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

ActiveServer最低システム仕様を下記に記載します。実際のシステム仕様はお客様のパフォーマンス要件に基づき変更してください。

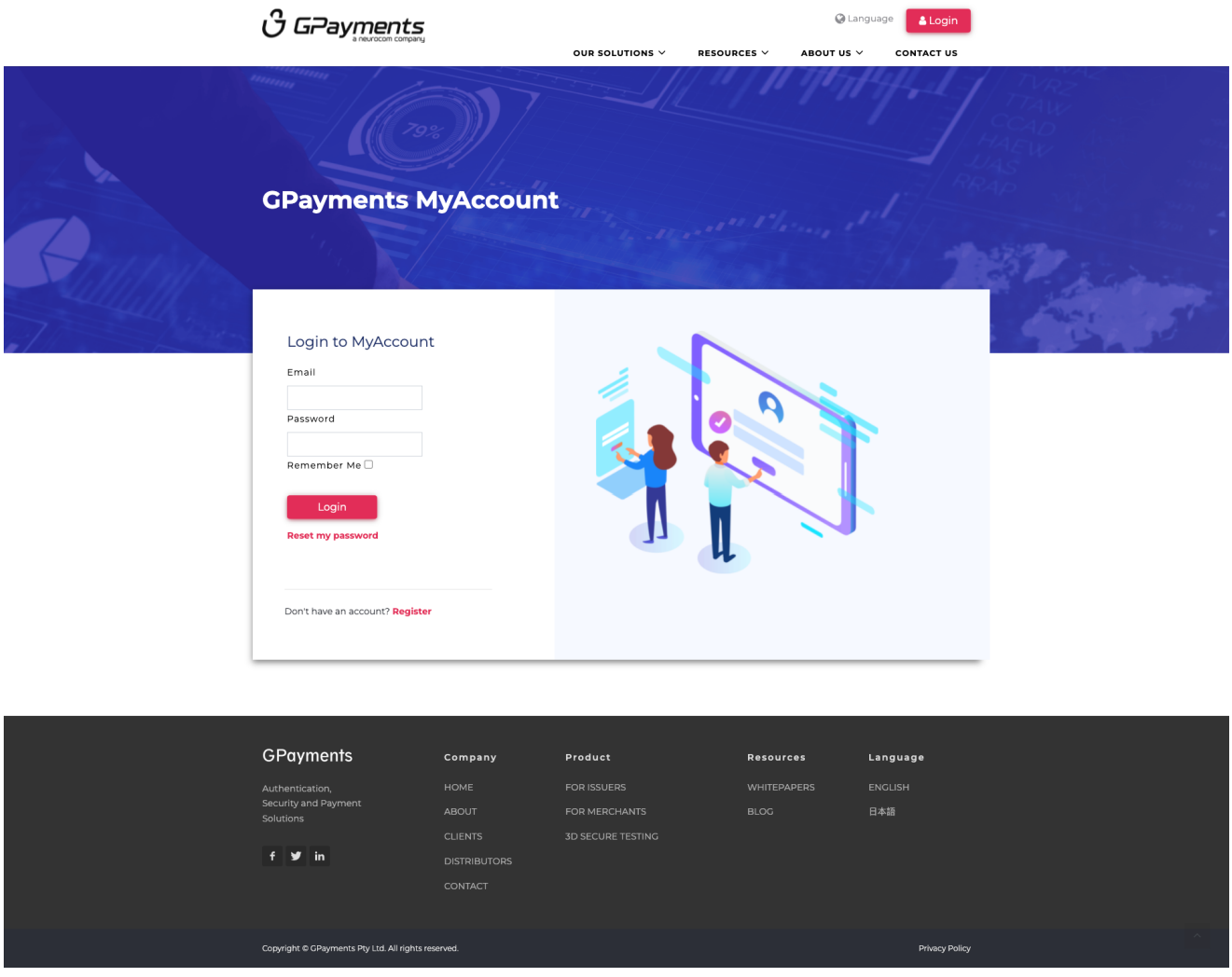
仕様	詳細
オペレーティング・システム(OS)	Linux、Windows Server
メモリ	4 GB RAM（最低）
CPU	2コア（最低）
ディスク容量	最低要件はありませんが、すべてのデータがデータベースに格納されるため、データベース用に十分なディスク容量を用意してください。
Java Development Kit	Java SE Development Kit 21.0.8
Javaコンテナ	`.jar` ファイルは、Servlet 2.4/JSP 2.0をサポートする任意のコンテナで実行できます。*デフォルト・コンテナは `UnderTow` です。*
Webブラウザー	Web管理インターフェイスは、Chrome、Firefox、またはEdgeブラウザーを使用してアクセスできます。

データベース： **ActiveServer**インスタンスとは別にデータベースサーバーを実行することをお勧めします。推奨されるデータベースサーバーの仕様とパフォーマンス要件については、データベースベンダーのドキュメントを参照してください。互換性のあるバージョンを以下に記載します。

データベース	互換バージョン
MySQL	8
Oracle	19c, 23
Microsoft SQL Server	2017, 2019
PostgreSQL	14, 15
IBM Db2	11.1 以降

