元のチャネルを示します。可能な値は以下のとおりです：
  - App-based (01-APP)
  - Browser-based (02-BRW)
  - 3DS Requestor Initiated (03-3RI)
- **Pay token indicator** - 値**True**は、取引がACSによって受信される前にトークン化解除されたことを示します。このデータ要素は、トークン化解除が発生した3-Dセキュアドメインに存在するシステム（すなわち3DSサーバーまたはDS）によって入力されます。存在する場合、このフィールドに対して有効なレスポンスはブール値trueのみです。

- **Authentication value** - サポートされている各DSのACS登録の一環として提供される決済システム固有の値。Authentication Valueは、オーソリゼーション中やクリアリング中など、認証の証明を提供するのにも使用できます。
- **Authentication type** - チャレンジが必要な場合にイシュアーがカード会員のチャレンジに使用した認証方法のタイプを示します。AResメッセージ内にある場合もありますし、RReqメッセージ内ではACSによって使用されたものを示します。可能な値は以下のとおりです：
  - **01** - 静的
  - **02** - 動的
  - **03** - OOB
  - **04–79** - EMVCoが今後使用するため予約されています（EMVCoによって定義されるまで値は無効です）
  - **80–99** - DSが使用するため予約されています
- **Merchant category code** - 加盟店の事業、製品、またはサービスの種類を説明するDS固有のコード
- **Message category** - 取引のカテゴリを識別します。可能な値は以下のとおりです：
  - **01** - PA（決済）
  - **02** - NPA（非決済）
  - **03–79** - EMVCoが今後使用するため予約されています（EMVCoによって定義されるまで値は無効です）
  - **80–99** - DSが使用するため予約されています
- **Card expiry date** - カード会員によって3DSリクエスターに提供されたPANまたはトークンの有効期限（**YYMM**形式）
- **Recurring frequency** - 定期取引の場合、オーソリゼーション間の最低日数を示します
- **Purchase currency** - 購入金額を表現する通貨コード
- **Recurring expiry date** - 定期取引の場合、それ以降オーソリゼーションが実行されない日付（**YYYYMMDD**形式）
- **Challenge mandated** - 局所的/地域的な義務またはその他の不確定要素によって、取引のオーソリゼーションにチャレンジが必要かどうかを示します。可能な値は以下のとおりです：
  - **Y** - チャレンジが必須
  - **N** - チャレンジが必須ではない

- **SDK transaction ID** - 単一の取引を特定するために3DS SDKによって割り当てられた、世界で1つだけの取引識別子です
- **DS transaction ID** - 単一の取引を特定するためにDSによって割り当てられた、世界で1つだけの取引識別子です
- **ACS transaction ID** - 単一の取引を特定するためにACSによって割り当てられた、世界で1つだけの取引識別子です
- **ThreeDS requestor challenge indicator** - この取引について加盟店によってチャレンジがリクエストされているかどうかを示します。例：
  - 01-PAの場合、3DSリクエスターは取引に懸念を持ち、チャレンジを要求できます。
  - 02-NPAの場合、新しいカードをウォレットに追加するときにチャレンジが必要な場合があります。
  - 局所的/地域的な義務またはその他の不確定要素。
  - 可能な値は以下のとおりです：
    - **01** - 指定なし
    - **02** - チャレンジリクエストなし
    - **03** - チャレンジがリクエストされました：3DSリクエスター指定
    - **04** - チャレンジがリクエストされました：義務
    - **05–79** - EMVCoが今後使用するため予約されています（EMVCoによって定義されるまで値は無効です）
    - **80–99** - DSが使用するため予約されています
    - 注：要素が指定されない場合、想定されるアクションは、ACSが01 =指定なしと解釈することです。
- **Error code** - 利用可能な場合、メッセージで識別された問題のタイプを示すコード。
- **Error detail** - 利用可能な場合、メッセージで識別された問題に関する追加の詳細。
- **Error component** - 利用可能な場合、エラーを識別した3-Dセキュア・コンポーネントを示すコード。可能な値は以下のとおりです：
  - **C** - 3DS SDK
  - **S** - 3DSサーバー
  - **D** - DS
  - **A** - ACS
- **Error description** - 利用可能な場合、メッセージで識別された問題を説明するテキスト。

- **Message version** - 3DSサーバーによって使用されるプロトコル・バージョン識別子。AReq メッセージに設定されます。メッセージバージョン番号は、3DS取引中に変更されません。

取引メッセージ

取引の3DS2メッセージを表示するには、Transactionsセクションの下部の `Requests / Responses` をクリックします。

Message type AReq

Time stamp 2019-04-18T01:00:38

Message content


```
{
  "threeDSCompInd": "U",
  "threeDSRequestorAuthenticationInd": "01",
  "threeDSRequestorAuthenticationInfo": {
    "threeDSReqAuthMethod": "02",
    "threeDSReqAuthTimestamp": "201711071307",
    "threeDSReqAuthData": "login GP"
  },
  "threeDSRequestorChallengeInd": "01",
  "threeDSRequestorID": "123456789.visa",
  "threeDSRequestorName": "3dsclient.local.visa",
  "threeDSRequestorURL": "http://gpayments.com",
  "threeDSServerRefNumber": "3DS_LOA_SER_GPPL_020100_00075",
  "threeDSServerOperatorID": "AS_TEST_LAB_OPER_00001",
  "threeDSServerTransID": "4aa7fd8e-436e-404c-81bd-d0b0bedf1a14",
  "threeDSServerURL": "https://api.asuat.testlab.3dsecure.cloud:9454/api/v1/auth/result/request",
  "acctType": "03",
  "acquirerBIN": "40001",
  "acquirerMerchantID": "123456789012345",
  "addrMatch": "N",
```


Message type ARes

Time stamp 2019-04-18T01:00:39

Message content


```
{
  "threeDSServerTransID": "4aa7fd8e-436e-404c-81bd-d0b0bedf1a14",
  "acsOperatorID": "1234567890",
  "acsReferenceNumber": "12345678900",
  "acsTransID": "7a7d97fc-18d4-4929-aa00-9cc4536deb38",
  "authenticationValue": "AAABBCABEhGQAAAAAAESAAAAAAA=",
  "dsReferenceNumber": "GP_TEST_LAB_VISA_001",
  "dsTransID": "37986e5c-9a86-4506-82b6-bf933a810027",
  "eci": "05",
  "messageVersion": "2.1.0",
  "messageType": "ARes",
  "transStatus": "Y"
}
```


- **Message type** - AReq、AResなど、3DSメッセージタイプ。
- **Time stamp** - メッセージが送受信された日時。
- **Message content** - 送受信された生のJSONメッセージコンテンツ。

ノード管理

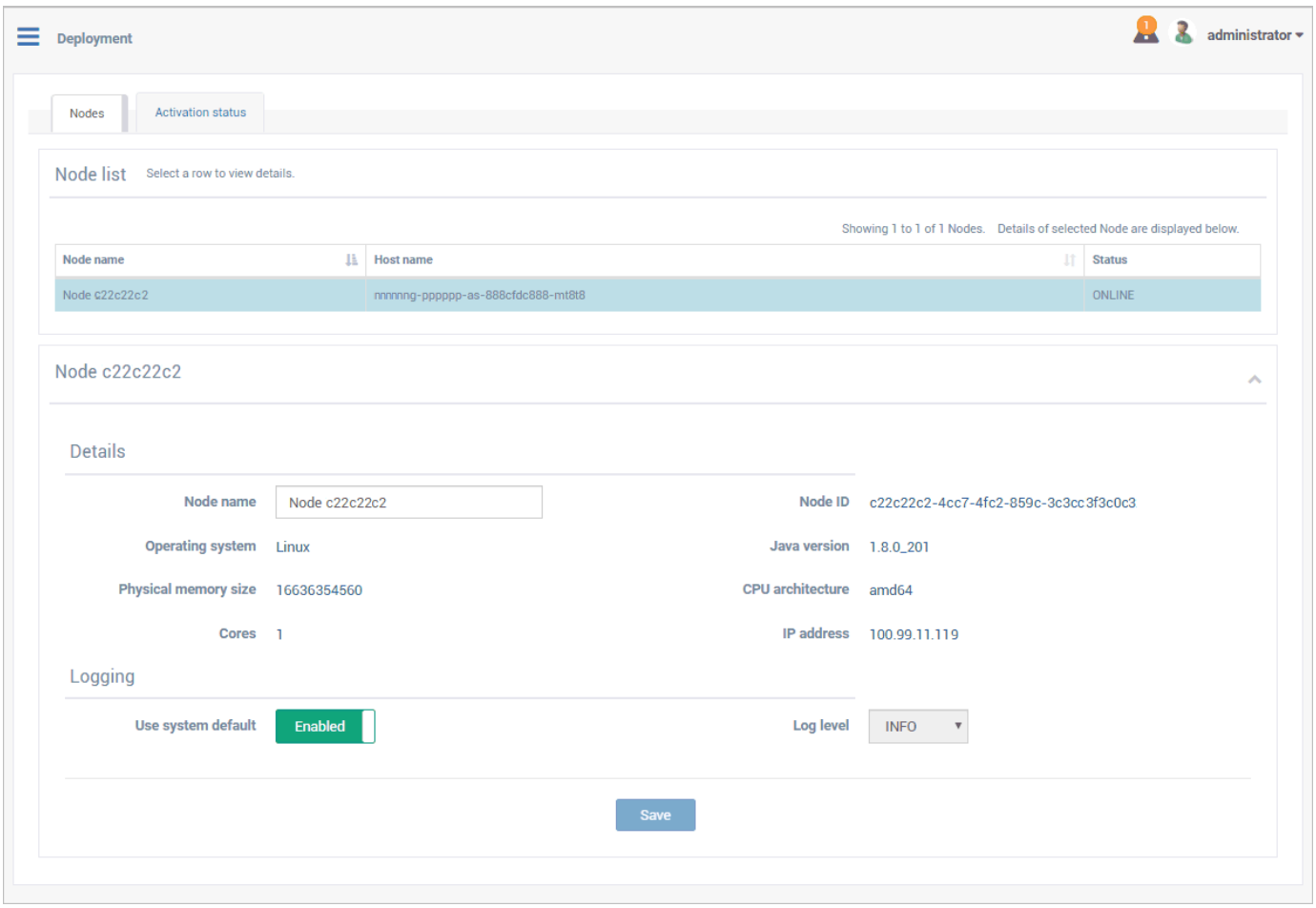
ActiveServerは、**単一ノード**または**複数ノード**セットアップのどちらでも実行できます。 これらのノードの詳細は、**Deployment**ページの**Nodes**タブで確認できます。 このページは、システム全体のログレベル設定とは異なる設定が必要な場合に、特定のノードのログレベルを設定できる場所でもあります。

