Quickstart

Quickstart is intended to guide you smoothly through downloading, setting up, and running ActiveServer. For specific user guides on how to configure and manage ActiveServer, refer to the **Guides** section, two menu items below this one.

Prerequisites

ActiveServer minimum specifications, actual specifications should be based on your performance requirements:

Specifications	Details
Operating System	Linux, Windows Server
Memory	4 GB RAM (minimum)
CPU	2 Cores (minimum)
Disk Space	No minimum requirements, but ensure that sufficient disk space is available for logging
Java Development Kit	Java SE Development Kit 8 (Open JDK v1.8)
Java Container	The `.jar` file can run in any container that supports Servlet 2.4/JSP 2.0 or later. *Default container is* `UnderTow`.
Web Browser	The Administration UI can be accessed using Google Chrome, Mozilla Firefox, or Microsoft Edge.

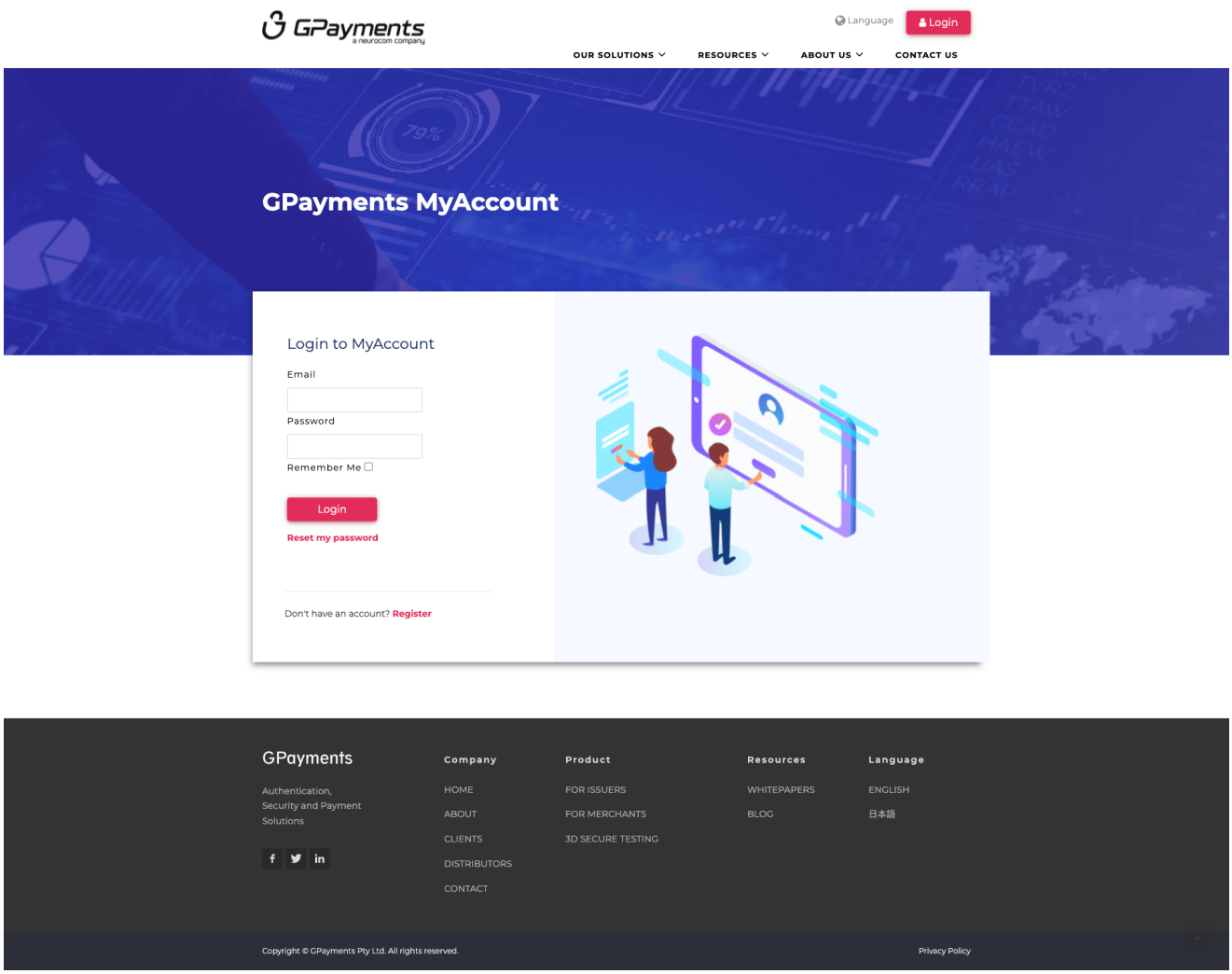
Database:

It is recommended to run the database server separate from the **ActiveServer** instance, consult your database vendor documentation for recommended database server specifications as well as performance requirements. Compatible versions are listed below:

Database	Compatible Versions
MySQL	5.7, 8
Oracle	12.2.0.1, 19c
Microsoft SQL Server	2012, 2014, 2017, 2019
PostgreSQL	10 - 14
IBM Db2	11.1 and later