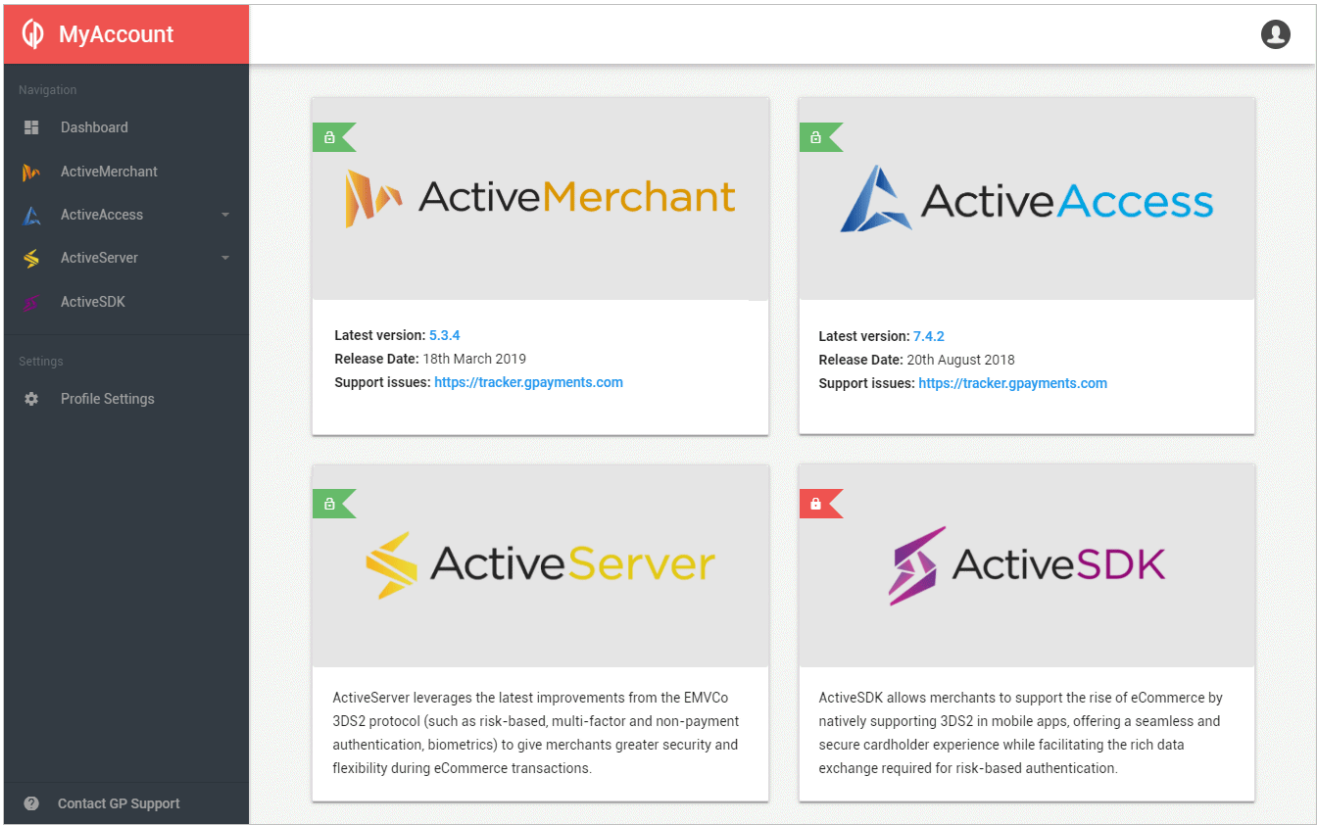
ActiveServerのダウンロード

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



!!! "アカウントをお持ちでない場合" アカウントをお持ちでない場合は、次のURLから登録してください。 https://login.gpayments.com/register

2. ログインすると、**MyAccountダッシュボード**が表示されます。



3. 左のメニューから**ActiveServer > Download**を選択して下さい。
4. **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
├── 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。
- `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5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
- 証明書有効期限通知設定

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
- Microsoft SQL Server
- PostgreSQL
- IBM Db2

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。
- `as.db.pool-size=`
  デフォルトでは、**ActiveServer**はプールサイズにHikariCPからの推奨サイズである`10`の値を使用します。プールサイズを増やす必要がある場合は、この値を使用してデフォルト値を上書きできます。パフォーマンスが安定するまで、プールサイズを少しずつ増やすことを推奨します。

MySQL

MySQL データベースを使用するには、下記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=<ユーザー名>
as.db.password=<パスワード>
as.db.password-plain=true
```

MySQL の max_allowed_packet

MySQL を使用している場合、 `Packet for query is too large` が発生する可能性があります。通常の操作では、 `max_allowed_packet` 値に 100M を使用することをお勧めします。

MySQL 8

MySQL 8 にて ActiveServer v2.0.8 またはそれ以前のバージョンを運用中のユーザーは、ActiveServer v2.0.9 以降へのアップグレード時に、as.db.url に `allowPublicKeyRetrieval=true` を付け足す必要があります。このパラメーターにはセキュリティ上の意味が含まれている場合があります。詳細につきましては、MySqlConnector をご参照下さい。

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=<ユーザー名>
as.db.password=<パスワード>
as.db.password-plain=true
```

Microsoft SQL Server

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>:<ポート番号>;databaseName=<DB名> または jdbc:sqlserver://<MS SQL DBホスト名を入力>\<インスタンス名>;databaseName=<DB名>;encrypt=false
as.db.username=<ユーザー名>
as.db.password=<パスワード>
as.db.password-plain=true
```

> **Microsoft SQL Serverの照合**
>
> Microsoft SQL Serverを使用する場合は、大文字と小文字を区別しないデフォルトのサーバー照合順序を維持してください。デフォルトの照合を使用しない場合はインストールが失敗します。SQL Server照合順序の使用を参照してください。

PostgreSQL

PostgreSQL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。

PostgreSQL データベース・プロパティー(例)

```
as.db.vendor=postgresql
as.db.url=jdbc:postgresql://<PostgreSQLホスト名を入力>:5432/<DB名を入力>
as.db.username=<ユーザー名>
as.db.password=<パスワード>
as.db.password-plain=true
```

IBM Db2

DB2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。

DB2 データベース・プロパティー(例)

```
as.db.vendor=db2
as.db.url=jdbc:db2://<DB2ホスト名を入力>:50000/<DB名>
as.db.username=<ユーザー名>
as.db.password=<パスワード>
as.db.password-plain=true
```

注意

VisaやMasterCardといった大手国際ブランドの場合、初期カードレンジデータは膨大なものになる可能性があります。例えばVisaの場合、カードレンジデータのサイズは40万行以上になり、トランザクションログのサイズが正しく設定されていないとDB2の性能劣化を招く可能性があります。GPaymentsでのテストでは、6コア/ 32Gサーバーで以下の設定を行うと、50万行のカードレンジデータで60秒未満の安定したinsert処理を行えることが判明しております:

- **logfilsiz**: 500000
- **logprimary**: 150
- **logsecond**: 100

なお、上記設定は参考情報です。実際の設定値はお使いのハードウェアやソフトウェアの仕様に準拠して設定してください。GPaymentsとしては、ログファイルのサイズとプライマリログファイルの数が上記の数以上であることをお勧めします。

AWS Secrets Managerを利用

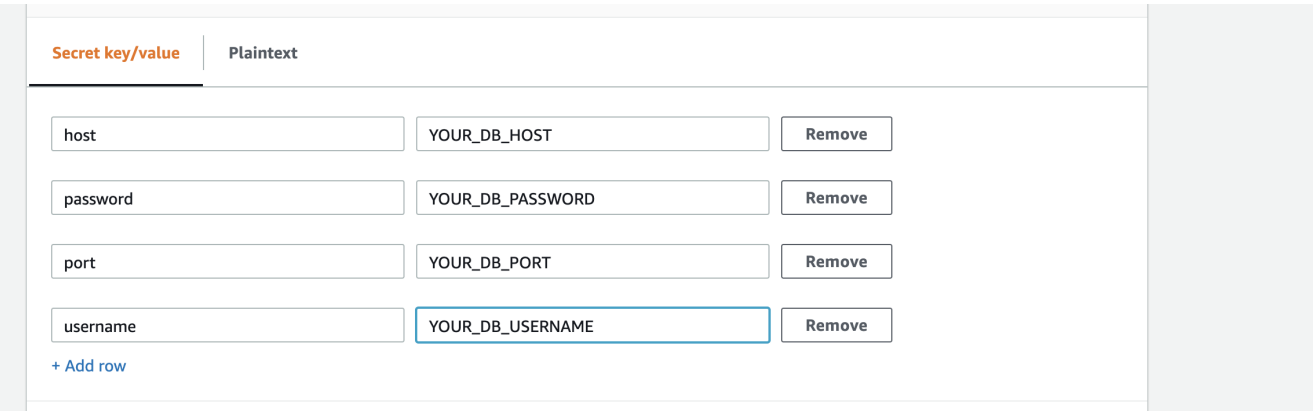
ActiveServerは、プロパティファイルの代わりにAWS Secrets Managerでのデータベース認証情報の保存をサポートしています。シークレットマネージャーにデータベースの認証情報を保存することで、ライフサイクルを通じてシークレットを簡単にローテーション、管理、および取得できます。

1. プロパティファイルで次のプロパティを設定します。

AWS Secrets Manager Properties (例)