Info

複数ノード・セットアップのガイドは、今後のドキュメントバージョンで提供されます。

ユーザーアクセス

ユーザーがノード設定を表示および編集するには、**System admin**ロールが必要です。



ノード詳細

詳細を表示するには、**Node list**からノードを選択します。

- **Node name** - ステータスの追跡を維持するためにノードに指定できる編集可能な名前。
- **Node ID** - データベース内のノードのシステム生成UUID。
- **Operating system** - ノードのサーバーに使用されているオペレーティング・システム。
- **Java version** - ノードのサーバーに使用されている Javaバージョン。
- **Physical memory size** - ノードのサーバーに搭載されている物理メモリの合計量。
- **CPU architecture** - ノードのサーバーに搭載されているCPUアーキテクチャ・チップのタイプ
- **Cores** - ノードのサーバーに搭載されている論理コア数。
- **IP address** - ノードがホストされているIPアドレス。

ログ

特定のノードに異なるログレベルが必要な場合は、以下の設定を変更できます。

- **Use system default** - *Enabled*の場合、ノードはシステム全体のログレベル設定を使用します。*Disabled*の場合、ノードのログ・ベルを手動で変更できます。
- **Log level** - **Use system default**が*Disabled*の場合に編集でき、ログレベルを変更できます。

ユーザーの管理

ユーザーは、管理インターフェイスへのアクセス権を提供するために作成されます。 ユーザー**ロール**は、特定の業務上の責任に関するタスクを完了するため、適切なアクセス権を提供するためにユーザーに割り当てられます。 **ロール**の詳細については、ロールと権限ガイドを参照してください。

ユーザーの作成、表示、編集、削除プロセスは以下のとおりです。

ユーザーの作成

ユーザーを作成するには、まず管理インターフェイスの**User Management**ページに移動し、***New***ボタンを選択します。

以下に記載のフィールドを使用して、**New user**画面のフィールドを入力します。

> **Important**
>
> 以下の**Role/s**および**Merchant**フィールドは、システムへの適切なアクセス権を割り当てるために重要です。
> 新しいユーザーを作成する前に、ロールと権限ガイドを確認することをお勧めします。

- **Username** - 管理インターフェイスへのログインに使用される、ユーザーに割り当てられた一意の値。*必須*
- **First name** - ユーザーの名前。*必須*
- **Last name** - ユーザーの名字。*必須*
- **Email** - ユーザーの有効な電子メールアドレス。このアドレスは、パスワードリセットやシステム通知などの電子メール通知を送信するのに使用されます。*必須*
- **Roles** - ユーザーに割り当てることができる、ロールの複数の選択ボックス。*必須*
- **Merchant** - 割り当てられたユーザーロールによって、単一の加盟店の**スコープ**がユーザーに付与された場合、すでに管理用に作成された加盟店をユーザーに割り当てることができます。割り当てられたユーザーロールによって、すべての加盟店のスコープがユーザーに付与された場合や、加盟店なしのスコープが付与された場合、このフィールドは入力する必要がなく、選択できません。*必須*
- **Time zone** - 管理インターフェイス上の日時の表示のデフォルト・タイムゾーン。*必須*

- **Status** - ユーザーのシステムステータス：
  - **Enabled** - ユーザーは管理インターフェイスの機能にアクセスできます。
  - **Disabled** - ユーザーは管理インターフェイスの機能にアクセスできません。

> **ユーザーアクセス**
>
> ユーザーがユーザーを作成するには、**User admin**ロールが必要です。

ユーザー詳細の表示

ユーザーの詳細は、管理インターフェイスの**User Management**ページからユーザーのリストから選択することでアクセスできます。

表示されるユーザー数は、以下のフィールドを使用してフィルタリングすることで制限できます。

- **Username** - ユーザー名の全体または一部。
- **Role(s)** - 単一システム・ロールのドロップダウン選択。

結果の表には、上記の**Username**、**First name**、**Last name**、**Roles**、**Status**が表示されます。

> **ユーザーアクセス**
>
> ユーザーがユーザー詳細を表示するには、**User admin**ロールが必要です。

ユーザー詳細の編集

ユーザーを編集するには、プロファイルを表示したり、利用可能なフィールドを編集したりします。

> **Important**
>
> 常に**User Admin**ロールのユーザーがシステムに1人は存在する必要があります。ユーザー詳細の編集によってこのチェックが失敗すると、エラーが発生します。

ユーザーアクセス

ユーザーがユーザー詳細を編集するには、**User admin**ロールが必要です。

ユーザーの削除

ユーザーを削除するには、まず管理インターフェイスの**User Management**ページに移動します。リストをフィルタリングしてユーザーを探し、検索結果の表でユーザー名に隣接する**delete チェックボックス**を選択します。***Delete***ボタンを選択し、ダイアログボックスで確認します。

Important

常に**User Admin**ロールのユーザーがシステムに少なくとも1人は存在する必要があります。ユーザーの削除によってこのチェックが失敗すると、エラーが発生します。

ユーザーアクセス

ユーザーがユーザーを削除するには、**User admin**ロールが必要です。

監査ログ

Audit logsでは、監査のため、管理者のアクティビティの包括的なログにアクセスできます。これには、設定、加盟店、ユーザー、および展開情報への変更に関するログが含まれています。

> **ユーザーアクセス**
>
> ユーザーが監査ログを閲覧するには、**System admin**ロールが必要です。

監査ログの検索

デフォルトでは、**Audit log list**に最も新しい監査ログエントリーが表示されます。以下を使用すると、リストをフィルタリングできます。

- **From** - この日付以降のエントリーのみが表示されます。
- **To** - この日付以前のエントリーのみが表示されます。
- **User** - 監査対象のエントリーを実行したユーザーのユーザー名を完全または部分一致検索します。
- **Revision type** - データベース・エンティティで実行された操作のタイプ。
  - **Addition** - データベース・エンティティに追加された新規レコード。
  - **Modification** - データベース・エンティティで変更された既存のレコード。
  - **Deletion** - データベース・エンティティから削除された既存のレコード。
- **Entity name** - 監査の影響を受けるデータベースのテーブル。

Searchボタンを選択すると、関連する**Audit logs**が表示されます。***Clear***ボタンを使用すると、検索フィールドがリセットされます。

監査ログ・リスト

Audit log listには、検索条件によって返されたログエントリーの数を含む**Audit logs**の表が、以下の情報と共に表示されます。

- **Entity name** - 監査の影響を受けるデータベースのテーブル。
- **Revision type** - データベース・エンティティで実行された操作のタイプ。
  - **Addition** - データベース・エンティティに追加された新規レコード。
  - **Modification** - データベース・エンティティで変更された既存のレコード。
  - **Deletion** - データベース・エンティティから削除された既存のレコード。
- **Revision date** - 監査ログの日時（dd/mm/yyyy形式）。
- **User** - 監査対象のエントリーを実行したユーザーのユーザー名。
- **IP** - ユーザーのIPアドレス。