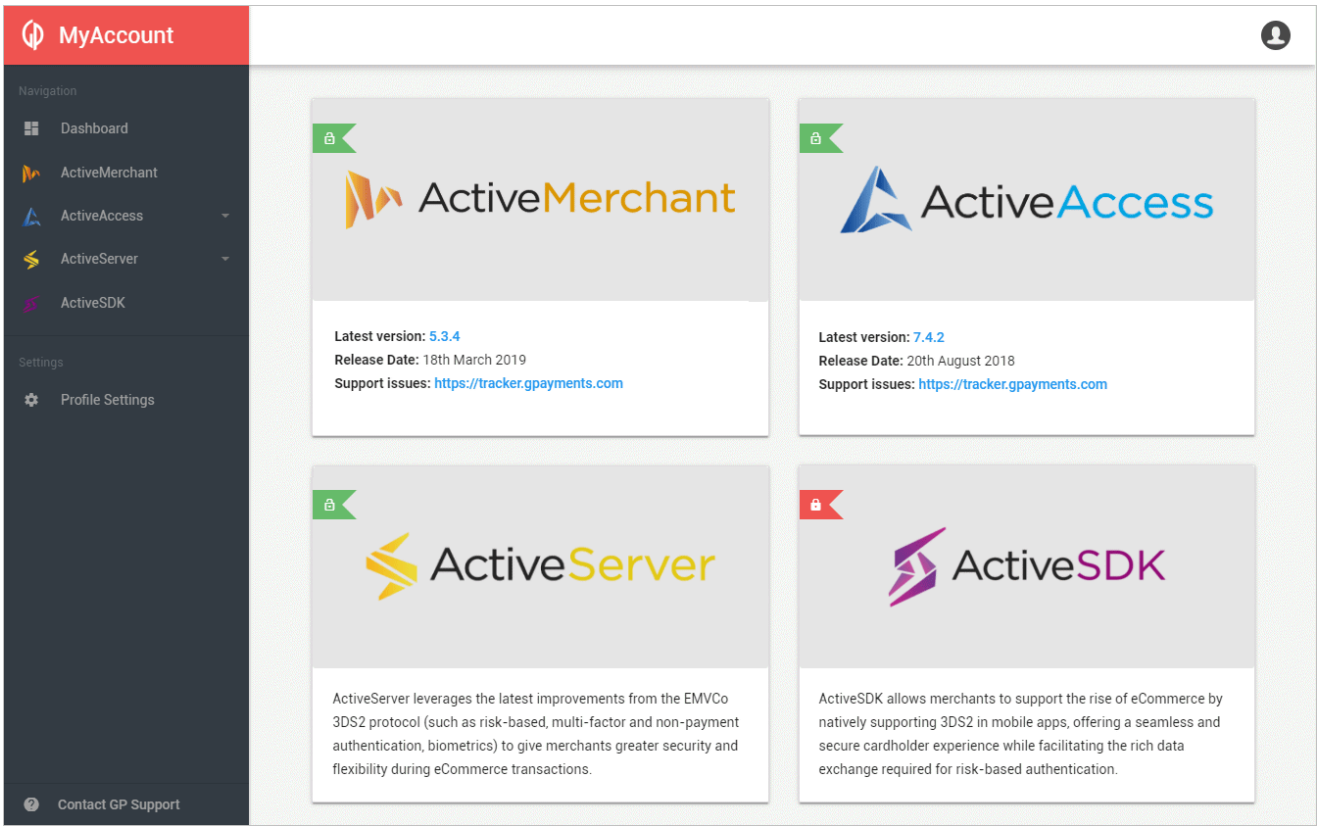
Download ActiveServer

1. Login to GPayments **MyAccount** at https://login.gpayments.com/login.



> **If you do not have an account...**
>
> If you do not have a GPayments MyAccount, register at: https://login.gpayments.com/register

2. Once logged in, you will see the **MyAccount Dashboard**.



3. From the Navigation Menu on the left, select **ActiveServer > Download**.
4. Select the release package to start the download.

MyAccount organisations

MyAccount utilises an organisation structure for clients to share product privileges between company users, such as downloading the software, and managing instance activation and licensing.

When downloading the software, if you are not part of an organisation yet, you will be prompted to either **Register** an organisation, or ask your point of contact with GPayments to invite you to your existing organisation. Inviting a user to an organisation can be done from the **MyAccount > Profile Settings > My Organisation** section.

> **Important**
>
> **To access your purchased software and manage existing instances for your organisation, confirm your company does not already have an organisation setup before creating a new one.**

Once part of an organisation that has already purchased **ActiveServer**, you should be able to download the package. If you are unable to download the software, but have confirmed you are in the correct organisation that has already purchased the product, please contact GPayments Tech Support at techsupport@gpayments.com for assistance. Or, if you would like to talk to our friendly sales team about purchasing **ActiveServer**, you can contact GPayments Sales at sales@gpayments.com.

Installation

Extract the downloaded `.zip` file, and you should see the files below.

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

The files are:

- `application-prod.properties` - Configuration file to initialize ActiveServer.
- `as.jar` - The main ActiveServer Java package.
- `README.txt` - General information about ActiveServer, as well as its documentation, licensing, and support.
- `release.txt` - Release notes for all versions of ActiveServer.
- `startup.bat` - The startup script for Windows.
- `startup.sh` - The startup script for Linux.

Configuration

Configure the system properties for ActiveServer by editing the `application-prod.properties` file.

> ⚠ **You can startup ActiveServer without configuring system properties, but...**
>
> The `application-prod.properties` file in the package you have downloaded includes default application properties. If you start up an instance of ActiveServer using these default application properties, ActiveServer will:
>
> - Use a default database that will only be temporarily stored in the RAM, and will be wiped once the ActiveServer instance is shut down.
> - Create a keystore file with SunJCE that is stored locally in `${AS_HOME}/conf/security/`.
> - Skip email server configurations, therefore disabling the system from sending email notifications to users.
>
> Using the default properties may be useful for trying out the software and its interface as it allows you to quickly startup an instance of ActiveServer without having to change any settings. However, you must configure the system properties before setting up a production instance.

To change the default application properties:

Open the file `application-prod.properties` and change the corresponding values associated with each relevant parameter.

There are 4 categories of system properties:

- Database settings
- Web server settings
- Keystore settings
- Email server settings

Database settings

ActiveServer supports the following databases:

- MySQL
- Oracle

- Microsoft SQL Server
- PostgreSQL
- IBM Db2

Each database has the following set of configurable properties:

- `as.db.vendor=`
  Database vendor/type. Default value is empty, where an in-memory test database will be used. You cannot enter production with the in-memory test database. Possible values are `mysql`, `oracle` or `sqlserver`.
- `as.db.url=`
  The database connection URL used to connect to your database. The URL must be in JDBC format.
- `as.db.username=`
  Database username. Enter the username set by your database administrator.
- `as.db.password=`
  Password. Enter the password set by your database administrator.
- `as.db.password-plain=`
  Whether to encrypt the password above or not. ActiveServer can encrypt the password when storing it in the `application-prod.properties` file. To enable password encryption, enter `false`. To leave the password as plain text, enter `true`.
- `as.db.pool-size=`
  By default, **ActiveServer** uses a value of 10 for the pool size, which is the recommendation from HikariCP. If the pool size needs to be increased, this value can be used to override the default value. It is recommended to increase the pool size in small increments until performance is stabilised.

MySQL

To use a MySQL database, you will need the following properties:

MySQL Database Properties (Example)

```
as.db.vendor=mysql
as.db.url=jdbc:mysql://<Your My SQL DB Host>:3306/<Your DB Name>?
useUnicode=true&characterEncoding=utf8&useSSL=false
as.db.username=<user name>
as.db.password=<password>
as.db.password-plain=true
```

max_allowed_packet for MySQL

When using MySQL, the `Packet for query is too large` may occur. For normal operations, it is recommended to use 100M for the `max_allowed_packet` value.

MySQL 8

Users running ActiveServer v2.0.8 or earlier on MySQL 8 should append `allowPublicKeyRetrieval=true` to as.db.url, when upgrading to ActiveServer v2.0.9 or higher. This parameter may have a security implication, please refer to MySqlConnector for more information.