```
as.db.url=<JDBC URL、「%s」を使用してホスト名とポートを置き換えます 例:
jdbc:postgresql://%s:%s/<Your DB name>
as.db.asm.region=<オプションのリージョンパラメーター、提供されていない場合デフォルトで
AWS認証情報のリージョンが使用されます。 例: us-west-1>
as.db.asm.secret-name=<AWSシークレットマネージャーのシークレット名>
```

2. 次のキー/値でAWSシークレットマネージャーを作成する:
   - `password` - データベースのパスワード
   - `username` - データベースのユーザーネーム
   - `host` - JDBC URLのホスト名。 `as.db.url` で最初に出現する「%s」はこの値に置き換えられます。
   - `port` - JDBC URLのポート部分。 `as.db.url` で2番目に出現する「%s」はこの値に置き換えられます。



注釈

AWSシークレットマネージャーを使用することで、 `as.db.username` 、 `as.db.password` 、および `as.db.password-plain` プロパティを省略できます。

アクセス権限

利用しているIAMが作成したシークレットへの**読み取り**権限があることを確認してください。資格情報の構成の詳細については、AWS認証情報を参照してください。

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。

デフォルトでは、 `サーバーポート` は、認証コールバックページおよび管理UIインターフェースのリクエストを含むすべてのウェブページリクエストに対応します。

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。

この設定を有効にすると、すべての管理UIインターフェイストラフィックが指定されたポートに制限されます。 `サーバーポート` は認証コールバックページにのみに対応します。

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
```

レート制限の構成

ログインページにおけるレート制限を管理するために、ActiveServer は以下の構成を提供します：


```
#default false
#as.security.limiter.enabled=false
#default is 1 request per second
#as.security.limiter.rate=1
```

許可ホスト名

ActiveServer をリバースプロキシサーバーの背後にデプロイする場合、この設定により、受け入れるホスト名を制限できます。

```
## 許可するホスト名のカンマ区切りリスト（大文字・小文字を区別しません）。
# as.settings.allowed-hosts=
```

Directory Serverポート設定

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(RReq/RRes)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。

> **備考**
>
> 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。

- 国際ブランドのDSへ接続するには `as.<Card Scheme>.port=` を設定、GPayments TestLabs DSに接続するには `as.testlab.<Card Scheme>.port=` を設定してください。

  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。

  > **備考**
  >
  > 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。

> **重要**
>
> ロードバランサーを使用してポートを転送する場合は、**ActiveServer**はチャレンジフロー中にDSが結果リクエストを送信するRReq URLを `https//<API URLまたは外部URL>:<ポート番号>` の形式に設定するので同じポート番号転送する必要があります。本番環境DSの場合、RReq URLは3DSサーバーURLから設定できるため、プロパティーで指定したポートとは違うポートを転送できます。

- 国際ブランドのDSは `as.<Card Scheme>.enabled=false` 、GPayments TestLabs DSは `as.testlab.<Card Scheme>.enabled=false` を設定できます。

  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

本番環境(例) TestLabs(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

本番環境(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
```

Discover / Diners Club International

本番環境(例) TestLabs(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

本番環境(例) TestLabs(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

本番環境(例) TestLabs(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

本番環境(例) TestLabs(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
```

eftpos

本番環境(例)

```
as.eftpos.port=9800
## 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.eftpos.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 - 利用可能な値は、 `local` 、 `s3` 、 `pkcs11` 、 `kms` です。

キーストアの種類

ローカル・キーストアは Sun JCE キーストアであり、暗号化キーはローカルファイルに格納されています。ActiveServer が暗号化を行うために使用するキーには主に３種類あり、これらはシステムにより自動的に生成されます。全てのキーストアファイルは `as.keystore.local.path` 内で指定されるディレクトリー内で生成されます。

1. **加盟店毎のデータキー** - 認証 API v1 取引では、ローカル・キーストアは加盟店毎に生成され、データキーは `as_<UUID>.jks` の形式でファイル内に格納されます。この加盟店により実行された取引は、キーがローテートされるまでは同じデータキーを用いて暗号化されます。キーの生成は、新しい加盟店の作成時に行われます。キーのエイリアスは、`AS_<UUID>` の形式で生成されます。
2. **マスターキー** - 認証 API v2 取引では、ActiveServer はデータがデータキーにより暗号化されてからデータキーがマスターキーにより暗号化されるエンベロープ暗号化を使用します。これは AWS KMS による CMK の管理に類似する方法です。マスターキーは `as_master_key.jks` ファイルに格納されています。キーのエイリアスは形式は `MASTER_<UUID>` となります。

PKCS11 と JCE のための任意のマスターキー構成

```
## default 365
# as.security.masterkey.validity=< days >
## default 20
# as.security.masterkey.cache.maxAge=< minutes >
## default 500
# as.security.masterkey.cache.messageLimit=< count >
```

1. **システムキー** - 暗号化システム関連のデータについては、データキーは `as_sys_<UUID>.jks` 形式で作成されたキーストアファイル内に格納されます。キーは最初のシステム起動時に生成されます。キーのエイリアスは `AS_SYS_<UUID>` 形式で生成されます。

> **バックアップが重要**
>
> マスターキーを紛失すると以前に暗号化したデータキーを復号化することができなくなるため、このマスター・キーストア・ファイルをバックアップすることが強く推奨されます。マスターキーをローテートした際には常に、ファイルのバックアップを作成することが推奨されます。

ローカル・キーストア（SunJCE）

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・キーストアとトア(SunJCE) (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。

> **警告**
>
> Windowsベースのマシンを使用している場合、キーストアフォルダーへのフルパスを設定するときにエスケープ文字（ `\` ）がおそらく必要になることに注意してください。
>
> 例： Windows共有フォルダがパス `\\ActiveServer\keystores` で使用されている場合、キーストアパスは次のように設定されます。 `as.keystore.local.path=\\\\ActiveServer\\keystores` 。

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。local keystore と同様、キーストアは Sun JCE キーストアであり、唯一の違いはキーストアファイルがローカルフォルダではなく S3 バケットに格納されるという点です。Amazon S3キーストアを使用するには、*AWS Bucket*、*AWS Region*、*AWS Credentials* 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。

   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>
   ```

アクセス権限

指定した鍵へのリードおよびライト権限があるIAMユーザーであることをご確認ください。認証情報の設定に関する詳しい情報はAWS認証情報をご参照ください。

以下のAWS認証情報をプロパティーから構成する事が可能です。(非推奨)

Amazon S3 keystore (例)

```
as.keystore.s3.credentials.access-key-id=<Your Amazon S3 access key ID>
as.keystore.s3.credentials.secret-access-key=<Your Amazon S3 secret access key>
```

> ⚠ **警告**
>
> この構成はActiveServer v1.3.0より古いバージョンを使用するお客様のために引き続き対応しますが、推奨できません。下記の他の認証情報設定方法を使用することを強く推奨します。

AWS Key Management Service (KMS)

設定

1. AWSのドキュメントに従って、KMSに対称カスタマーマスターキー(CMK)を作成します。
2. キーARNをAWSダッシュボードから `properties` ファイルへコピーします。

AWS KMS (例)

```
as.keystore.kms.key-arn=<Your AWS Key ARN> e.g. arn:aws:kms:us-east-1:123456789012:key/19c7b3dc-c49d-401f-bb97-f10bf3e116c9
```

アクセス権限

指定した鍵へのリードおよびライト権限があるIAMユーザーであることをご確認ください。認証情報の設定に関する詳しい情報はAWS認証情報をご参照ください。

> ⚠ **警告**
>
> AWS KMSは**API v2**のみに対応することにご留意ください。**API v1**で取引を実行すると**エラーコード1027**になります。

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。

マスターキーの生成前に HSM が具体的な属性を要求する場合に備え、ActiveServer では PKCS11 経由での HSM キー生成属性を構成するための設定をすることができます。

```
## a comma-separated list of PKCS11 CKA_XXX attributes to set for the Master
Key’s private key
# as.keystore.pkcs11.mk-private-key-gen-attrs.enabled=
## a comma-separated list of PKCS11 CKA_XXX attributes to unset for the Master
Key’s private key
# as.keystore.pkcs11.mk-private-key-gen-attrs.disabled=
## a comma-separated list of PKCS11 CKA_XXX attributes to set for the Master
Key’s public key
# as.keystore.pkcs11.mk-public-key-gen-attrs.enabled=
## a comma-separated list of PKCS11 CKA_XXX attributes to unset for the Master
Key’s public key
# as.keystore.pkcs11.mk-public-key-gen-attrs.disabled=
```