Audit log listからログエントリーを選択すると、**Audit log details**が表示されます。

システム設定

Settingsでは、ActiveServerインスタンスのシステム設定を構成できます。**Settings**には以下の3つのタブがあります：

- 3Dセキュア2
- システム
- セキュリティー

3Dセキュア 2

3D Secure 2タブには2つのセクションがあります：

システム

- **External URL** - 認証コールバックおよび製品のアクティブ化に使用される、外部からアクセス可能なURL。
- **Auth API URL** - 認証APIリクエスト、および加盟店のクライアント証明書の生成に使用されるURL。指定しなかった場合、代わりに**External URL**が使用されます。
- **Cache refresh interval** - すべての利用可能な国際ブランドのPResキャッシュのリフレッシュ間隔。PReq/PResメッセージは、**ActiveServer**が、利用可能なACSでサポートされるプロトコルバージョン番号、DS、および3DSメソッド・コールに使用される任意のURLに関する情報をキャッシュするのに使用されます。データはDSで設定されたカード範囲ごとに整理されます。ACSによってサポートされるプロトコルバージョン番号で提供された情報、およびDSは、アプリベース、ブラウザベース、および3RIフローで使用できます。この交換が最低24時間ごと、最大1時間ごとに発生することが3DS2の仕様要件です。

タイムアウト時間

- **Preparation Response (PRes)** - PResメッセージのタイムアウト間隔
- **Authentication Response (ARes)** - AResメッセージのタイムアウト間隔
- **Challenge Response (CRes)** - CResメッセージのタイムアウト間隔

セキュリティー

- **Session timeout** - 有効期限が切れ、ユーザーにログイン認証情報の再入力を要求するまでの、ログイン・セッションが有効な間隔（単位： 分）
- **Session failed attempts** - セッション・ロック時間で指定された期間ログインが一時的に無効化されるまでの失敗ログイン試行回数。一定時間経過後、正しい認証情報を指定することでセッションを再確立できます（単位： 試行回数）。
- **Session lock time** - 失敗ログイン試行回数を超えた場合にユーザーがロックされる間隔（単位： 分）
- **Password expiry period** - 新しいパスワードの作成が要求されるまでの、パスワードが有効な日数（単位： 日数）
- **Password history check** - 特定のパスワードを再度利用できるようになるまで一意のパスワードの使用が要求される数 （単位： 一意のパスワード数）
- **Force two factor login** - サーバー上の**すべての**ユーザーに対して2要素認証を**enable**（有効化）**disable**（無効化）します。ActiveServerでは、ユーザーに2要素認証を提供するのにGoogle認証システムを使用しています。この設定が**有効化**されると、アカウントで2要素認証がまだセットアップされていないユーザーは、システム機能を使用する前に、次回ログイン時に2要素認証をセットアップすることが強制されます。Google認証システムのセットアップの手順が画面上に表示されます。

鍵ローテーション

Rotate keyを選択すると、現在の暗号化キーの作成日が表示され、ユーザーが使用される鍵をローテーションできます。

HSM

この機能を使用すると、変更された場合にユーザーがHSM PINを更新できます。

- **Full file name and path of PKCS#11 library** - この値は `application-prod.properties` から読み取られ、 `application-prod.properties` ファイルを更新し、サーバーを再起動することでのみ変更できます。
- **Slot number of HSM** - この値は `application-prod.properties` から読み取られ、 `application-prod.properties` ファイルを更新し、サーバーを再起動することでのみ変更できます。

- **HSM PIN** - 新しいHSM PINを入力できます。

Test HSM connectionボタンを選択すると、入力した**HSM PIN**を使用したHSMへの接続が試行されます。テストが成功すると、システムによって"HSM connection successful"というメッセージが表示されます。失敗した場合は"Invalid HSM Pin"と表示されます。

Updateボタンを選択すると、**HSM PIN**の値でデータベースが更新されます。**更新後はサーバーの再起動が必要です**。

> **Warning**
>
> HSM PINテストの結果にかかわらず、HSM PINは更新されます。これは、必要に応じて、HSM PINが変更される前に、**ActiveServer**データベースを更新できるようにするためです。誤ったHSM PINを使用すると取引が失敗するため、システムを更新する前に正しいPINが入力されていることを確認してください。

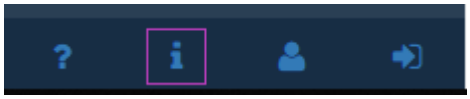
> **Version 1.0.4**
>
> この機能はバージョン1.0.4リリースで追加されました。

システム全体

- **Log level** - コンソール出力およびシステムログの詳細度。利用可能な値(右に行くほど詳細度が上がります）：**ERROR** > **INFO** > **DEBUG**.

ActiveServer情報を見る

About ActiveServerページは、ページの左下のアイコンからアクセスできます。



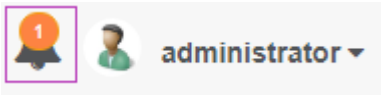
About ActiveServerには、ソフトウェア展開に関する基本情報が表示されます。これには以下が含まれます。

- **ActiveServer version** - 現在インストールさているActiveServerのソフトウェアバージョン。
- **OS name** - 使用しているオペレーティング・システム。
- **OS version** - 使用しているオペレーティング・システムのバージョン。
- **Database name** - 使用しているデータベース。
- **Database version** - 使用しているデータベースのバージョン。
- **Java edition and version** - インストールされている Javaのエディションとバージョン。
- **Node count** - このActiveServerのインスタンスに展開されているノードの数。
- **Supported 3DS version** - サポートされている3Dセキュアバージョン。
- **3DS Server reference number** - このActiveServerのインスタンスの一意の参照番号。
- **Update available** - ActiveServerの新しいバージョンに更新できるかどうかを示します。

通知

システム通知は、重要なイベントに注意が必要なときにユーザーに対して表示されます。すべてのユーザーには、パスワードの有効期限がもうすぐ切れるなど、アカウントベースの通知に関連する通知が送信されます。**System admins**には、ライセンスや使用状況のアップロードの問題など、サーバーイベントに関する通知も表示されます。

通知は、📡アイコンを選択することで管理インターフェイスの右上に表示されます。



システム通知

システム通知は**System admins**に対して表示され、監視していない場合、システム実行手順に悪影響を及ぼす可能性があります。以下の表は、発生する可能性があるシステム通知とその解決方法について示しています。

通知	シナリオ	通知メッセージ	解決策
ライセンスなし	ソフトウェアが最初に初期化されたときに、システムにライセンスがなくなり、製品がアクティブ化されるまでこの通知が表示されます。	**ライセンス警告：**このインスタンスはアクティブ化されていません。**Deployment > Activation status**ページで**Product Activation Key (PAK)**を追加してください。PAKはGPaymentsの**MyAccount activation**ページにあります。	製品を正しくライセンス認証するため、ユーザーはライセンスガイドに従う必要があります。
ライセンスの問題：警告	GPaymentsのライセンス・サーバーは、現在のインスタンスに関連付けられたアカウントで決済が未決であることを示しています。これは指定された期間内に予告なく無効化されます。	**ライセンス警告：**アカウントの期限が切れているため、このインスタンスはy日で無効化されます。詳細については、**GPayments support**にお問い合わせください。	ユーザーが請求の問題を解決するには、適宜アカウント・チームに連絡し、GPaymentsサポートに問い合わせる連絡する必要があります。

通知	シナリオ	通知メッセージ	解決策
ライセンスの問題：停止	GPaymentsのライセンス・サーバーは、現在のインスタンスに関連付けられたアカウントで決済が未決であることを示しています。現在のインスタンスは無効化されています。	**ライセンス警告：**アカウントの期限が切れているため、このインスタンスは無効化されています。詳細については、**GPayments support**にお問い合わせください。	ユーザーが請求の問題を解決するには、適宜アカウント・チームに連絡し、GPaymentsサポートに問い合わせる連絡する必要があります。
アップロードの問題：警告	ActiveServerインスタンスが一定期間GPaymentsライセンス・サーバーへのアップロードに失敗する場合、システムは警告プロセスを開始し、サーバーが無効化されるまで、ユーザーにエラーを修正するための期間を60日間提供します。	**ライセンス警告：**GPaymentsライセンス・サーバーに対する正常な認証の使用がx日間報告されていないため、このインスタンスはy日で無効化されます。詳細については、**GPayments support**にお問い合わせください。	ユーザーは使用状況のアップロードが失敗する理由を調査するか、問題解決のための支援についてGPaymentsサポートに問い合わせる必要があります。
アップロードの問題：停止	ActiveServerインスタンスが60日間GPaymentsライセンス・サーバーへのアップロードに失敗した場合、サーバーは無効化されます。	**ライセンス警告：**GPaymentsライセンス・サーバーに対する正常な認証の使用が60日間報告されていないため、このインスタンスは無効化されました。詳細については、**GPayments support**にお問い合わせください。	ユーザーは使用状況のアップロードが失敗する理由を調査するか、問題解決のための支援についてGPaymentsサポートに問い合わせる必要があります。