Oracle

To use a Oracle database, you will need the following properties:

Oracle Database Properties (Example)

```
as.db.vendor=oracle
as.db.url=jdbc:oracle:thin:@//<Your Oracle DB Host>:1521/<Your Oracle DB Name>
as.db.username=<user name>
as.db.password=<password>
as.db.password-plain=true
```

Microsoft SQL Server

To use a Microsoft SQL Server database, you will need the following properties:

MS SQL Database Properties (Example)

```
as.db.vendor=sqlserver
as.db.url=jdbc:sqlserver://<Your MSSQL DB Host>:<Your Port
Name>;databaseName=<Your DB Name> or jdbc:sqlserver://<Your MSSQL DB
Host>\<Your Instance Name>;databaseName=<Your DB Name>
as.db.username=<user name>
as.db.password=<password>
as.db.password-plain=true
```

Collation for Microsoft SQL Server

When using Microsoft SQL Server, please keep the default server collation, which is case-insensitive, otherwise the installation will fail. Read Using SQL Server Collations.

PostgreSQL

To use a PostgreSQL database, you will need the following properties:

PostgreSQL Database Properties (Example)

```
as.db.vendor=postgresql
as.db.url=jdbc:postgresql://<Your PostgreSQL DB Host>:5432/<Your DB name>
as.db.username=<user name>
as.db.password=<password>
as.db.password-plain=true
```

IBM Db2

To use an IBM Db2 database, you will need the following properties:

DB2 Database Properties (Example)

```
as.db.vendor=db2
as.db.url=jdbc:db2://<Your DB2 host>:50000/<Your DB name>
as.db.username=<user name>
as.db.password=<password>
as.db.password-plain=true
```

Warning

For major card schemes like Visa or MasterCard, the initial card range data could be very big. E.g. for Visa, the size of the card range data can be 400K+ rows, which could cause performance issues in DB2 if the transaction logs sizes are not properly configured. In GPayments tests, we found the following settings on a 6 core/32G server produces stable less-than-60-seconds insert results on a 500K card range data set:

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

Note this setting is for reference only, the actual number can vary based on your hardware and software specifications. GPayments suggests the log file size and the primary log file number to be greater or equal to the number mentioned above.

Using the AWS Secret manager

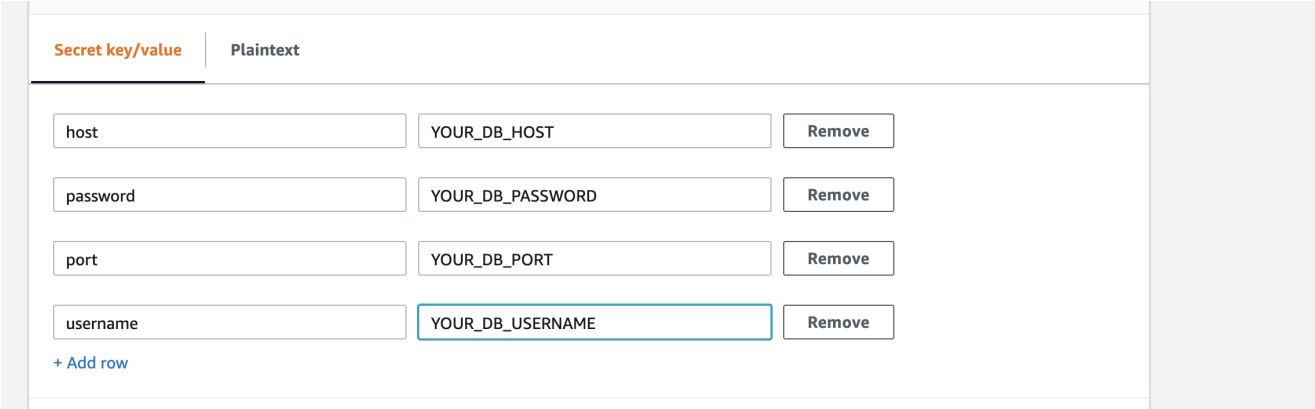
ActiveServer supports storing database credentials in the AWS Secrets Manager instead of the properties file. By storing the database credentials in the Secrets Manager you can easily rotate, manage, and retrieve secrets throughout their lifecycle.

1. Configure the following properties in the properties file.

AWS Secrets Manager Properties (Example)

```
as.db.url=<JDBC url, use "%s" to replace the host name and port e.g.
jdbc:postgresql://%s:%s/<Your DB name>
as.db.asm.region=<Optional region parameter if not provided AWS credentials
default region will be used. Example: us-west-1>
as.db.asm.secret-name=<Secrets name of the AWS Secrets manager>
```

2. Create an AWS secrets manager with the following key/value:
   - `password` - The password of the database.
   - `username` - The username of the database.
   - `host` - The host name of the JDBC URL. The first occurrence of "%s" in the `as.db.url` is replaced by this value.
   - `port` - The port part of the JDBC URL. The second occurrence of "%s" in the `as.db.url` is replaced by this value.



Note

By using AWS secrets manager, you can omit the `as.db.username`, `as.db.password` and `as.db.password-plain` properties.

ACCESS PERMISSIONS

Make sure you have configured the credentials with **Read** permission to the secret manager. For more information about configuring the credentials refer below.

Web server settings

Web server settings allows you to configure the default server ports, as well as other networking related variables. Depending on your network setup, you can choose to use HTTP or HTTPS. With HTTP, the entry point must be accessible from the Internet, and so a load-balancer or reverse proxy may be required to handle HTTPS traffic as well as SSL termination.

By default, the `server port` serves all web page requests including authentication callback pages and admin UI interface requests.

Server port, protocol and SSL settings

```
## Server port, protocol and SSL settings
## protocol http|https|both
as.server.protocol=http
as.server.http.port=8080
as.server.https.port=8443
as.server.https.key-store=<Your keystore file path>
## keystore type, can be pkcs12 or jks
as.server.https.key-store-type=pkcs12
as.server.https.key-store-password=<Your keystore password>
## Set to false to disable this listening port
# as.server.enabled=false
```

- `as.server.https.key-store=`
  Keystore file path. A keystore is required for HTTPS. The keystore should contain server certificates for the specified HTTPS listening port. Note that for a production instance, the server certificate must be commercially signed by a CA.
- `as.server.https.key-store-type=`
  Keystore type. ActiveServer supports two different keystore types. Possible values are `pkcs12` or `jks`. Commercially signed certificates issued by a CA are typically in "pkcs12" format with a file extension of `.p12` or `.pfx`.
- `as.server.enabled=`
  Enable or disable the server listening port. To enable the server listening port, enter `true`. To disable server listening port, enter `false`.

Warning

HTTP is not recommended to access the web pages served by the server port. The HTTP setting for `as.server.protocol` should only be used for SSL termination in a production environment where a front-end load balancer handles TLS/SSL traffic and forwards the request to the back-end AS instance via HTTP.

Depending on your network configuration, you may want to setup administration access via a separate port. To do so, the following settings must be applied. By default, the Admin port number is disabled. If enabled, port numbers set below must not conflict with any other port numbers.

Enabling this setting will restrict all administration UI interface traffic to the specified port. The `server port` will then serve the authentication callback pages only.