例：

```
## adds CKA_TOKEN = true and CKA_PRIVATE = true to the private key template
# as.keystore.pkcs11.mk-private-key-gen-attrs.enabled=CKA_TOKEN,CKA_PRIVATE

## adds CKA_TOKEN = false and CKA_PRIVATE = false to the public key template
# as.keystore.pkcs11.mk-public-key-gen-attrs.disabled=CKA_TOKEN,CKA_PRIVATE
```

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。

HSMトークンログインは必須です

キー管理にHSMを使用する場合、ASはHSMが*「常にトークンログインが必要 (Always require Token Login)」*の設定を有効にしている必要があります。通常、この設定はほとんどのHSMで自動的にオンに設定されますが、SafeNetなどのHSMでは、この設定はデフォルトでオンにならない場合があります。手順については、HSMのドキュメントを参照してください。

SafeNet HSMの場合、これは `パブリック暗号なし(No Public Crypto)` セキュリティフラグを設定することで実行できます。管理者は、提供されたコマンドラインユーティリティ `ctconf` を使用してフラグを設定できます。

AWS 認証情報

ActiveServerがAWSサービスへアクセスできるようにAWS認証情報を設定する必要があります。AWS認証情報には `access_key_id` と `secret_access_key` があります。AWS認証情報は以下の通り、いくつかの設定方法があります。

1. 下記と通りAWS認証情報を設定してください。

   AWS認証情報 (例)

   ```
   as.aws.credentials.access-key-id=<Your AWS access key ID>
   as.aws.credentials.secret-access-key=<Your AWS secret access key>
   ```

2. 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認証情報を設定してください。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` 。AWS認証情報プロファイル・ファイルには、以下の形式で行を含める必要があります。

```
[default]
aws_access_key_id = <Your Amazon S3 access key ID>
aws_secret_access_key = <Your Amazon S3 secret access key>
```

この方法を使用する場合、ActiveServer `properties` ファイルの `as.aws.credentials.access-key-id` と `as.aws.credentials.secret-access-key` パラメーターを使わずに空白のままにすることができます。

3. 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。

   この方法を使用する場合、ActiveServer `properties` ファイルの `as.aws.credentials.access-key-id` と `as.aws.credentials.secret-access-key` パラメーターを使わずに空白のままにすることができます。

E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サーバープロパティー (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メールサーバーのユーザーネーム>
as.mail.password=<Eメールサーバーのパスワード>
as.mail.auth=true
as.mail.start-tls=true
as.mail.from=<送信元のメールアドレス, e.g. admin@activeserver.com>
```

- `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=`
  電子メールサーバーのユーザーネーム。
- `as.mail.password=`
  電子メールサーバーのパスワード。
- `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。
- `as.mail.from=`
  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。

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。

証明書有効期限通知設定

有効期限が近づいている証明書について、**Business Admin**または**System Admin**ロールを持つユーザーに電子メール通知が送信されます。

この機能はデフォルトで有効になっており、プロパティファイルで設定できます。

> **Important**
>
> これらのプロパティを変更した後は、サーバーの再起動が必要です。

証明書通知

以下のプロパティは、**加盟店証明書およびマスター証明書**の有効期限に関する電子メール通知を制御します。

```
# 有効期限が近づいた加盟店証明書およびマスター証明書の電子メール通知を有効にする。デフォルト：true
as.notifications.cert.enabled=true

# 失効の何日前から通知の送信を開始するか
as.notifications.cert.cutoff=90

# 通知メール間の間隔（日数）
as.notifications.cert.windowDays=7
```

ディレクトリサーバー通知

以下のプロパティは、**ディレクトリサーバー（DS）クライアント、サーバーおよびCA証明書**の有効期限に関する電子メール通知を制御します。

```
# 有効期限が近づいたDSクライアント、サーバーおよびCA証明書の電子メール通知を有効にする。デフォルト：true
as.notifications.ds.enabled=true

# 失効の何日前から通知の送信を開始するか
as.notifications.ds.cutoff=90

# 通知メール間の間隔（日数）
as.notifications.ds.windowDays=7
```

上記の例では、90日以内に有効期限が切れる証明書のリストが記載された通知メールが、すべての**Business Admin**および**System Admin**ユーザーに7日ごとに送信されます。

フロントランナー／サイドカーを用いたデプロイメント

フロントランナー／サイドカーを用いたデプロイメントでは、**サイドカー**のみで証明書通知を有効にし、**フロントランナー**では無効にすることをお勧めします。

サイドカー（プロパティファイル）：

```
as.notifications.cert.enabled=true
as.notifications.ds.enabled=true
```

フロントランナー（プロパティファイル）：

```
as.notifications.cert.enabled=false
as.notifications.ds.enabled=false
```

ログ構成

デフォルトでは、**ActiveServer**はすべてのログファイルを `{AS_HOME}/logs/` フォルダーに出力します。もし、フォルダーが存在しない場合は自動的に作成されます。別のログフォルダーの場所を指定する場合は、コメントを外して、次の設定をログを主力したいパスに編集します。

```
# as.logging.path=<Your log file path>
```

マルチノードセットアップの場合、ファイルの命名規則の制約により、ノードごとに個別のフォルダーが必要です。

> **警告**
>
> この値を変更した場合、セットアップでエラーが発生すると、ログファイルが正しく記録されない場合があることに注意してください。パスが正しいこと、および**ActiveServer**インスタンスからアクセスでき、指定されたパスの読み取り/書き込み権限があることを確認してください。

ローカルファイルへのログ出力を無効にする

ローカルファイルへのログ出力が不要な場合、**ActiveServer**にはこの機能を無効にするオプションがあります。

ActiveServerのアプリケーションログをローカルファイルに保存するのを無効にするには、アプリケーションを起動する前に、 `startup.sh` （Linux）または `startup.bat` （Windows）ファイルの `AS_PROFILES` の最後にコマンド `disableFileLogs` を追加します。

例：

startup.sh startup.bat

```
export AS_PROFILES=prod,disableFileLogs
```

セキュリティーおよびアプリケーションのテクニカルサポートの要件から、この設定に関係なく、ログは引き続き標準のコンソール出力に送信されます。

JSON 形式

AS_LOGGING_FORMAT 環境変数を設定することで、JSON 形式のコンソールログを有効化します。このオプションが選択されている時には、ファイルログ保存は利用できません。

startup.sh startup.bat

```
export AS_LOGGING_FORMAT=json
```

取引ログへのリクエストのリンク

ActiveServer は、リバースプロキシサーバーによって提供される `x-request-id` ヘッダーを用いて、受信した要求を取引処理ログに紐付けます。

```
as.logging.requestIdHeader=x-request-id
```


Trans-type上書き設定

`trans-type` パラメーターは、DSプロファイルを利用して、Production Directory ServerとTestLabsを切り替えるために使用されます。 API呼び出しで指定する必要がないように、`trans-type` パラメータの機能を永続的に上書きする場合は、次のパラメータのコメントを外して、目的の値で編集する必要があります。