ユーザー通知

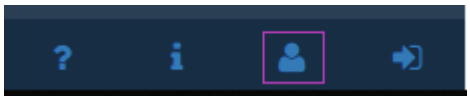
ユーザー通知は、イベントがアカウントに影響を与える場合にユーザーに対して表示されます。以下の表は、発生する可能性があるユーザー通知とその解決方法について示しています。

通知	シナリオ	通知メッセージ	解決策
パスワード期限切れ	ユーザーのパスワードの有効期限が切れるまで、あと7日しか残っていません。	**user**のパスワードは**expiry date**に期限切れになります。	ユーザーは**User profile > Change password**画面からパスワードを更新する必要があります。

ユーザープロフィール

User profileでは、ユーザーが自身のアカウントに関する詳細を編集したり、パスワードを変更したりすることができます。

User profileにアクセスするには、管理インターフェイスの左下の**Profile**アイコンを選択します。 **edit profile**と**change password**の2つのセクションがあります。



> **Note**
>
> すべてのユーザーは自分のユーザープロフィールを管理できます。

プロフィールを編集

ユーザーのアカウント詳細には以下が含まれます：

- **Username (required)** - 管理インターフェイスにログインするのに使用される名前。
- **First name (required)** - ユーザーの名前。
- **Last name (required)** - ユーザーの名字。
- **Email (required)** - ユーザーの完全修飾電子メールアドレス。**このアドレスは、パスワードリセットやその他のシステム通知を送信するのに使用されます。**
- **Time zone** - 管理ダッシュボード上において表示される日時のローカル・タイムゾーン。
- **Two factor login status (required)** - 現在のユーザーについて2FAの**enable**（有効）または**disable**（無効）を切り替えます。無効化された状態から有効に切り替えると、ユーザーは、画面に表示される手順に従ってGoogle認証システムを使用して2FAをセットアップするように要求されます。
- **Enabled** - ユーザーはログイン時にこのアカウントに関連付けられた認証コードを入力するよう求められます。そうしないとログインが失敗します。
- **Disabled** - 2FAが無効化され、ログイン時に2FAが必要なくなります。

> **Important**
>
> インスタンスで強制2要素認証が有効な場合、ユーザーは2FAを無効化できなくなります。

管理APIクライアント証明書

- **Download** - 管理APIリクエストに使用されるユーザーのクライアント証明書のダウンロードを許可します。この機能の詳細については、APIドキュメント概要を参照してください。
- **Revoke** - セキュリティが侵害された場合、現在のクライアント証明書を失効させ、新しいIDで証明書を再発行します。

> **Warning**
>
> **Revoke**した場合過去に発行されたクライアント証明書は失効され、新しいクライアント証明書をダウンロードするまではAPIリクエストができなくなります。

CA証明書

- **Download** - 管理APIにリクエストを送信する際に必要なサーバーCA証明書をダウンロードします。この機能の詳細については、APIドキュメント概要を参照してください。

> **Tip**
>
> **管理APIクライアント証明書**と**CA証明書**の機能はアクティブ化されたインスタンスでのみご利用可能です。

> **バージョン1.0.3**
>
> 証明書のダウンロードはバージョン1.0.3で追加されました。

パスワード変更

ユーザーは以下のフィールドを入力することでパスワードを変更できます。

- **Current password (required)** - ユーザーの現在のパスワード。正しくない場合は、パスワードの変更が失敗します。

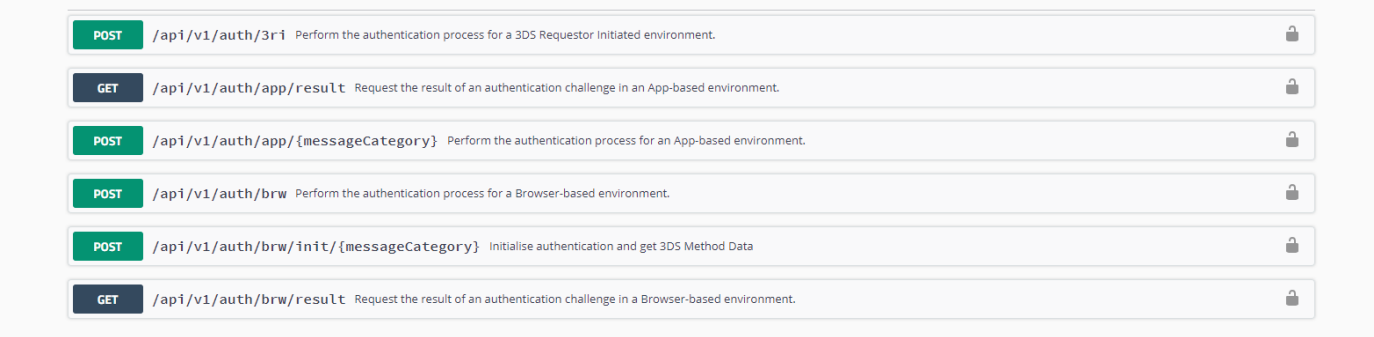
- **New password (required)** - ユーザーの新しいパスワード。パスワード履歴チェックルールを確認する必要があります。
    - *Requirements - 8～100文字で、少なくとも1つの文字と1つの数字を含む必要があります。*
- **Confirm new password** - ユーザーの新しいパスワード。**New password**フィールドと一致する必要があります。

APIドキュメントまとめ

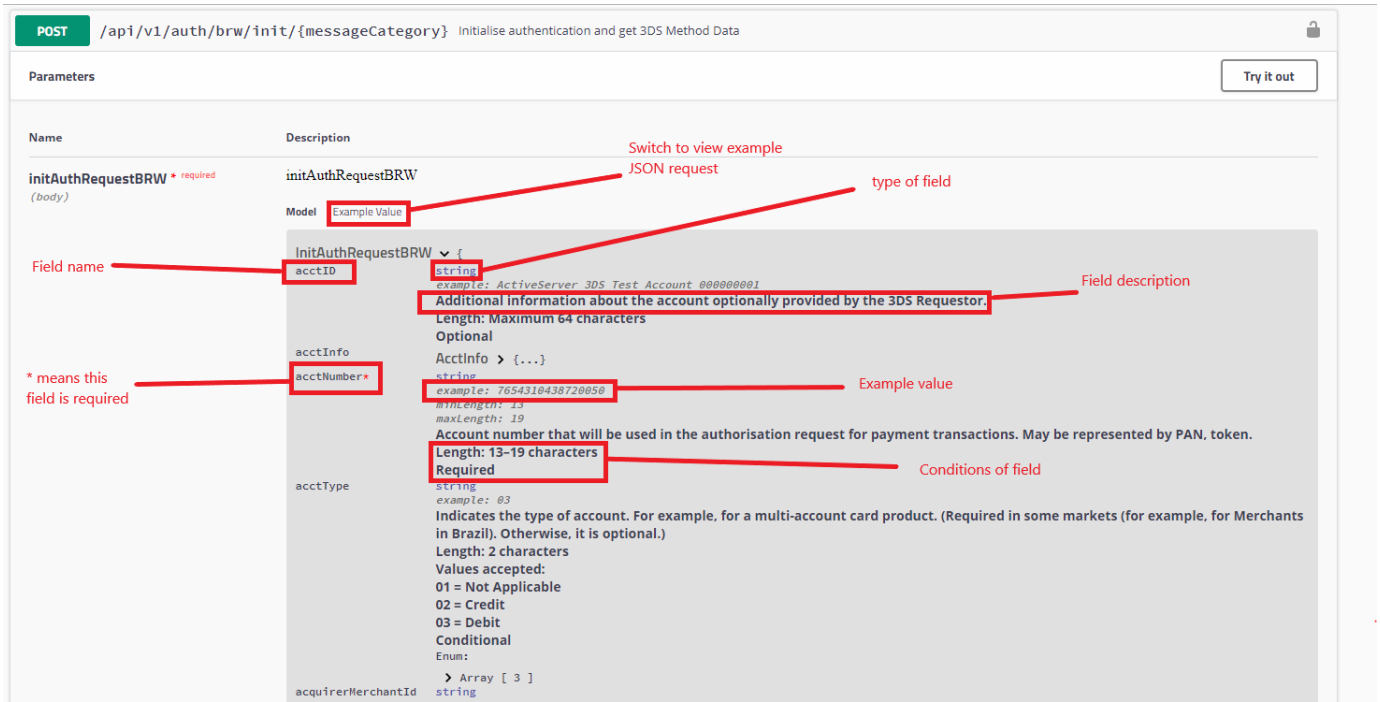
はじめに

APIドキュメントの読み方

GPaymentsのActiveServer APIのドキュメントには、各APIの許可されるエンドポイントおよびリクエストHTTPメソッドが含まれています。たとえば、 `/api/v1/auth/3ri` はエンドポイントURLであり、許可されるHTTPメソッドは `POST` です。