Admin port (Example)

```
#Admin port , protocol and ssl config
#By default, Admin port configuration shares with Server port configuration
#Set following setting to true to enable this listening port
as.admin.enabled=false
as.admin.http.port=9090
as.admin.https.port=9443
#protocol http|https|both
as.admin.protocol=http
as.admin.https.key-store=<Your keystore file path>
#keystore type, can be pkcs12 or jks
as.admin.https.key-store-type=pkcs12
as.admin.https.key-store-password=<Your keystore password>
```

Warning

HTTP is not recommended to access your administration UI. The HTTP setting for `as.admin.protocol` should only be used for SSL termination in a production environment where a front-end load balancer handles TLS/SSL traffic and forwards the request to the back-end AS instance via HTTP.

Authentication and Admin API port. Only available in HTTPS with mutual authentication. This port will be enabled once the ActiveServer instance is activated. A server restart is required to enable this port.

Auth API port (Example)

```
#Auth api port, only https port is configurable
as.api.port=7443
```

Directory Server port settings

The following Directory Server listening port settings are for **ActiveServer** to send and receive requests with the Directory Server in mutual authentication during the RReq/RRes processing. These connectors are always HTTPS enabled. Server and client certificates for Directory Servers can be configured later on, as described in Manage DS certificates.

> **Note**
>
> Once you complete the certificate settings on the Administrator UI, a server restart is required.

Each Directory Server has ports used to connect to the card brand production Directory Server and GPayments TestLab Directory Server, which are configurable from the following properties:

- `as.<Card Scheme>.port=` for **production** DS and `as.testlab.<Card Scheme>.port=` for **GPayments TestLabs** DS.

  These are the port numbers for each card scheme for listening on their Directory Servers. Default values can be found below.

  > **Note**
  >
  > The port number must not conflict with any other port numbers.

  > **Important**
  >
  > If you are using load balancers to forward ports, make sure to use same ports for forwarding (e.g. forwarding 9802 to 9802) for **TestLab** DS as **ActiveServer** will set the ports in the RReq URL to be in the format `https://<API URL or External URL>:<Port number>` to which the DS will send the results request during the challenge flow. For production DS, different ports can be used as the RReq URL is configurable from 3DS server URL.

- `# as.<Card Scheme>.enabled=false` for **production** DS and `as.testlab.<Card Scheme>.enabled=false` for **GPayments TestLabs** DS.

  This parameter is commented out by default. Determines the status of the Directory Server listening port, disabled or enabled.
  To disable the Directory Server listening port, enter `false`, otherwise, leave it commented out.

American Express

Production TestLabs

```
as.amex.port=9600
## Set to false to disable DS HTTPS listening port
# as.amex.enabled=false
```

China UnionPay

Production

```
as.chinaunionpay.port=9601
## Set to false to disable DS HTTPS listening port
# as.chinaunionpay.enabled=false
```

Discover / Diners Club International

Production TestLabs

```
as.discover.port=9602
## Set to false to disable DS HTTPS listening port
# as.discover.enabled=false
```

JCB

Production TestLabs

```
as.jcb.port=9603
## Set to false to disable DS HTTPS listening port
# as.jcb.enabled=false
```

Mastercard

Production TestLabs

```
as.mastercard.port=9604
## Set to false to disable DS HTTPS listening port
# as.mastercard.enabled=false
```

Visa

Production TestLabs

```
as.visa.port=9605
## Set to false to disable DS HTTPS listening port
# as.visa.enabled=false
```

eftpos

Production

```
as.eftpos.port=9800
## Set to false to disable DS HTTPS listening port
# as.eftpos.enabled=false
```

Keystore settings

ActiveServer provides 4 options for storing encryption keys:

- Local keystore (SunJCE)
- Amazon S3 keystore
- PKCS11 HSM
- Amazon KMS

Use the following property to set the keystore type:

```
as.keystore.type=<keystore type>
```

- `as.keystore.type=`
  Keystore type - possible values are `local` , `s3` , `pkcs11` , `kms` .

Local keystore (SunJCE)

To use a local keystore file, use the following property:

Local keystore (SunJCE) (Example)

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

- `as.keystore.local.path=`
  Keystore file path. Enter the file path to your keystore file, which should point to the folder that contains the keystore files.

> **Warning**
>
> If using a Windows based machine, note that an escape character (`\`) is probably required when setting the full path to the keystore folder.
>
> E.g. if a Windows share folder is being used with the path `\\ActiveServer\keystores` , the keystore path would be set as `as.keystore.local.path=\\\\ActiveServer\\keystores` .

TYPES OF KEYSTORES

Local keystores are SunJCE keystore in which the encryption keys are stored in local file. There are 3 main types of cryptographic keys used in **ActiveServer**, which are created automatically by the system. All keystore files are created in the directory specified in the `as.keystore.local.path` .

1. **Per merchant data key** - For authentication API v1 transactions, local keystores are created per merchant in which the data keys are stored in a file in the format `as_<UUID>.jks` . Transactions performed by this merchant will be encrypted using the same data key until the key is rotated. The key is created during the creation of new merchant. The key alias will be created in the format `AS_<UUID>` .

2. **Master key** - For authentication v2 transactions, **ActiveServer** uses envelope encryption in which the data is encrypted by a data key, and the data key is then encrypted by the master key. This is similar to how the AWS KMS manages CMK. The master key is stored in the file `as_master_key.jks`. The key alias will be in the format `MASTER_<UUID>`.
3. **System key** - For encrypting system related data, the data keys are stored in keystore file created in the format `as_sys_<UUID>.jks`. The keys are created during the initial system startup. The key alias will be created in the format `AS_SYS_<UUID>`.

> **Important to backup**
>
> A backup of this master key store file is very highly recommended, as losing the master key will result in being unable to decrypt the previously encrypted data keys. We recommend making a backup of the file whenever you rotate the master key.

Amazon S3 keystore

ActiveServer supports using Amazon S3 as the keystore. Similar to local keystore, the keystore is a SunJCE keystore, the only difference being the keystore files are stored in a S3 bucket rather than in a local folder. To utilize an Amazon S3 keystore, you need to set the *AWS Bucket*, *AWS Region*, and *AWS Credentials* settings.

AWS BUCKET

1. Please create an empty S3 bucket by following the AWS guide.
2. Set your AWS Bucket path in the following properties:

Amazon S3 keystore (Example)

```
as.keystore.s3.bucket-name=<Your S3 Bucket Name>
```

If only the bucket name is provided e.g. `as.keystore.s3.bucket-name=as-example-bucket`, required keystores will be created by ActiveServer in the root directory. If a path is added to the bucket name, it will create the directory with given name inside the bucket. For example, `as.keystore.s3.bucket-name=as-test-bucket/keystores` will create all the required keystores under `keystores` directory inside the bucket `as-example-bucket`.

AWS REGION

The AWS Region can be set in several ways. A list of region codes can be found in the *Region* column of this table: Amazon AWS - Regions and Availability Zones.