```
# as.auth.allowed-trans-type=all
```

- `as.auth.allowed-trans-type = testlab` が指定されている場合、TestLabsトランザクションのみが許可され、すべてのAPIリクエストがTestLabs DSプロファイルに転送されます。 API呼び出しの `trans-type` パラメータは、提示されても無視されます。
- `as.auth.allowed-trans-type = prod` が指定されている場合、本番トランザクションのみが許可され、すべてのAPIリクエストが本番DSプロファイルに転送されます。 API呼び出しの `trans-type` パラメータは、提示されても無視されます。
- `as.auth.allowed-trans-type` が提示されていないか、値が `null` / `all` の場合、オーバーライドは強制されません。 DSプロファイルガイドで説明されているように、クライアント側は「trans-type」を使用できます。

> **警告**
>
> **上書きが行われる旨ユーザーが通知されている場合、`as.auth.allowed-trans-type` は API 呼び出しの `trans-type` パラメーターを上書きし、認証要求を予期せぬ DS へと転送する可能性がありますので、ご注意ください。**

TLSバージョンの設定

ActiveServer は、HTTPSコネクターがデフォルトで TLS 1.2 のみを使用するよう指定します。プロトコルを無効化させるには、下記のように Java セキュリティーファイルを直接編集することで回避策を講じることができます。

> **警告**
>
> **`java.security` ファイルを編集すると、サーバー上の全ての Java アプリケーションに影響が及びますのでご注意ください。GPayments は、この変更により生じる可能性のある問題について一切の責任を負いかねます。**

特定のプロトコルを無効にするには、 `<jdk directory>/jre/lib/security` にある `java.security` ファイルを編集し、 `jdk.tls.disabledAlgorithms` エントリを更新します。 元のエントリは次のようになります。

```
jdk.tls.disabledAlgorithms=SSLv3, RC4, DES, MD5withRSA, DH keySize < 1024, \
    EC keySize < 224, 3DES_EDE_CBC, anon, NULL
```

`SSLv2Hello、TLSv1、TLSv1.1` のように、デフォルトでは含まれていない脆弱と見なされるプロトコルを無効にするには、これらの値を以下のようにエントリに追加します。

```
jdk.tls.disabledAlgorithms=SSLv3, RC4, DES, MD5withRSA, DH keySize < 1024, \
    EC keySize < 224, 3DES_EDE_CBC, anon, NULL,SSLv2Hello, TLSv1, TLSv1.1
```

> **重要**
>
> 提案されている無効にするプロトコルはあくまで提案値です。**ActiveServer**のセキュリティ構成を決定する際には、貴社のセキュリティチームにご相談ください。

`java.security` ファイルを編集したら、実行中の**ActiveServer**インスタンスを再起動して、変更を有効にします。 Linux端末で次のコマンドを実行することにより、有効になっているプロトコルを確認できます。

```
nmap --script +ssl-enum-ciphers -p 7443 127.0.0.1
```

3DSサーバー参照番号の上書き

下位互換性のために、**ActiveServer**はv2.1.0の認定中にEMVCoによって発行された3DSサーバー参照番号をデフォルトで使用します。デフォルトの参照番号はv2.1.0要求の送信時にのみ有効で

あり、この参照番号を使用するv2.2.0要求は国際ブランドのディレクトリサーバーによって拒否される可能性があります。AReqで送信される3DSサーバー参照番号を更新するには、2つの方法があります。

1. 国際ブランド毎に3DSサーバー参照番号を上書きする
2. すべての国際ブランドの3DSサーバー参照番号を上書きする

いずれかの方法を使用し、v2.2.0プロトコルの使用を開始する準備ができたら、3DS サーバー参照番号を GPayments 発行の EMVCo v2.2.0 準拠の参照番号である `3DS_LOA_SER_GPPL_020301_01060` に設定してください。

> **⚠ コンプライアンステスト後にのみ更新してください**
>
> **各国際ブランドのコンプライアンステストを完了した後にのみ 3DS サーバー参照番号を更新してください。**準拠していない場合、DS は V2.2.0 3DS サーバー参照番号を拒否する可能性があります。

国際ブランド毎に3DSサーバー参照番号を上書き

次の構成を**すべてのアプリケーションノード**の**application-prod.properties**ファイルに追加することにより、各国際ブランドに送信される3DSサーバー参照番号を上書きできます。

> **⚠ すべてのアプリケーションプロパティファイルを更新する必要があります**
>
> これらの設定はノードごとであるため、正しい参照番号が送信されるようにするには、すべてのアプリケーションファイルを更新する必要があります。

AMEX Discover JCB Mastercard Union Pay Visa eftpos

```
## AmericanExpressに使用される3DSサーバー参照番号を上書きします
as.prod.server-ref-number.amex=Reference Number
```

> **コンプライアンステスト中のオーバーライド**
>
> 一部の国際ブランドでは、プロバイダーがテスト目的で発行した特定の3DSサーバー参照番号を使用する必要があるため、このオーバーライド機能は、国際ブランドのコンプライアンステスト中にも使用できます。

3DSサーバー参照番号をグローバルに上書きする

このグローバルオーバーライドは、すべての国際ブランドの構成を個別に更新したくない場合に役に立ちます。指定すると、すべての国際ブランド構成が上書きされます。

```
## すべての国際ブランドに使用される3DSサーバー参照番号を上書きします
as.prod.server-ref-number-overriding=Reference Number
```

> **注意**
>
> グローバルオーバーライドを使用する場合は、すべての国際ブランドのコンプライアンステストが完了していることを確認してください。完了していない場合、一部のディレクトリサーバーが要求を拒否する可能性があります。

PReqメッセージのバージョン番号を上書きする

ActiveServerv2.0.0は、Visa、Mastercard、American Expressのメッセージバージョン2.2.0でPReqを送信します。これらのディレクトリサーバーは、デフォルトでV2.2.0のバージョンを処理することが可能なためです。JCBとDiscoverでは、2.2.0 PReqを送信する前に、国際ブランドのコンプライアンステストと3DSサーバー参照番号の登録を完了する必要があるため、これらの国際ブランドはデフォルトで2.1.0PReqを送信します。

国際ブランドコンプライアンステスト中に、2.1.0または2.2.0PReqメッセージのいずれかを送信する必要がある場合があります。また、**コンプライアンステスト後**に本番インスタンスで2.2.0のPReqを使用するようにすべての国際ブランドを更新する必要がありますが、これをサポートするために、PReqメッセージの上書き設定が `application-prod.properties` ファイルに追加されました。