エンドポイントの詳細を確認するには、エンドポイント・ボックスのいずれかをクリックし、詳細を表示します。以下のスクリーンショットのように表示されます。このスクリーンショットは、APIドキュメントの読み方を説明しています。



認証API

認証APIを使用すると、加盟店と決済代行会社が3Dセキュア2フローをeコマースサイトに統合できます。すべての認証APIは `/api/v1/auth/...` で始まり、RESTFulの命名規則に従っています。利用可能な主な認証フローには3つあります。

- **アプリベース** - 登録されたエージェント（加盟店、デジタル・ウォレットなど）によって指定されたアプリから発生したコンシューマー・デバイス上での取引中の認証。たとえば、コンシューマー・デバイス上の加盟店のアプリ内でチェックアウト・プロセス中に発生したeコマース取引などです。
- **ブラウザベース** - ブラウザを使用してWebサイトから発生したコンシューマー・デバイスまたはコンピュータ上での取引中の認証。たとえば、Webサイト内でチェックアウト・プロセス中に発生したeコマース取引などです。
- **3DS Requestor Initiated (3RI)** - 直接カード会員を介さないアカウント情報の確認。たとえば、アカウントがまだ有効であることを確認するサブスクリプションベースのeコマース加盟店などです。

最新の認証APIはこちらで確認できます。

項目条件の説明

各項目には下記のいずれかの条件が記載されています。

- **必須** - この項目はリクエストを送信する際に必ず必要です。ActiveServerはこの項目の有無を確認し、妥当性の確認を行います。必須項目には*印が記載されています。
- **条件付き** - 項目に記載された条件が満たされていた場合にはリクエストに定義する必要があります。項目が定義されていればActiveServerは妥当性の確認を行います。送信されたJSONにこの項目が定義されていなければActiveServerは妥当性の確認を行いません。
- **オプション** - オプションでリクエストに定義できます。項目が定義されていればActiveServerは妥当性の確認を行います。送信されたJSONにこの項目が定義されていなければActiveServerは妥当性の確認を行いません。

最新の認証APIはここからアクセスできます。

API認証方法

すべての認証APIエンドポイントは、クライアントとサーバーの手動による認証が必要なX.509認証によって保護されています。これは、ActiveServerによって発行されたクライアント証明書を使用して実行されます。

1.このガイドに従ってインスタンスがアクティブ化されていることを確認します。

2.ダッシュボードの加盟店ページから認証APIのクライアント証明書をダウンロードします。この証明書は `.p12` 形式です。

3.ユーザープロフィールページからCA証明書チェーンをダウンロードします。CA証明書は、最初の証明書がサーバーCAで、次にGPayments中間CA、最後にGPaymentsルートCAとなっている `.pem` 形式です。

4.統合ガイドまとめに従って、クライアント証明書を使用して認証APIをコールするように3DSリクエスターをセットアップできます。

> **Warning**
>
> インスタンスがアクティブ化されていない場合、 `DOWNLOAD` ボタンは表示されません。

管理API

管理APIは、加盟店の管理などの管理タスクの選択を実行するのに利用できます。すべての管理APIは `/api/v1/admin/...` で始まり、RESTFulの命名規則に従っています。たとえば、こちらで説明されている `creating merchant` は、POST APIエンドポイント **`/api/v1/admin/merchants`** で実行できます。

最新の管理APIはこちらで確認できます。

API認証方法

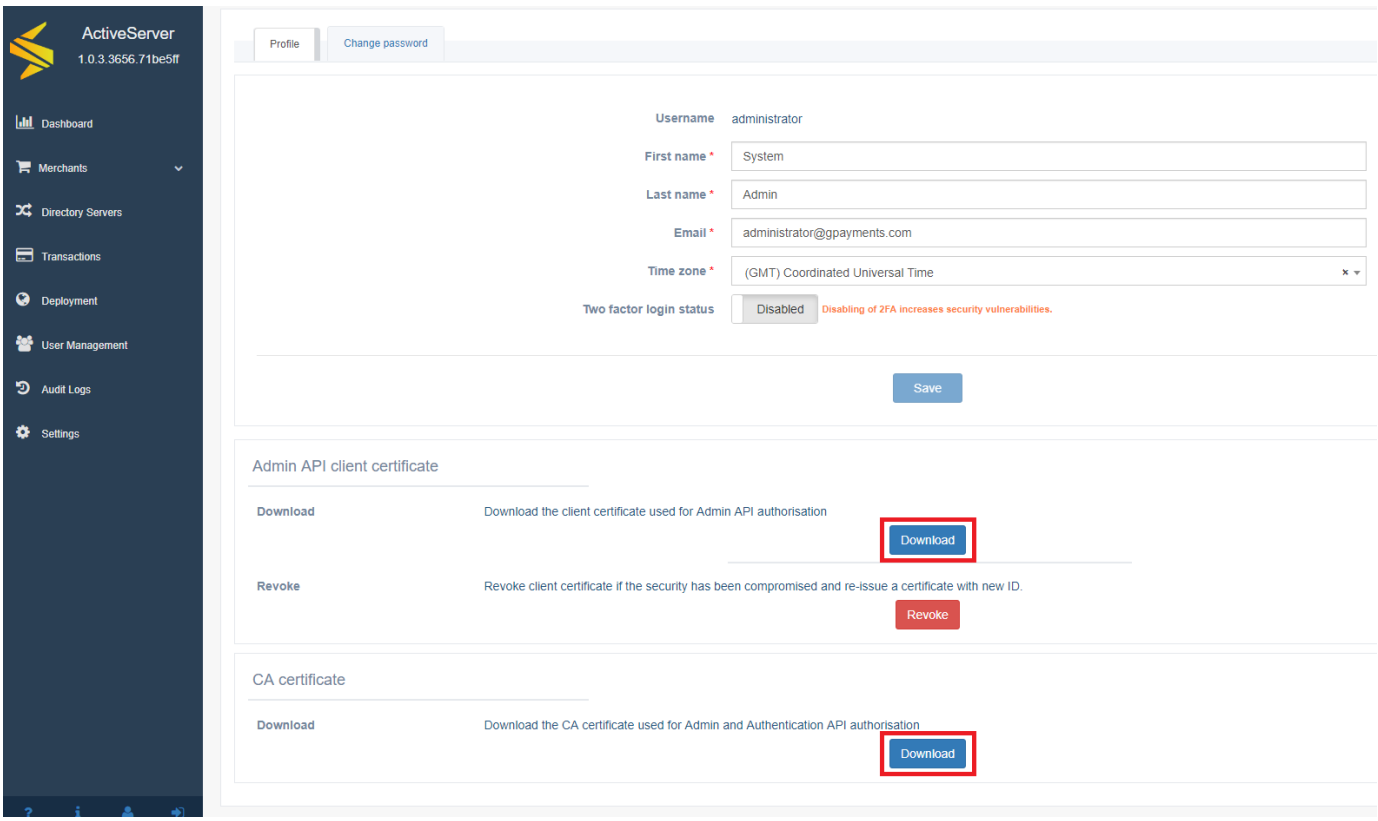
API認証はクライアント証明書を使用した認証APIと同様に実行されます。

1.このガイドに従ってインスタンスがアクティブ化されていることを確認します。

2.ダッシュボードのuser profileページから管理APIのクライアント証明書をダウンロードします。この証明書は `.p12` 形式です。

3.同ページからCA証明書チェーン証明書をダウンロードします。CA証明書は、最初の証明書がサーバーCAで、次にGPayments中間CA、最後にGPaymentsルートCAとなっている `.pem` 形式です。

4.HTTPクライアント・リクエスト内に含めることで、API認証に対してダウンロードしたクライアント証明書とCAバンドルを使用します。この方法については使用しているHTTPクライアントを参照してください。テストについてはAPIクイック・スタートに従ってください。



管理APIクイック・スタート

cURLを使用してAPIを試すには、**openssl**を使用し、以下の `curl` コマンドを使用して、X.509証明書をPEM形式に変換する必要があります。

1.https://www.openssl.org/からopensslコマンド ラインをダウンロードしてインストールします。

2.上記の管理API認証方法セクションに従って、クライアント証明書とCA証明書をダウンロードします。

3.ターミナルを開きます。

4.以下のコマンドを使用して `.p12` ファイルから秘密鍵を抽出します。要求された場合はパスワードを入力します。

```bash
openssl pkcs12 -in YOUR_P12_FILE_NAME.p12 -nocerts -out key.pem
```