1. Set the AWS Region code in the following properties:

Amazon S3 keystore (Example)

```
as.keystore.s3.region=<Your S3 Region Name>
```

2. Or, set the AWS Region in the AWS config file on your local system. The config file should be located at: `~/.aws/config` on Linux, macOS, or Unix, or `C:\Users\USERNAME\.aws\config` on Windows. This file should contain lines in the following format:

```
[default]
region = <Your S3 Region Name>
```

ACCESS PERMISSIONS

Make sure you have configured the credentials with **Read** and **Write** permission to the bucket.

Set your AWS Credentials in the following properties if you want to use different credentials for the S3 bucket.

Amazon S3 keystore (Example)

```
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>
```

Warning

The above properties are no longer recommended but these credentials properties are maintained for anyone using an ActiveServer version older than 1.3.0. It is strongly recommended to use the other credential options available and configure ActiveServer wide IAM credentials with the right permissions.

AWS Key Management Service (KMS)

AWS Key Management Service (KMS) makes it easy for you to create and manage cryptographic keys in **ActiveServer**. There are no different types of keys for KMS, one CMK is used to manage all the encryptions/decryptions performed within **ActiveServer**.

CONFIGURATION

1. Create a **symmetric** Customer Managed Key (CMK) in KMS by following the guide provided by AWS.
2. Copy and paste the **key ARN** from the AWS KMS dashboard to the properties file:

AWS KMS (Example)

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

ACCESS PERMISSIONS

Make sure you have configured the credentials with **Read** and **Write** permission to the bucket. For more information about configuring the credentials refer below.

> **Warning**
>
> Note that only **Auth API v2** is supported when using AWS KMS. **Auth API v1** transactions will be result in **error code 1027**.

PKCS11 HSM

ActiveServer supports using a HSM as the keystore. If HSM keystore is used, the data keys are created inside the HSM specified and will never leave the HSM. The cryptographic keys created are similar to local keystore and you may check the key alias names stored in your HSM to verify the key formats. The HSM must support the PKCS11 API. To use hardware encryption with a PKCS11 HSM, you will need the following properties:

PKCS11 HSM (Example)

```
as.keystore.pkcs11.library=<the library to the pkcs11 driver>
as.keystore.pkcs11.slot=<the slot number>
```

- `as.keystore.pkcs11.library=`
  HSM driver library. For Linux, this is typically a `.so` file. For Windows, this is typically a `.dll` file. Consult your HSM's documentation for details on the library that should be use.
- `as.keystore.pkcs11.slot=`
  The slot number on your HSM. Consult your HSM's documentation for details on the slot number that should be use.

Info

Note that setting up and configuring an HSM is out of scope for this document. Please ensure your HSM is fully functional before configuring with ActiveServer.

HSM Token Login is required

When using a HSM for key management, AS requires the HSM to enable *"Always require Token Login"*. This setting is usually set as on automatically for most HSMs, however for HSMs like Safenet, this setting may not be on by default. Please refer to your HSM documentation for instructions.

For SafeNet HSMs, this can be done by setting the `No Public Crypto` security flag. Administrators can use the provided command line utility `ctconf` to set the flag.

AWS Credentials

AWS credentials needs to be configured so that ActiveServer can access the AWS services. Make sure the right permissions are assigned to this credentials. AWS Credentials include an `access_key_id` and a `secret_access_key` . AWS Credentials can be set in a number of ways:

1. Set your AWS credentials in the following properties:

AWS credentials (Example)

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

2. Or, set the AWS Credentials in the AWS credentials profile file on your local system. The credentials profile file should be located at: `~/.aws/credentials` on Linux, macOS, or Unix, or `C:\Users\USERNAME\.aws\credentials` on Windows. If using this method, the **ActiveServer** properties file can be left blank. This file should contain lines in the following format:

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

3. Or, if you deploy **ActiveServer** on an AWS EC2 instance, you can specify an IAM role and then give your EC2 instance access to that role. In this case, you need to follow the Amazon AWS - Using IAM Roles to Grant Access to AWS Resources on Amazon EC2 guide. If using this method, the **ActiveServer** properties file can be left blank.

Email server settings

ActiveSever allows you to send email notifications to users. Email notifications can be used to notify a user of their activation URL, remind users when a license is about to expire etc.

You will need an email account with its associated credentials and server details to setup email notifications.

Email Server Properties (Example)

```
as.mail.host=<Your SMTP server host>
as.mail.port=<Your SMTP server port>
as.mail.user-name=<Email server username>
as.mail.password=<Email server password>
as.mail.auth=true
as.mail.start-tls=true
as.mail.from=<Email address to send emails from, e.g. admin@activeserver.com>
```

- `as.mail.host=`
  The SMTP domain of your email server.
- `as.mail.port=`
  The port number of your email server.
- `as.mail.user-name=`
  The email server username.
- `as.mail.password=`
  The email server password.
- `as.mail.auth=`
  Whether the email account requires SMTP authentication. If the email account requires authentication, enter `true`. Otherwise, enter `false`. If unsure, consult your email server administrator for details.
- `as.mail.start-tls=`
  TLS required by the SMTP server or not. If SMTP server requires TLS, enter `true`. Otherwise, enter `false`. If unsure, consult your email server administrator for details.
- `as.mail.from=`
  Email address in which to send the email from.

Info

Note that setting up and configuring email servers is out of scope for this document. Please ensure your email server is fully functional before configuring with ActiveServer.

Log settings

By default, **ActiveServer** will output all log files to the `{AS_HOME}/logs/` folder, which will be created if it does not exist. If you would like to specify a different log folder location, uncomment and edit the following setting with the desired path:

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

If using a multi node setup for the instance, separate folders are required for each node due to the file naming conventions.

> **Warning**
>
> If this value is changed, note that any errors in setup may result in the log files not being recorded correctly. Ensure that the path is correct and that the **ActiveServer** instance can access and has read/write permissions for the specified path.

Disabling local file output logging

If local file output logs are not required, **ActiveServer** has an option to disable this function.

To disable **ActiveServer** saving logs to local files, add the command `disableFileLogs` at the end of `AS_PROFILES` in the `startup.sh` (Linux) or `startup.bat` (Windows) file before starting the application, e.g.

startup.sh startup.bat

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

Logs will still be sent to the standard console output regardless of this setting for security and application support purposes.

Trans-type override setting

`trans-type` parameter is used in DS Profile for switching between Production Directory Servers and TestLabs. If you want to override the functionality of `trans-type` parameter permanently so

that it doesn't need to be specified in the API call, the following parameter should be uncommented and edited with the desired value:

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

- If `as.auth.allowed-trans-type=testlab` is specified, then only TestLabs transactions are allowed and all API requests will be forwarded to the TestLabs DS profile. The `trans-type` parameter in API calls will be ignored if presented.
- If `as.auth.allowed-trans-type=prod` , is specified, then only Production transactions are allowed and all API requests will be forwarded to the Production DS profile. The `trans-type` parameter in API calls will be ignored if presented.
- If `as.auth.allowed-trans-type` is not presented or the value is `null` / `all` , then no overriding is enforced. The client side can use `trans-type` as described in the DS Profiles guide.