```
as.prod.preq-message-version-overriding.<card scheme>=<message version>
```

- **Message version:** `2.2.0` または `2.1.0`
- **Card scheme:** `amex` 、 `discover` 、 `jcb` 、 `mastercard` 、 `unionpay` または `visa`

E.g. `as.prod.preq-message-version-overriding.jcb=2.1.0`

> **すべてのアプリケーションプロパティファイルを更新する必要があります**
>
> これらの設定はノードごとであるため、PReqメッセージのバージョンを正しく送信されるようにするには、すべてのノードのアプリケーション構成を更新する必要があります。

デプロイ方法

ActiveServerは以下3つの方法でデプロイ可能です。

1. スタンドアローン(デフォルト) - 全プロセスを処理するノード。
2. フロントランナー - 取引処理とUI機能のみ処理するノード。
3. サイドカー - カードレンジ更新など高負荷なバックグラウンドプロセスのみ処理するノード。

フロントランナー／サイドカーを用いたデプロイメント

ActiveServerのフロントランナー・サイドカーモードを使う際は、両方を用意しなければ正常に機能しません。モードごとに別個のサーバーを用意する必要があり、トラフィックはフロントランナーにのみ集中するようにします。サイドカーはAPI呼び出しを受信するロードバランサーに登録しないでください。

ActiveServerの `sidecar` インスタンスを起動するための新規サーバーを用意してください。また `AS_HOME` ディレクトリは既存のものをコピーし、ローカルキーストア（または HSM）がアクセス可能であることを確認してください。

取引処理を行うノードの `application.properties` に `as.settings.runtime-type=frontrunner` を新規追加設定してください。環境変数をお使いの場合は、`AS_SETTINGS_RUNTIME_TYPE=frontrunner` をシステム設定に追加してください。取引処理を行うノードを更新したあと、起動時のバナーの `Runtime Type` からそのノードが `Frontrunner` モードで稼働していることを確認してください。

新規 `sidecar` サーバーに `as.settings.runtime-type=sidecar` を新規追加設定してください。

GPaymentsではディレクトリサーバを用いてカードレンジ更新をテストしており、4GBのメモリがあれば主要国際ブランドのカードレンジ更新を処理できることを確認しております。

そのためGPaymentsとしてはインスタンスに最低4GB以上のメモリを割り当てるよう推奨しておりますが、理想値として8GBのメモリ割り当てがあれば件数が増加したカードレンジの全件更新も確実に対応できます。

下記のJVMパラメータを既存のパラメータに含めてください。(メモリ割り当てが8GBの場合を想定した設定です。お使いのメモリに準じて変更してください。)

```
-XX:MaxDirectMemorySize=512m -XX:+ExitOnOutOfMemoryError -Xmx5664877K -
XX:MaxMetaspaceSize=227218K -XX:ReservedCodeCacheSize=240M -Xss1M
```

`sidecar` ノードを更新すると、全バックグラウンドプロセスが同ノードで処理されていることがログコンソールからわかります。同時にフロントランナーノードからはバックグラウンドプロセスがスキップされているように見えます。

Second Level Card Range Cache (SLCRC)

この機能は、サイドカーノードにキャッシュを作って格納し、カードレンジクエリの際にデータベースへのアクセスを効果的にスキップするために導入されました。

SLCRC の構成では、フロントランナーがカードレンジクエリのクライアントであり、サイドカーは SLCRC プロバイダーです。フロントランナーはサイドカーの SLCRC を探し、このクエリが失敗した場合にはデータベースへとフォールバックします。

サイドカーのプロパティファイルにて下記の構成を追加して下さい。

- `as.slcrc.enableProvider=true|false`：機能を有効化または無効化します。
- `as.slcrc.provider.port`：キャッシュが到達可能なポートです。
- `as.slcrc.token`：要求とともに送信されるトークンです。

フロントランナーのプロパティファイルにて下記の構成を追加して下さい。

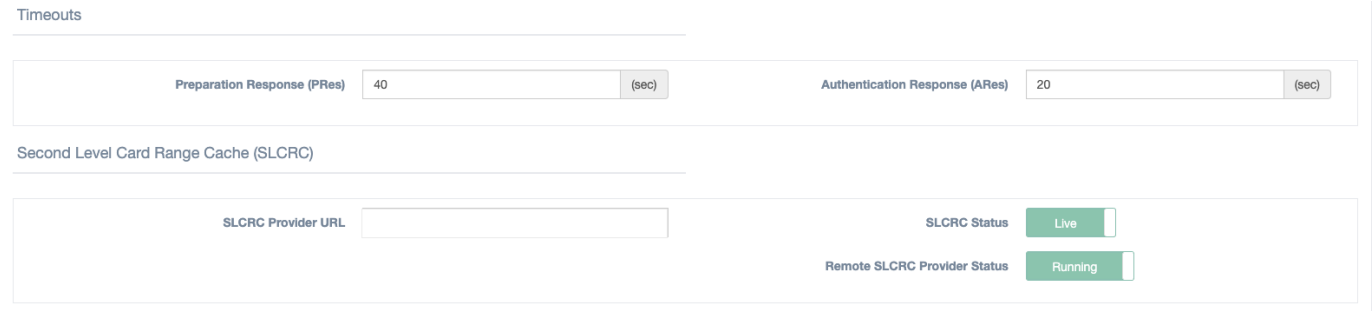
- `as.slcrc.url`：形式は `https://<sidecar server internal host name or IP>:<the slcrc port>` です。
- `as.slcrc.token`：全ての要求とともに送信されるトークンであり、サイドカーのプロパティファイル内の一つのセットと合致している必要があります。

SLCRC 有効時に最適な性能を引き出すためには、サイドカーノードが 16GB（最低でも 8GB）のメモリを持つことが推奨されます。

テストモード

フロントランナーノードには、SLCRC のクライアント構成として `as.slcrc.clientTestMode=true|false` を使用し SLCRC 構成が機能しているかテストするこ

とができます。テストモードを `true` に設定し、インスタンスを表示させ、管理 UI の DS プロファイルページにて導通確認を実施していただくことを GPayments は推奨します。



テストモードでは、SLCRC の導通確認が失敗しても、フロントランナーは起動に失敗しません。管理 UI は SLCRC プロバイダーの状態を示します。管理 UI 内の **Card range** クエリページにて疑似カード番号を使用し SLCRC カードレンジクエリをテストすることも可能です。

ライブモード

ライブモードでは、`as.slcrc.clientTestMode` が `false` に設定されます。このモードでは、SLCRC プロバイダーが到達不可の場合には ActiveServer インスタンスは起動しません。

クエリ処理

SLCRC が有効な場合、デフォルト設定として ActiveServer は SLCRC によるカードレンジ検索を行います。このクエリに失敗すると、元のデータベースクエリへとフォールバックします。

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。

> **備考**
>
> 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。

JVMメモリー上書き

ActiveServerのパフォーマンスを向上させるために、JVMに割り当てられたメモリのレベルを調整する必要がある場合があります。 JVMメモリの調整はこのガイドの範囲外ですが、以下のコマンドを使用して、JVMが**ActiveServer**を実行するために使用するメモリの正確な量を指定できます。 **`Xmx"メモリーサイズ"`** コマンドを**ActiveServer**起動スクリプトに追加できます。

```
java -Xmx3600m -cp . -Djava.security.egd=file:/dev/./urandom -jar ls *.jar
```

上記の設定では、使用可能なメモリの¼のデフォルトの割り当てではなく、3600MBのメモリが使用されるように割り当てられます。

アウトバウンドプロキシの設定

アウトバウンドプロキシサーバをご利用いただく際は、 `startup.sh` を下記の通り修正してください。:

```
java -cp . -Djava.security.egd=file:/dev/./urandom -Dhttp.proxyHost=<host> -
Dhttp.proxyPort=<proxy port> -Dhttps.proxyHost=<host> -Dhttps.proxyPort=<proxy
port> -jar as.jar -Dhttps.proxyUser=myuser -Dhttps.proxyPassword=mypass
```

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。

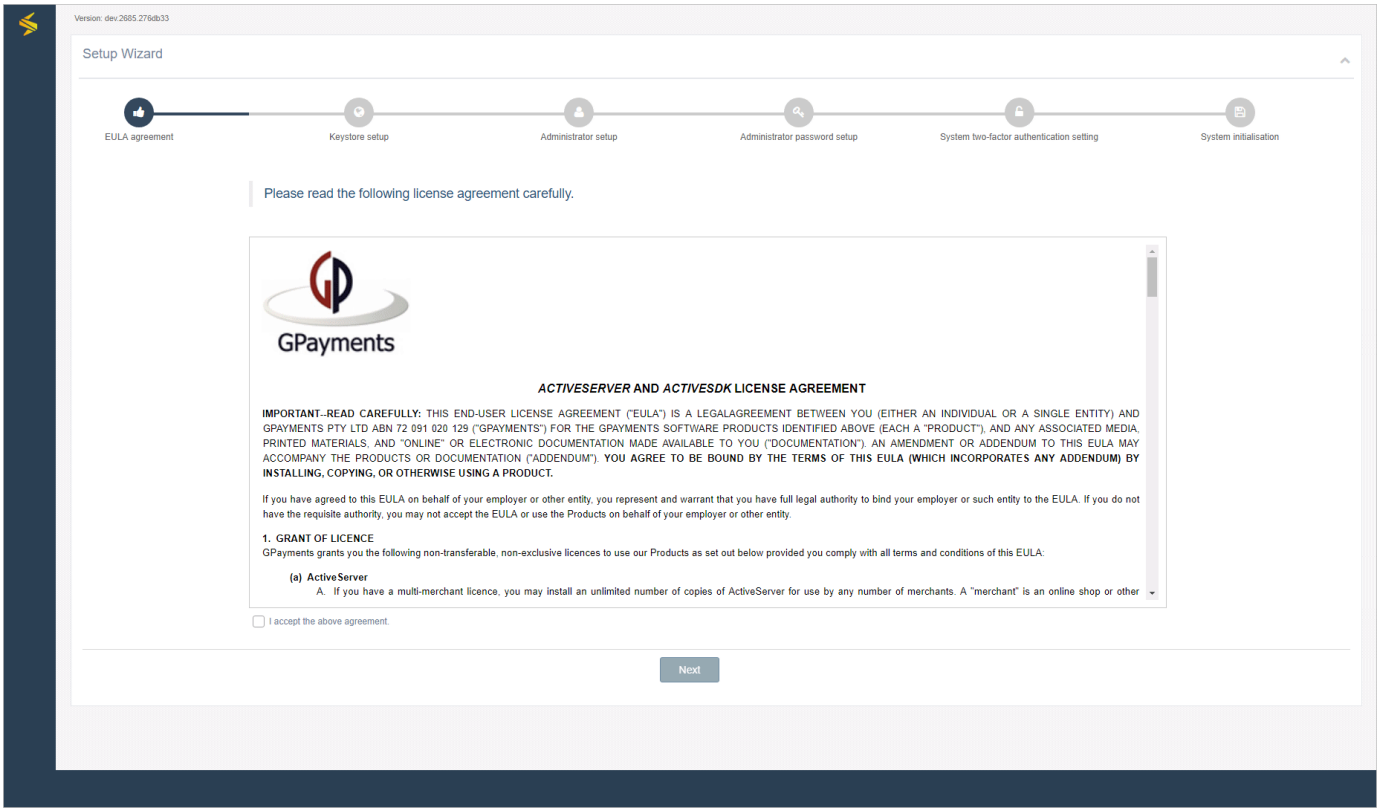
初めてActiveServerを実行する場合、セットアップ・ウィザードが表示され、セットアップ・プロセスがガイドされます。

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

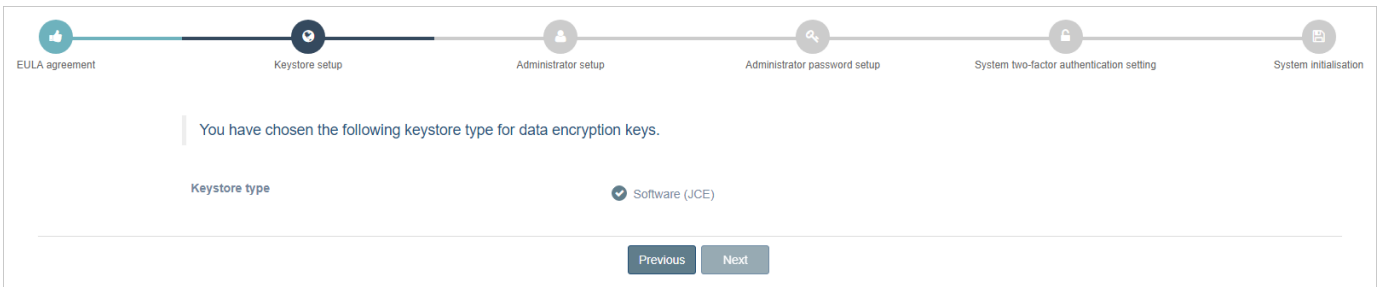
- システム2要素認証設定
- システム初期化

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



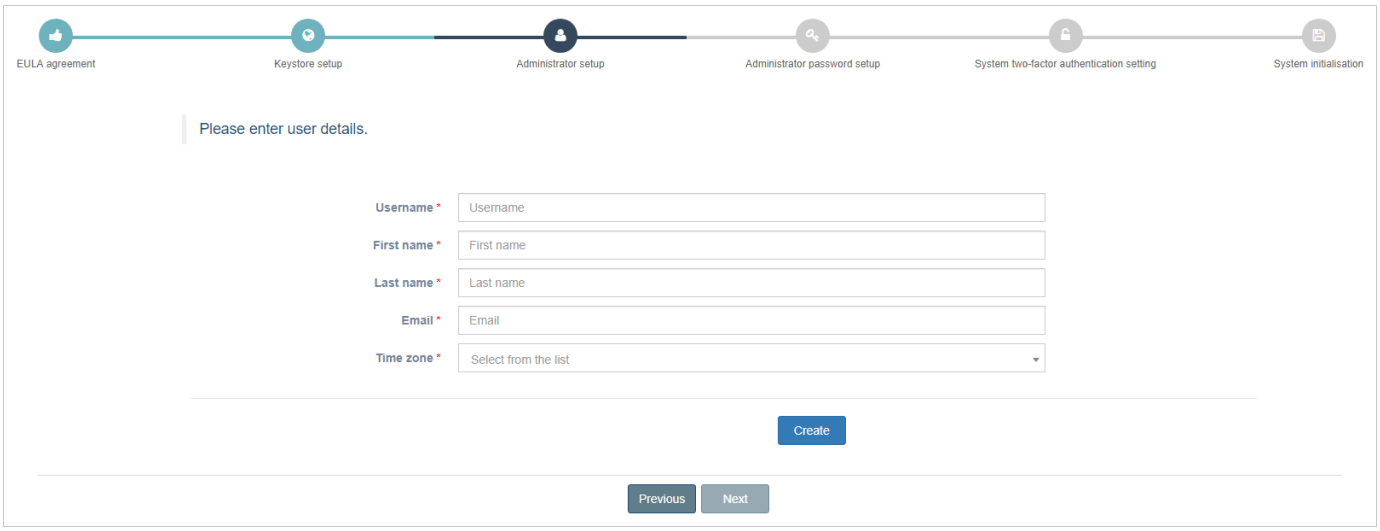
- **Keystore type**を選択します。