5.以下のコマンドを使用して `.p12` ファイルから証明書を抽出します。要求された場合はパスワードを入力します。

```bash
openssl pkcs12 -in YOUR_P12_FILE_NAME.p12 -clcerts -nokeys -out crt.pem
```

6.AS管理APIエントリー ポイントを使用して、cURLリクエストを実行します。この場合、`/api/v1/admin/merchants` から加盟店のリストが取得されます。

```bash
curl -k https://your.server.address:7443/api/v1/admin/merchants --cacert
cacerts.pem --cert crt.pem -v --key key.pem
```

7.以下のcURLレスポンスが表示されるはずです。

```bash
< HTTP/1.1 200 OK
< Expires: 0
< Cache-Control: no-cache, no-store, max-age=0, must-revalidate
< X-XSS-Protection: 1; mode=block
< Pragma: no-cache
< X-Frame-Options: DENY
< Date: Mon, 27 May 2019 09:51:59 GMT
< X-Total-Count: 1
< Connection: keep-alive
< X-Content-Type-Options: nosniff
< Strict-Transport-Security: max-age=31536000 ; includeSubDomains
< Transfer-Encoding: chunked
< Content-Type: application/json;charset=UTF-8
< Link: </api/v1/admin/merchants?page=0&size=20>; rel="last",</api/v1/admin/
merchants?page=0&size=20>; rel="first"
<
* Connection #0 to host your.server.adress left intact
[{"bins":"40001 (Test Acquirer)","merchantId":"123456789012345","name":"Test
Merchant","status":"ENABLED","merId":"48f3b030-ed1d-4f7f-aa70-85cda288e0cb"}]
%
```

エラーコード

このセクションでは、ActiveServerの実行中に発生する可能性があるエラーの詳細について説明します。

3DSエラーコード(101 ~ 404)

コード	名前	説明
101	MESSAGE_RECEIVED_INVALID	受信したメッセージが無効です。
102	MESSAGE_VERSION_NUMBER_NOT_SUPPORTED	サポートされていないメッセージバージョン番号です。
103	SENT_MESSAGES_LIMIT_EXCEEDED	送信済みメッセージが上限を超えました。
201	REQUIRED_DATA_ELEMENT_MISSING	仕様に従って定義された必須のメッセージ要素がありません。
202	CRITICAL_MESSAGE_EXTENSION_NOT_RECOGNISED	重要なメッセージ拡張が存在しません。
203	FORMAT_OF_ONE_OR_MORE_DATA_ELEMENTS_IS_INVALID_ACCORDING_TO_THE_SPECIFICATION	データ要素が要求されている形式ではないか、仕様に従って定義された値が無効です。
204	DUPLICATE_DATA_ELEMENT	重複したデータ要素が見つかりました。
301	TRANSACTION_ID_NOT_RECOGNISED	コンポーネントの受信について、受信した取引IDは無効です。
302	DATA_DECRYPTION_FAILURE	データの暗号化が失敗しました。
303	ACCESS_DENIED_INVALID_ENDPOINT	APIリクエストのエンドポイントが無効です。リクエストURLを確認してください。
304	ISO_CODE_INVALID	ISOコードが無効です。

コード	名前	説明
305	TRANSACTION_DATA_NOT_VALID	取引データが無効です。
306	MERCHANT_CATEGORY_CODE_MCC_NOT_VALID_FOR_PAYMENT_SYSTEM	加盟店カテゴリーコードが無効です。
307	SERIAL_NUMBER_NOT_VALID	シリアル番号が無効です。
402	TRANSACTION_TIMED_OUT	取引がタイムアウトしました。
403	TRANSIENT_SYSTEM_FAILURE	システムが短期間故障しました。
404	PERMANENT_SYSTEM_FAILURE	システムが恒久的に故障しました。
404	SYSTEM_CONNECTION_FAILURE	システムに接続できませんでした。

取引エラーコード(1000 ~ 1026)

コード	名前	説明
1001	DIRECTORY_SERVER_NOT_FOUND	指定されたプロバイダーのディレクトリ・サーバーが見つかりませんでした
1002	ERROR_SAVE_TRANSACTION	取引の保存中にエラーが発生しました
1003	ERROR_SAVE_TRANSACTION_MESSAGE	取引メッセージの保存中にエラーが返されました
1004	UNHANDLED_EXCEPTION	未処理の例外
1005	PAN_NOT_PARTICIPATING	プライマリ・アカウント番号(PAN)が参加していません。
1009	MERCHANT_INTERFACE_DISABLED	この加盟店のインターフェイスは無効です
1011	INVALID_LICENSE	加盟店に有効なライセンスがありません
1013	INVALID_TRANSACTION_ID	3DSサーバーの取引IDを認識できません

コード	名前	説明
1014	INVALID_REQUESTOR_TRANSACTION_ID	3DSリクエスターの取引IDを認識できません
1015	THREEDS_REQUESTOR_NOT_FOUND	3DSリクエスターID/加盟店IDが無効です
1016	MISSING_REQUIRED_ELEMENT	必須要素がありません。
1018	ELEMENT_NOT_DEFINED	メッセージ要素が定義済みメッセージではありません。
1019	PROTOCOL_OLD	プロトコルバージョンが古すぎます。
1020	ERROR_TRANSMISSION_DATA	データ転送中にエラーが発生しました。
1021	PRIOR_TRANS_ID_NOT_FOUND	リクエスターの事前取引IDの設定中にエラーが発生しました。事前取引IDが見つかりませんでした。
1022	INVALID_FORMAT	仕様に従って、1つ以上の要素の形式が無効です。
1023	CARD_RANGE_IS_NOT_VALID	指定されたカード範囲が無効です。
1024	CACHE_UPDATE_IS_DISABLE	キャッシュ更新が無効です。
1025	CACHE_REFRESH_INTERVAL_IS_NOT_SET	キャッシュリフレッシュ間隔が設定されていません。
1026	MERCHANT_ID_THREEDS_REQUESTOR_ID_INVALID	acquirerMerchantID/threeDSRequestorIDが無効です

汎用エラーコード(2000 ~ 2010)

コード	名前	説明
2000	NOT_FOUND	リソースが見つかりません。
2001	DUPLICATE_RECORD	レコードがすでに存在します。

コード	名前	説明
2002	VALIDATION_ERROR	入力が無効です。
2003	INVALID_REQUEST	リクエストが無効です。
2004	CONCURRENCY_FAILURE	ノードの更新に失敗しました。
2005	ACCESS_DENIED	アクセスが拒否されました。
2006	METHOD_NOT_SUPPORTED	リクエストHTTPメソッドがサポートされていません。
2007	INTERNAL_SERVER_ERROR	内部サーバーエラー。
2008	DATA_INTEGRITY_VIOLATION_ERROR	指定された値は整合性制約に違反しています。
2009	SESSION_TIMED_OUT	セッションがタイムアウトしました。

セキュリティ・エラーコード(3001 ~ 3024)

コード	名前	説明
3002	NO_SUCH_ALGORITHM	そのようなアルゴリズムはありません。
3003	INVALID_CERT	証明書の公開鍵には対応する秘密鍵との互換性がありません。
3004	INVALID_CHAIN	CA証明書ストアで1つ以上の中間証明書が見つからないため、ActiveServerは完全な証明書チェーンを構築できません。再度試す前に、完全なチェーンを含む証明書をインストール/インポートするか、見つからない中間証明書をインストールする必要があります。
3005	NO_PRIVATE_KEY_FOUND	秘密鍵が見つかりませんでした。
3006	INVALID_CERTIFICATE_CONTENT	証明書のコンテンツが無効です

コード	名前	説明
3007	CERTIFICATE_IO_READ	証明書を読み取ることができませんでした。
3008	SUCH_PROVIDER_EXCEPTION	そのようなプロバイダー例外はありません。
3009	NO_KEY	このオブジェクトには既存のキーがないため、証明書をインストールできませんでした。
3010	CERTIFICATE_CHAIN_BAD_FORMAT	証明書チェーンの形式が無効です。
3011	MISMATCHED_PASSWORDS	パスワードフィールドが一致しません。
3012	IMPORT_CERTIFICATE	クライアント証明書をインストールしてください。
3013	IMPORT_NO_CERTIFICATE	エクスポートする証明書がありません。
3014	FAILED_TO_INITIALIZE	初期化に失敗しました。
3015	ENCRYPTION_FAIL	暗号化に失敗しました。
3016	DECRYPTION_FAIL	復号化に失敗しました。
3017	INVALID_HSM_PROVIDER	"ハードウェア暗号化用に指定されたプロバイダー名はサポートされていません
3018	INVALID_PKCS11_CONFIG	PKCS11設定パスが無効です
3019	FAILED_TO_INITIALIZE_PKCS11	PKCS11の初期化に失敗しました。
3020	IMPORT_FAIL	インポートに失敗しました。
3021	NOT_SUPPORTED_IBM_PROVIDER	SUNプロバイダーのみがサポートされています。
3022	UNABLE_TO_LOAD_KEYSTORE	キーストアのロードに失敗しました。
3023	UNABLE_TO_LOAD_CERTIFICATE	証明書のロードに失敗しました。
3024	INVALID_KEY_SIZE	キーサイズが無効です。