> **Warning**
>
> **Note that `as.auth.allowed-tans-type` overrides the `trans-type` parameter in API calls, which may forward authentication requests to an unexpected Directory Server** if the user is informed an override is in place.

Setting TLS version

ActiveServer specifies HTTPS connectors to use TLS 1.2 only by default. However, due to a reported issue with Java 1.8, the TLS 1.0 and 1.1 protocols are also enabled. If you wish to disable protocols, a workaround is possible by editing the Java security file directly and is shown below.

> **Warning**
>
> **Note that editing the `java.security` file will affect all Java applications on the server. GPayments cannot be held responsible for any issues that may result from this change.**

To disable a specific protocol, edit the `java.security` file located at `<jdk directory>/jre/lib/security` and update the `jdk.tls.disabledAlgorithms` entry. The original entry might look like the below:

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

To disable protocols considered weak and not included by default, such as `SSLv2Hello, TLSv1, TLSv1.1`, append those values to the entry like below:

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

Important

The suggested protocols to disable are just suggestions, consult your security team when deciding your security configuration for **ActiveServer**.

After editing the `java.security` file, restart any running **ActiveServer** instances for the change to take effect. You can check what protocols are enabled by running the following command on a linux terminal:

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

Overriding the 3DS Server Reference Number

For backward compatibility, by default **ActiveServer** uses the 3DS Server Reference number issued by EMVco during v2.1.0 certification. The default reference number is only valid for sending v2.1.0 requests and the card scheme Directory Servers will likely reject the v2.2.0 requests using this reference number. If you would like to update the 3DS Server Reference number sent in the AReq, there are two ways to do this.

1. Override the 3DS Server Reference number per card scheme.
2. Override the 3DS Server Reference number server for all card schemes.

Using either method, please configure the 3DS Server Reference number to GPayments issued EMVCo v2.2.0 compliant reference number `3DS_LOA_SER_GPPL_020200_00351` when you are ready to start using the v2.2.0 protocol. Note that the new reference number also supports sending v2.1.0 requests.

> **Only update after compliance testing**
>
> **You should only update the 3DS Server Reference number after completing the compliance testing for each card scheme**, as the DS will likely reject the v2.2.0 3DS Server Reference number if not compliant.

Override 3DS Server Reference number per DS

The following configuration can be used to override the 3DS Server Reference number sent for each card scheme by adding it to the **application-prod.properties** file for **every application node**.

> **All application property files must be updated**
>
> These settings are per node, so updating all application files is required to ensure the correct reference number is sent.

AMEX Discover JCB Mastercard Union Pay Visa eftpos

```
## Overrides the 3DS Server Reference number used for American Express
as.prod.server-ref-number.amex=Reference Number
```

> **Overriding during compliance testing**
>
> This overriding feature may also be used during card scheme's compliance testing, as some card scheme requires the provider to use a specific 3DS server reference number issued by them for testing purposes.

Override 3DS Server Reference number globally

This global overriding is useful if you do not want to update the configurations for all the card schemes individually. If provided, it will override all the card scheme configuration overriding values.

```
## Overrides the 3DS Server Reference number used for all card schemes
as.prod.server-ref-number-overriding=Reference Number
```

> **Warning**
>
> If using global override, ensure compliance testing has been completed for all card schemes, otherwise some Directory Servers may reject requests.

Overriding the PReq message version number

ActiveServer v2.0.0 sends PReq's with message version 2.2.0 for Visa, Mastercard and American Express, as these Directory Servers handle this version by default. JCB and Discover require card scheme compliance testing and registration of the 3DS Server Reference number to be completed before 2.2.0 PReq's can be sent, so these card schemes still send 2.1.0 PReq's by default.

During card scheme compliance testing, it may be required to send either 2.1.0 or 2.2.0 PReq messages. It will also be required to update all card schemes to use 2.2.0 PReq's on your production instance, **although this should only be done after compliance testing is completed**. To assist with this, a PReq message overriding setting has been added to the application-prod.properties file:

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

- **Message version:** Can be `2.2.0` or `2.1.0`
- **Card scheme:** Can be `amex`, `discover`, `jcb`, `mastercard`, `unionpay` or `visa`

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

> **All application property files must be updated**
>
> These settings are per node, so updating all application files is required to ensure the PReq message version is sent.

Deployment options

ActiveServer can be deployed in three modes:

1. Standalone (default) - the node will perform all processes.
2. Frontrunner - the node will only perform transactions and UI functions.
3. Sidecar - the node will only perform the heavy background processes such as the card range cache updates.

Frontrunner/Sidecar Deployment

Frontrunner and Sidecar modes must be used together for ActiveServer to work properly. Each mode should have a separate server, with traffic only going to Frontrunner. Sidecar should not be registered with a load balancer for incoming API calls.

Prepare a new server to run the `sidecar` instance of ActiveServer. Copy your existing `AS_HOME` directory to the new server and make sure the local keystore (or HSM) are still accessible.

For the worker nodes, add a new setting `as.settings.runtime-type=frontrunner` to the `application.properties`. If you are using environment variables, add `AS_SETTINGS_RUNTIME_TYPE=frontrunner` to your system settings. Update the worker nodes. To confirm the worker nodes are running under `frontrunner` mode, check the `Runtime Type` in the start-up banner.

For the new `sidecar` server, add a new setting `as.settings.runtime-type=sidecar`. Optionally, you may want to reduce the thread pool size by setting `as.settings.worker-threads=2` to free up memory and a smaller db connection pool, e.g. `as.db.pool-size=5` to optimise db server's load.

GPayments tested full card range update with the Directory servers, and it was found that 4G of memory was sufficient for the major card scheme updates.

Based on this GPayments recommend that a minimum of 4G of memory is allocated to the instance, but ideally allocating 8GB of memory would ensure increases in the size of the full card range could be supported.

Include the following JVM parameters with your existing parameters (assuming your memory setting is 8G, please adjust accordingly):

```
-Xmx8g -Xms6g -XX:+UseG1GC -XX:InitiatingHeapOccupancyPercent=70
```

Update the `sidecar` node. The log console should show that all background process are now running in this node. The frontrunner nodes should show that background process are skipped.

Second Level Card Range Cache (SLCRC)

This feature is introduced to build and store cache on the Sidecar node and effectively skip the database access when querying card ranges.

In the SLCRC setup, the Frontrunners are the card range query client, and the Sidecar is the SLCRC Provider. Frontrunners will try to query the Sidecar SLCRC and fallback to the database when the query is failed.

In the Sidecar properties file add the following configurations:

- `as.slcrc.enableProvider=true|false` : To enable or disable the feature.
- `as.slcrc.provider.port` : Port the cache is reachable at.
- `as.slcrc.token` : Token to be sent with the request.