  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。

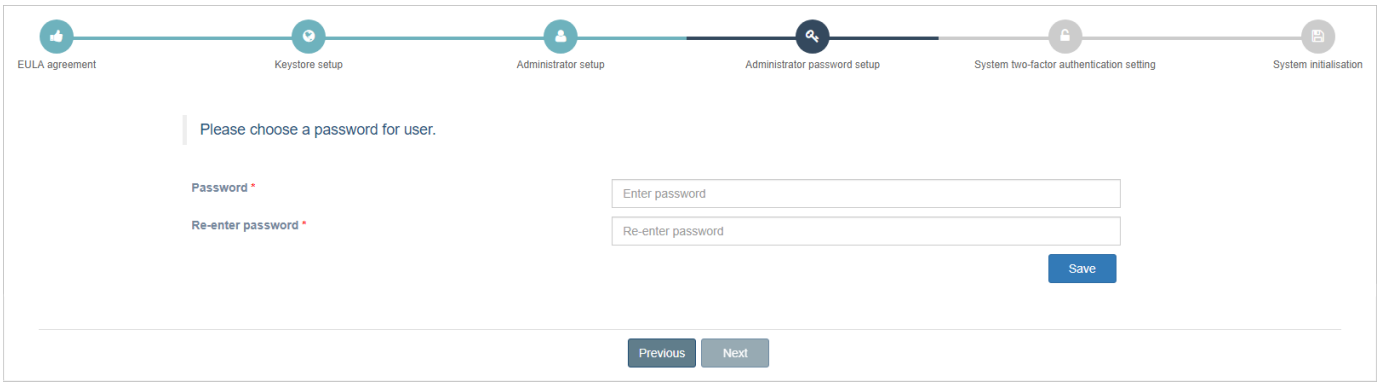
- 続行するには、***次***ボタンを選択します。

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。

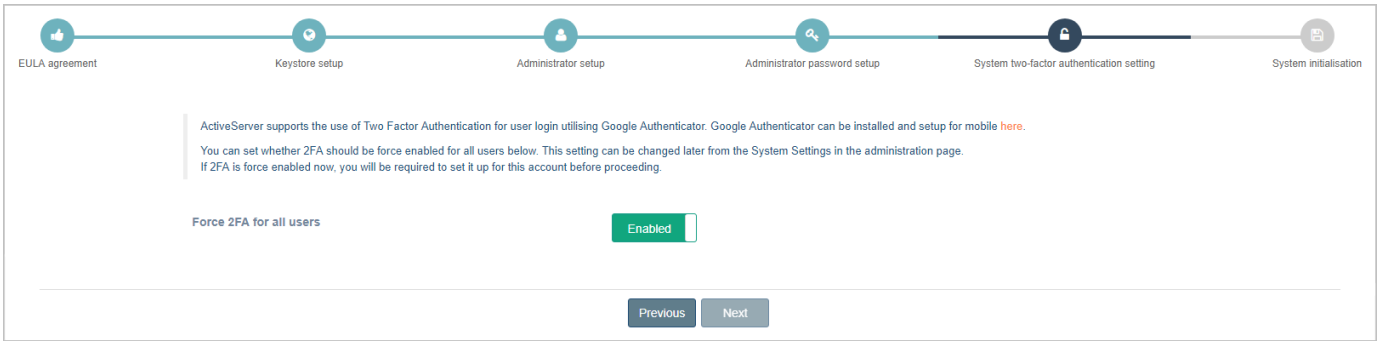
管理者パスワード・セットアップ



- 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。

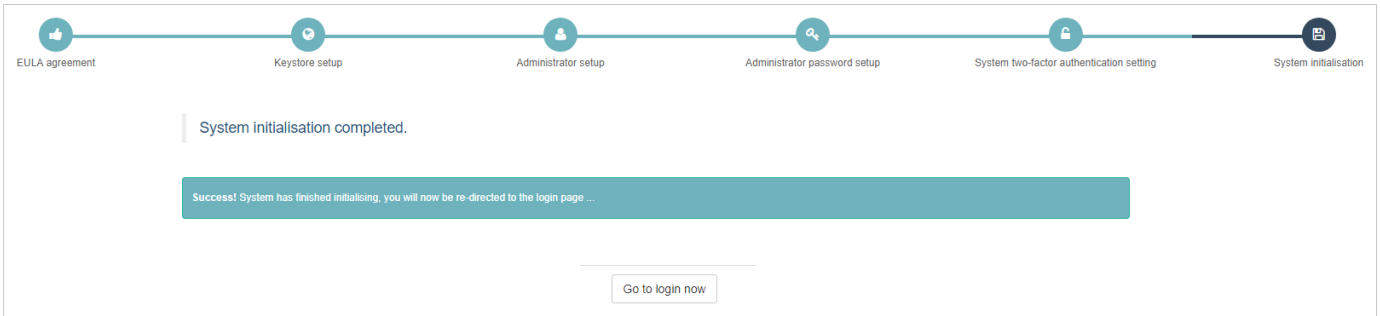
> **備考**
>
> この機能を使用するには、モバイル・デバイスにGoogle認証システムがインストールされている必要があります。
>
> Google Authenticatorのセットアップ手順については、Google Authenticatorをインストールを参照してください。

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。

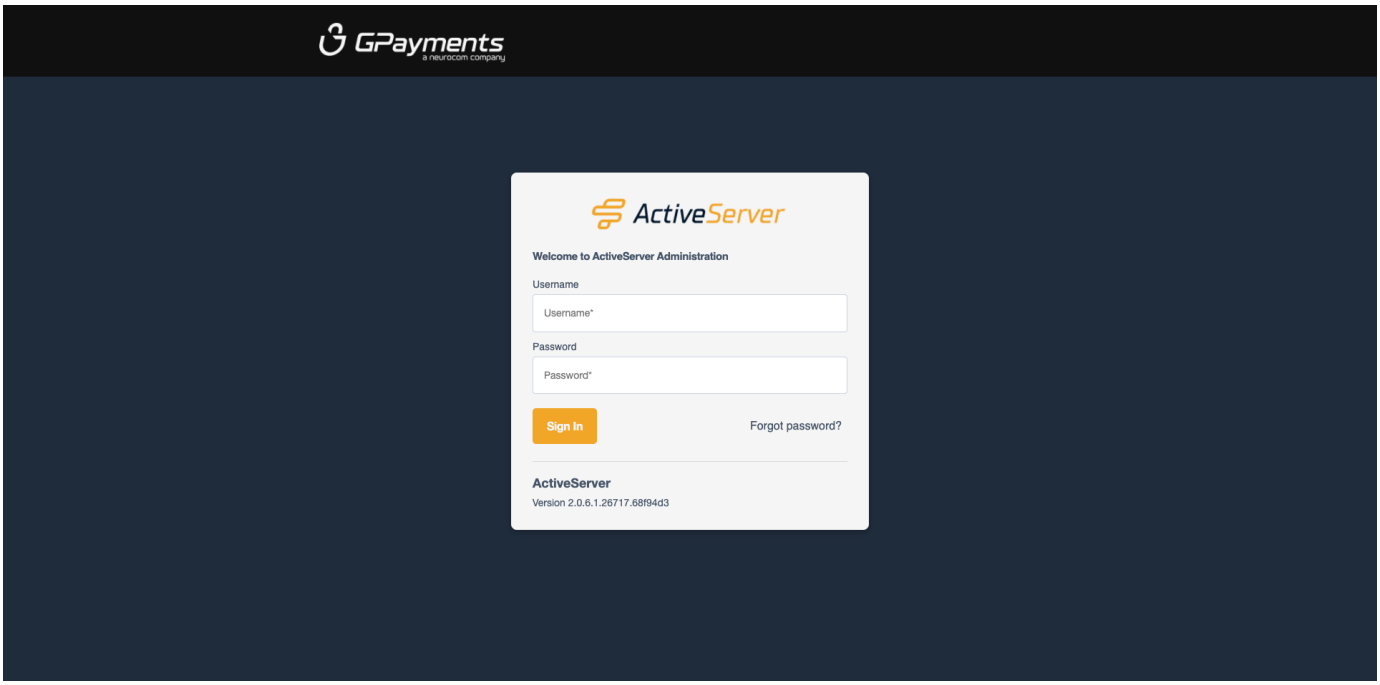
すべてのユーザーに2要素認証を強制するには：

- **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