ユーザー・エラーコード(4000 ~ 4022)

コード	名前	説明
4000	DUPLICATE_EMAIL	電子メールはすでに使用されています。
4001	LAST_ADMIN_DELETE_NOT_ALLOWED	このアクションを実行するには、少なくともシステム管理者ユーザーである必要があります。
4002	ACCOUNT_IS_LOCKED	アカウントがロックされています。
4003	ACCOUNT_IS_DISABLED	アカウントが無効です。
4004	ACCOUNT_WILL_BE_LOCKED	あと1回誤るとアカウントがロックされます。パスワードを忘れた場合は、「Lost your password」をクリックしてください。
4005	ACCOUNT_WAS_LOCKED	パスワードは1時間ロックされます。
4006	ACCOUNT_IS_INACTIVE	アカウントがアクティブ化されていません。
4007	PASSWORD_POLICY_MATCH	パスワードは最低8文字であり、少なくとも1つの文字と1つの数字が含まれている必要があります。
4008	LOGIN_ALREADY_IN_USE	ユーザー名はすでに使用されています。
4009	EMAIL_ALREADY_IN_USE	電子メールはすでに使用されています。
4010	INVALID_TOTP_CODE	totp認証コードが無効です。
4011	EMAIL_SENDING_FAILED	電子メールの送信に失敗しました。
4012	EMAIL_NOT_REGISTERED	電子メールが登録されていません。
4014	FAILED_TO_CREATE_ACCOUNT	アカウントの作成に失敗しました。
4015	TWO_FA_MANDATORY	2要素認証の使用は必須です。
4016	PASSWORD_EXPIRED	ユーザーのパスワードの有効期限が切れました。

コード	名前	説明
4017	PASSWORD_EXPIRED_WARNING	ユーザーのパスワードが期限切れになります。
4018	PASSWORD_HISTORY_MATCHED	パスワードが過去のパスワードと一致しました。
4019	INVALID_TOKEN	トークンが無効です。
4020	INVALID_HSM_PIN	HSM PINが無効です。
4021	INVALID_PASSWORD	パスワードが無効です。
4022	EMAIL_INVALID_ACTIVATION	アカウントアクティブ化コードが無効です。

セットアップ・エラーコード(5000)

コード	名前	説明
5000	SETUP_NOT_ALLOWED	セットアップが許可されていません。

用語集

このページでは、3Dセキュア2に関連する用語の一覧を提供しています。このドキュメントの他の場所で使用されていないものもありますが、内容を完全にするために記載しています。 用語を調べる場合や、よくわからない単語に遭遇したときにこのページを参照してください。

Term	Acronym	Definition
3DS Client		EMV 3-Dセキュア・プロトコルを開始するための消費者と3DSリクエスターのやり取りを容易にする、ブラウザベースまたはモバイルアプリのオンラインショッピングサイトなどの消費者向けコンポーネント。
3DS Integrator		3DSリクエスター環境を容易化および統合し、オプションで加盟店とアクワイアラー間の統合を容易にするEMV 3-Dセキュア参加者。
3DS Requestor		AReqメッセージとも呼ばれるEMV 3-Dセキュア認証リクエストのイニシエーター。たとえば、購入フロー内で認証をリクエストする加盟店やデジタル・ウォレットなどです。
3DS Requestor App		3DS SDKを使用して、3-Dセキュア取引を処理できるコンシューマー・デバイス上のアプリ。3DSリクエスターアプリは、3DS SDKとの統合によって有効化されます。
3DS Requestor Environment		これは、通常、3DSインテグレーターによって容易化される加盟店/アクワイアラードメインの、3DSリクエスター制御コンポーネントを指します。これらのコンポーネントには、3DSリクエスターアプリ、3DS SDK、3DSサーバーが含まれます。3DSリクエスター環境の実装は、3DSインテグレーターで定義されているとおりに変化します。
3DS Software Development Kit	**3DS SDK**	3-Dセキュア・ソフトウェア開発キット。3DSリクエスターアプリに組み込まれているコンポーネント。3DS SDKは、3DSサーバーの代わりに3-Dセキュアに関連する機能を実行します。
3DS Requestor Initiated	**3RI**	アカウントがまだ有効であることを確認するために、3DSリクエスターによって開始された3-Dセキュア取引。主なユースケースは、サブスクリプションユーザーの決済方法がまだ有効であることを確認するために、加盟店が非決済取引を実行することを要求する定期的な取引（TVサブスクリプション、公共料金決済など）です。

Term	Acronym	Definition
3DS Server	**3DSS**	オンライン取引を処理し、3DSリクエスターおよびディレクトリ・サーバー間の通信を容易にする3DSインテグレーターのサーバーまたはシステムを指します。
3-D Secure	**3DS**	3ドメイン・セキュア。バージョン2以降では、決済、非決済、およびアカウント確認カード取引の安全な処理が可能なeコマース認証プロトコル。
Access Control Server	**ACS**	イシュアードメインで機能するコンポーネント。カード番号およびデバイスタイプで認証が利用可能かどうかを確認し、特定のカード会員を認証します。
Attempts		EMV 3DS仕様では、決済認証が利用できない場合に認証試行の証明が生成されるプロセスを示すのに使用されます。試行のサポートは各DSで判断されます。
Authentication		3-Dセキュアのコンテキストでは、eコマース取引を行った個人に決済カードを使用する資格があることを確認するプロセスです。
Authentication Request Message	**AReq**	認証プロセスを開始するために、DSを介して3DSサーバーからACSに送信されるEMV 3-Dセキュア・メッセージ。
Authentication Response Message	**ARes**	認証リクエスト・メッセージに対するレスポンスで、DSを介してACSから返されるEMV 3-Dセキュア・メッセージ。
Authentication Value	**AV**	オーソリゼーション処理中にオーソリゼーションシステムが認証結果の完全性を検証するための方法を提供する、ACSによって生成された暗号値。AVアルゴリズムは各決済システムによって定義されます。
Authorisation		イシュアーまたはイシュアーの代理のプロセッサーが決済の取引を承認するプロセス。
Authorisation System		決済システムがイシュアーおよびアクワイアラーに対してオンライン金融処理、オーソリゼーション、清算、決算サービスを提供するシステムおよびサービス。
Bank Identification Number	**BIN**	発行金融機関を一意に識別する決済カードアカウント番号の最初の6桁。ISO 7812ではイシュアー識別番号（IIN）とも呼ばれます。
Base64		RFC 2045で定義される、認証値データ要素に適用されるエンコーディング。

Term	Acronym	Definition
Base64 URL		RFC 7515で定義される、3DSメソッドデータ、デバイス情報、CReq/CResメッセージに適用されるエンコーディング。
Card		カードは_EMV 3-Dセキュア・プロトコルおよびコア機能仕様_において、決済カードのアカウントと同義です。
Certificate Authority	**CA**	デジタル証明書を発行するエンティティ。
Cardholder		カードの発行対象の個人、またはそのカードの使用がオーソリゼーションされた個人。
Challenge		ACSが3DSクライアントと通信し、カード会員とのやり取りを通じて追加情報を取得するプロセス。
Challenge Flow		_EMV 3-Dセキュア・プロトコルおよびコア機能仕様_で定義される、カード会員の操作に関する3-Dセキュア・フロー。
Challenge Request Message	**CReq**	3DS SDKまたは3DSサーバーによって送信されるEMV 3-Dセキュア・メッセージ。認証プロセスをサポートするためにカード会員からACSに追加情報が送信されます。
Challenge Response Message	**CRes**	CReqメッセージに対するACSレスポンス。カード会員認証の結果を示したり、アプリベース・モデルの場合は、認証の完了にはさらなるカード会員操作が必要であることを示す信号を示したりすることができます。
Consumer Device		カード会員が認証や購入を含む決済アクティビティを実行するのに使用するスマートフォン、ノートPC、またはタブレットなど、カード会員によって使用されるデバイス。
Device Channel		取引の発生元のチャネルを示します。以下のいずれか：・ アプリベース（01-APP） ・ ブラウザベース（02-BRW） ・ 3DSリクエスター起因（03-3RI）
Device Information		認証プロセスで使用される、コンシューマー・デバイスによって提供されるデータ。