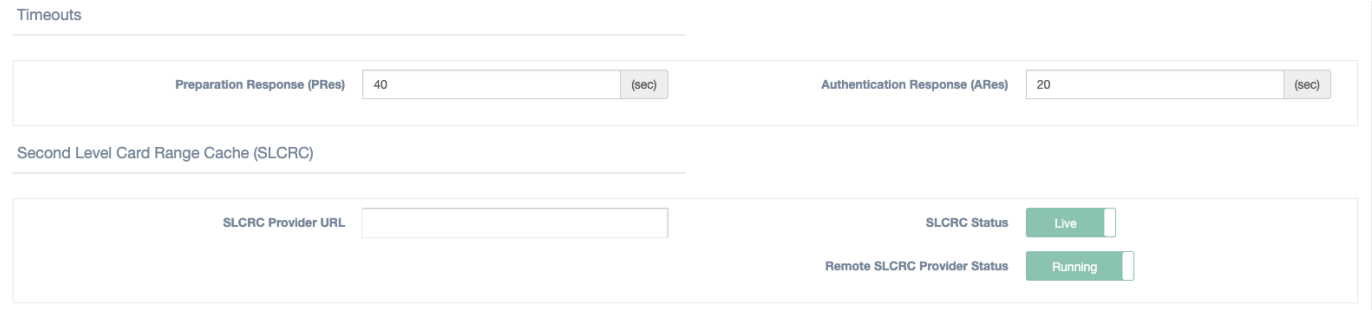
In the Frontrunner properties file add the following configurations:

- `as.slcrc.url` : Format is `https://<sidecar server internal host name or IP>:<the slcrc port>`
- `as.slcrc.token` : Token to be sent with all requests, must match the one set in the Sidecar properties file

For optimal performance when SLCRC is enabled, it is recommended the Sidecar node have 16GB of memory or a minimum of 8GB of memory.

TEST MODE

For the Frontrunner nodes, the SLCRC client configuration `as.slcrc.clientTestMode=true|false` can be used to test if the SLCRC configuration is working. GPayments suggests setting the test mode to `true`, then bring up the instance and test the connectivity on the Admin UI DS profile page.



Under test mode, the Frontrunners will not fail to start if the SLCRC connectivity check fails. The Admin UI will show the SLCRC Provider's status, and using pseudo card numbers, you can test the SLCRC card range queries on the **Card range** query page in Admin UI.

LIVE MODE

Live mode is when the `as.slcrc.clientTestMode` is set to `false`. In this mode, the ActiveServer instance will not start if the SLCRC provider is not reachable.

QUERY PROCESS

By default, when SLCRC is enabled, ActiveServer will try querying the card ranges with SLCRC, if the query fails, it will fallback to the original database queries.

Startup ActiveServer

Now that all properties are correctly configured, you can startup an instance of ActiveServer.

If you use Linux, open **Terminal**. If you use Windows, open **Command Prompt**.

Change your working directory to the folder that contains the startup scripts (`.sh` or `.bat` file).

Now you can startup ActiveServer with the following command.

Linux Windows

```
./startup.sh
```

> ✕ **Make sure the `as.jar` file is in the same directory as the startup scripts**
>
> The startup command will not work if the `as.jar` file is not in the same directory as the startup scripts.

You should now see the following output in **Terminal** or **Command Prompt**. Make sure to take note of the **Administration** URL, as it will be needed in the next step.

ActiveServer Instance Info

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

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]
--------------------------------------------------------------------------------
```

Startup script

In the startup scripts, the environment variable `AS_HOME` is set to the directory in which `application-prod.properties` is located. ActiveServer uses `AS_HOME` to find the configuration

files as well as maintain keystores and output logs. By pointing `AS_HOME` to different locations, it is possible to run multiple ActiveServer instances on the same server. This can be done by copying the package to a different directory, or creating different startup scripts, and within those scripts, pointing `AS_HOME` to different directories.

> **Note**
>
> Where there are multiple instances on the same server, the port numbers must not conflict in any of the `application-prod.properties` files.

JVM memory override

Sometimes it may be required to adjust the level of memory allocated to the JVM to increase performance of **ActiveServer**. While JVM memory tuning is outside the scope of this guide, the below command can be used to specify the exact amount of memory to be used by the JVM to run **ActiveServer**. The `Xmx"memory size"` command can be added to the **ActiveServer** startup script:

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

The above setting would allocate 3600 MB's of memory to be used, instead of the default allocation of ¼ of available memory.

Outbound proxy settings

For users who want to use outbound proxy server, the `startup.sh` can be modified as follows:

```
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 profile

In addition to setting `AS_HOME`, the startup script also sets the environment variable `AS_PROFILES`. This is a convenient mechanism to specify profile based configurations.

By default, the profile is set to `prod`.

ActiveServer uses the pattern `application-<profile>.properties` to load the profile's configuration file. Therefore under the default `prod` profile, `application-prod.properties` will be loaded. However, you can create new profiles (such as `test`) to setup different configurations for ActiveServer.

To create a new profile:

- Create a new configuration file named `application-test.properties` and place it in the same directory as the `prod` configuration file.
- Open the startup scripts and set the value of `AS_PROFILES` to `test`.

  ActiveServer will load the new profile instead of the old `prod` properties.

  ActiveServer can also load multiple profiles at the same time, to do this set the value of `AS_PROFILES` to `prod,test` and ActiveServer will load properties files from both `prod` and `test` profiles.

  With these options available, you can maintain settings separately under separate `.properties` files.

> **Tip**
>
> If you maintain all database settings in `application-db.properties`, and web server settings in `application-web.properties`, by setting the value of `AS_PROFILES` to `db,web`, ActiveServer settings can be segregated for different administrators to manage.

Setup Wizard

Once ActiveServer is up and running, you can access the Administrator UI via the **Administration** URL.