Term	Acronym	Definition
Directory Server	**DS**	相互運用ドメインで機能するサーバーコンポーネント。以下を含むいくつかの機能を実行します： 3DSサーバーの認証、3DSサーバーおよびACS間のメッセージのルーティング、3DSサーバー、3DS SDK、および3DSリクエスターの検証。
Directory Server Certificate Authority	**DS CA**または**CA DS**	相互運用ドメインで機能するコンポーネント。認証局（DS）が選択したデジタル証明書を生成し、3-Dセキュアに参加しているコンポーネントに分配します。通常、DSが接続されている決済システムがCAを操作します。
Directory Server ID		決済システムに対して一意の登録済みアプリケーション・プロバイダー識別子（RID）。RIDはISO 7816-5標準によって定義されています。
Electronic Commerce Indicator	**ECI**	カード会員を認証するための試行の結果を示す、ACSによって提供される決済システム固有の値。
Frictionless		カード会員の介入なく行われた場合に、認証プロセスを説明するのに使用されます。
Frictionless Flow		EMVCoコア仕様セクション2.5.1で定義される、カード会員が介入しない3-Dセキュア・フロー。
Message Authentication Code	**MAC**	第三者によるデータの改変および偽造から送信者および受信者を保護する対称（秘密鍵）暗号化方式。
Merchant		決済カードを使用して行われた決済を受理するためにアクワイアラーと契約しているエンティティ。加盟店は、カード番号を取得してカード会員のオンラインショッピング体験を管理し、その後決済認証を行う3DSサーバーに制御を移します。
Non-Payment Authentication	**NPA**	取引が付随しない3DS認証タイプ。アイデンティティ確認に使用されます。
One-Time Passcode	**OTP**	コンピューターシステムまたはその他のデジタル・デバイス上の1ログイン・セッションまたは取引に対してのみ有効なパスコード。

Term	Acronym	Definition
Out-of-Band	**OOB**	3-Dセキュア・フロー外で並行して実行されるチャレンジ・アクティビティ。最終チャレンジ・リクエストは、ACSによってチェックされるデータの送信には使用されませんが、認証が完了したことのみを通知します。ACS認証方式または実装は3-Dセキュア仕様では定義されません。
Preparation Request Message	**PReq**	DSカード範囲に対応するACSおよびDSプロトコルバージョン、およびオプションで3DSサーバーの内部ストレージ情報を更新するための3DSメソッドURLを要求する、3DSサーバーからDSに送信される3-Dセキュア・メッセージ。
Preparation Response Message	**PRes**	3DSサーバーの内部ストレージを更新できるようにするための、DSカード範囲、ACSおよびDSのアクティブ・プロトコル・バージョン、3DSメソッドURLを含むPReqメッセージに対するレスポンス。
Proof or authentication attempt		試行を参照。
Registered Application Provider Identifier	**RID**	決済システムに対して一意の登録済みアプリケーション・プロバイダー識別子（RID）。RIDはISO 7816-5標準によって定義されており、ISO/IEC 7816-5登録局によって発行されます。RIDは5バイトです。
Results Request Message	**RReq**	3DSサーバーに認証取引の結果を送信するために、ACSからDSに送信されるメッセージ。
Results Response Message	**RRes**	結果リクエスト・メッセージの受信を確認するために、DSを介して3DSサーバーからACSに送信されるメッセージ。

ドキュメントバージョン

リリースノート

ActiveServer v1.0.5

[Release Date: 04/07/2019]

変更点	詳細
[#322] 修正	日付時刻がユーザーのローカルタイムゾーンで表示されない点を修正しました。(User Profileページで設定されたタイムゾーン)
[#378] 改善	Merchantページの詳細からCA証明書をダウンロードする機能を追加しました。
[#401] 変更	新規にインストールした場合のシステムキーストアの名前を「as_sys_randomUUID.jks」に変更しました。
[#402] 修正	"3DS Server Transaction ID"、"Min purchase amount"、"Max purchase amount"を取引のフィルターに設定した場合に表示される結果が違う問題を修正しました。
[#412] 修正	ユーザーがパスワードを間違えた回数が上限を超えた場合にロックされる問題を修正しました。
[#422] 修正	Directory Servers > Settings > HTTPS callback portにおいて使用されているポートと違うポートが表示される問題を修正しました。
[#428] 変更	/api/v1/auth/3ri 認証APIリクエストにおいて{messageCategory}を追加しました。
[#433] 変更	.htmlの接尾辞を全てのページから削除しました。
[#446] 改善	Merchantページの詳細でエラーが発生した際のエラーメッセージを改善しました。
[#448] 改善	Directory Server証明書のインポートのロジックとエラー処理を改善しました。

変更点	詳細
[#449] 改善	Directory Server > Settings > 3DS Server URL (前External URL), Directory Server > Settings > HTTP listening port (前HTTPS callback port), Settings > 3DS2 > API URL (前Auth API URL)システムラベルの読みやすさを改善しました。
その他	小規模なバグ修正、パフォーマンス、セキュリティの強化

ActiveServer v1.0.4

[Release Date: 31/05/2019]

変更点	詳細
[#386] 修正	HSMが使用された時にアクティブ化がエラーになる問題を修正しました。
[#390] 改善	Settings > SecurityページからHSM PINを変更する機能を追加しました。
[#380] 改善	Amazon Aurora MySQL 5.7のサポートを追加しました。

ActiveServer v1.0.3

[リリース日：2019/05/27]

変更点	詳細
[#376] 変更	結果の列挙を `00` または `01` の値で提供するように `enrol` APIレスポンスを更新しました。
[#379] 修正	ダッシュボード履歴データが表示されない場合があるという問題を修正しました。
[#380] 修正	アクセス時にDS enum値が古い加盟店でエラーが表示されるという問題を修正しました。

ActiveServer v1.0.2

[リリース日：2019/05/24]

変更点	詳細
データベース・サポート	MSSQL Server 2017のサポートを追加しました。
[#301] 改善	.x509認証を使用するように管理APIエンドポイントを更新しました。
[#349] 変更	ログファイルの形式をas.dd-mm-yyyy.logからas.yyyy-mm-dd.logに変更し、ベースのログフォルダに保存されるように変更しました。
[#356] 変更	application-prod.propertiesのDSポートのデフォルト値が9600の範囲になるように変更しました。
[#368] 修正	`enrol` APIが内部サーバーエラーを返すという問題を修正しました。
[#373] 改善	User ProfileページにAPIリクエストで使用されるCA証明書のダウンロードを追加しました。

ActiveServer v1.0.1

[リリース日：2019/05/17]

変更点	詳細
[#326] 修正	一部のブラウザでサイドメニューの読み込みが遅くなるという問題を修正しました。
[#327] 修正	Oracle DBの使用時の互換性の問題を修正しました。
[#328] 変更	AuthResponseApp APIにacsReferenceNumberを追加しました。
その他	小規模なバグ修正、パフォーマンス、セキュリティの強化

ActiveServer v1.0.0

[リリース日：2019/05/09]

変更点	詳細
リリース	初期リリース

法的通知

機密保持に関する声明

GPaymentsはこの文書に含まれる機密情報と知的財産に対するすべての権利を留保します。この文書には、GPaymentsの事業、商業、財務、または技術活動に関する情報を含まれる場合があります。第三者に対するこの情報の開示はGPaymentsに大きな不利益をもたらすため、この情報は受領者の単独使用を意図しています。この文書のいかなる部分も、書面による事前の許可なしに、いかなる形式または方法による複製、検索システムへの保存、または転送が禁じられています。この情報は、受領者との既存の秘密保持契約に基づいて提供されます。

著作権に関する声明

この文書で使用されているすべてのサードパーティ製品およびサービス名および口ゴは、それぞれの所有者の商号、サービスマーク、商標、または登録商標です。

この文書のスクリーンショットで使用されている企業、組織、製品、個人、およびイベントの例は架空のものです。実際の企業、組織、製品、個人、またはイベントとの関連は意図しておらず、推測されるべきではありません。

免責事項

GPayments Pty Ltdは、この文書に含まれる製品、プロトコル、または標準のいずれについても説明しておらず、説明を意図するものではありません。GPayments Pty Ltdは、いかなる目的のためにも、この情報の内容、完全性、正確性、または適切性を保証するものではありません。情報は明示的または暗黙的な保証なしに「現状のまま」提供され、予告なしで変更されることがあります。GPayments Pty Ltdは、商品適格性および特定の目的への適合性に関するすべての暗黙的な保証、および侵害に対する保証を含む、この情報に関するすべての保証を否認します。この文書に含まれる製品、プロトコル、または標準に関して、GPayments Pty Ltdが行った決定および/または声明は信頼されるべきではありません。

責任

いかなる場合においても、GPayments Pty Ltdは、GPaymentsがそのような損害の可能性について知らされていた場合であっても、契約上の行為、過失、またはその他の不正な行為において、この情報またはここに記載されている製品、プロトコル、標準の使用または使用不能に起因するか、それらと関連して発生するかにかかわらず、いかなる特別、付随的、間接的、または結果として起こる損害（事業利益の喪失、事業の中断、事業情報の喪失、またはその他の金銭的損失に対する損害を含みますが、これらに限定されません）についても責任を負いません。