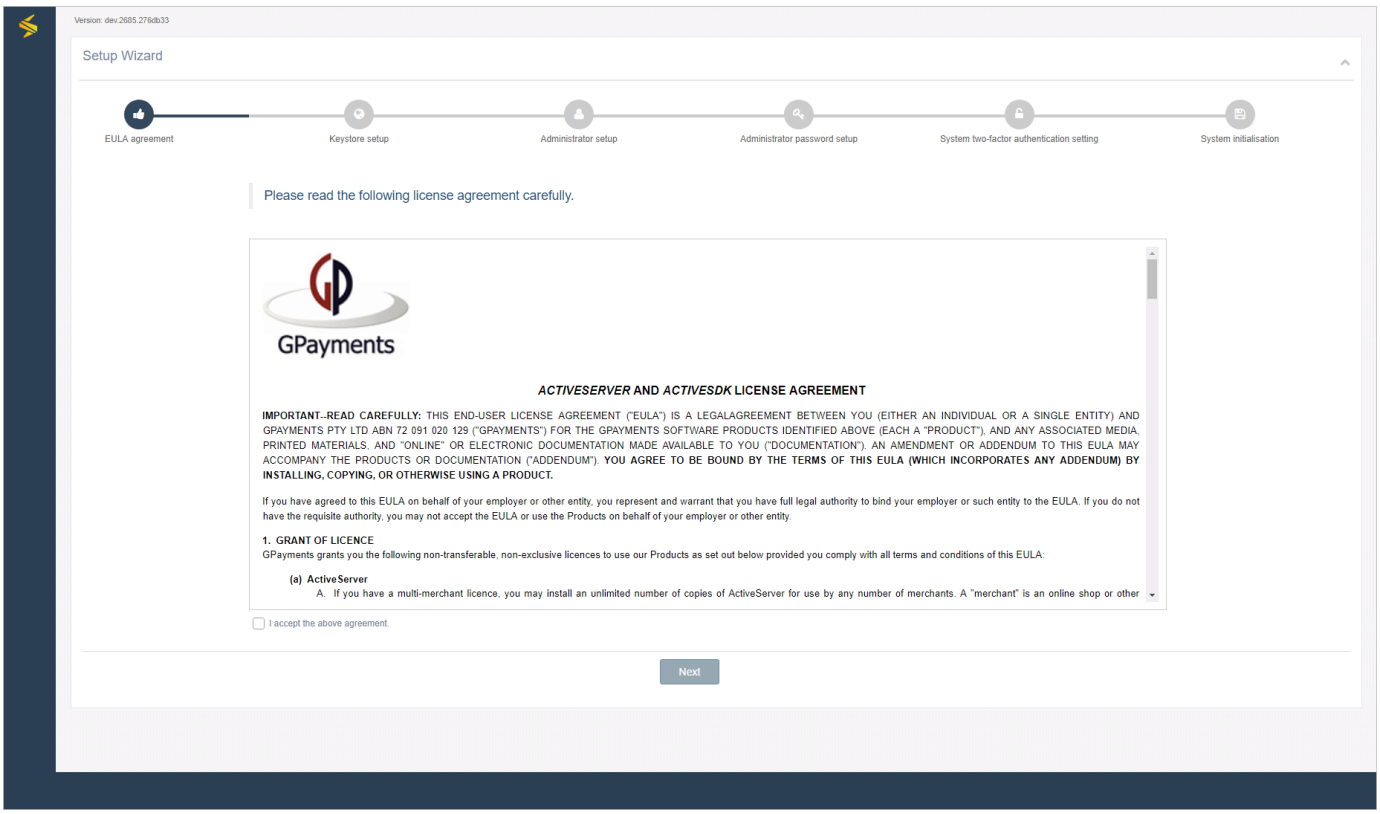
If this is your first time running ActiveServer, the Setup Wizard will appear and guide you through the setup process.

The Setup Wizard involves the following steps:

- EULA agreement
- Keystore setup
- Administrator setup

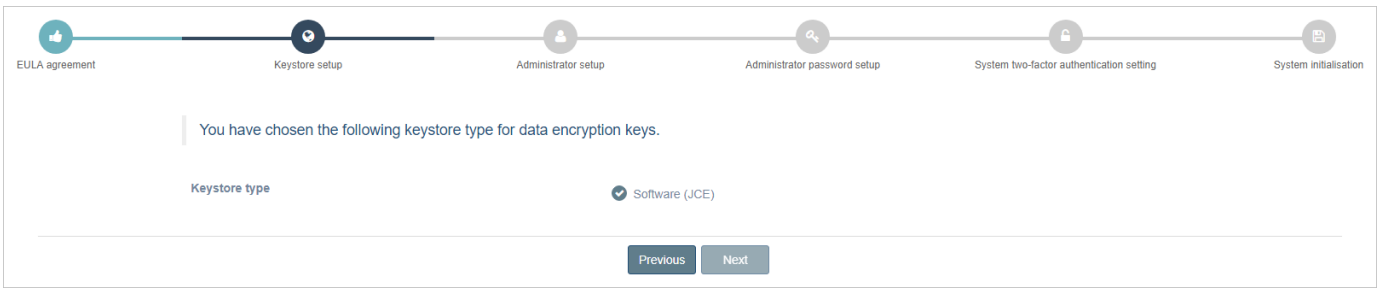
- Administrator password setup
- System two-factor authentication setting
- System initialisation

EULA agreement



Read the EULA Agreement. If you agree to the terms and conditions, select the **I accept the above agreement.** checkbox to proceed.

Keystore setup



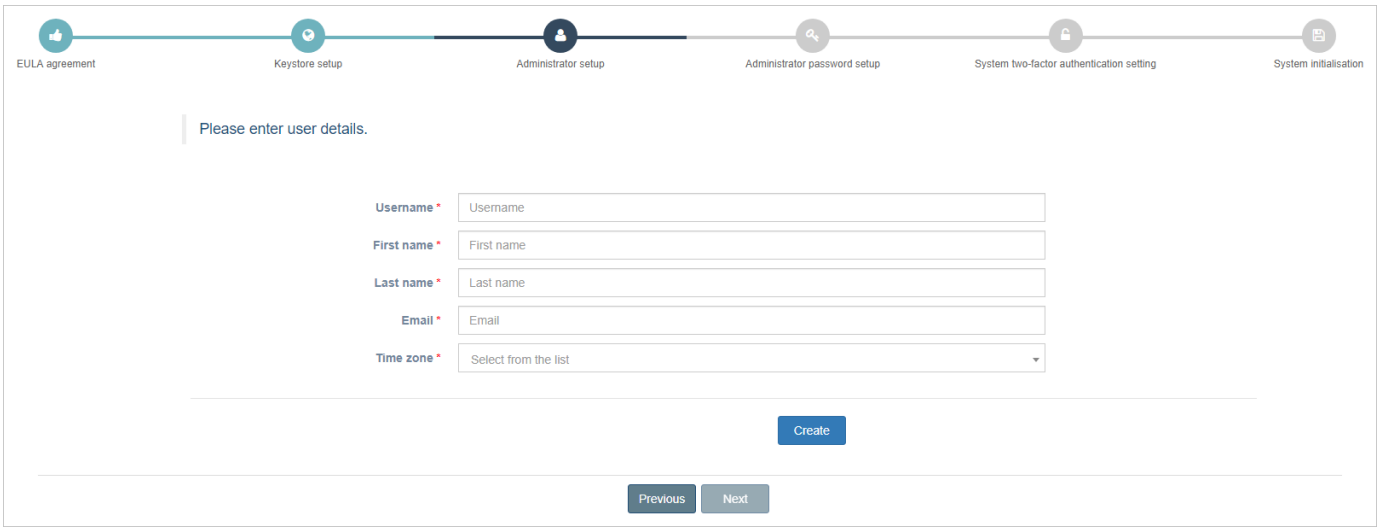
- Select the **Keystore type**.

  If **Software** was selected by during setup, SUNJCE will be utilised.

There is also the option to use a PKCS#11 HSM by entering the appropriate details in the `application-prod.properties` file. See the Encryption Module on how to setup a PKCS#11 HSM.

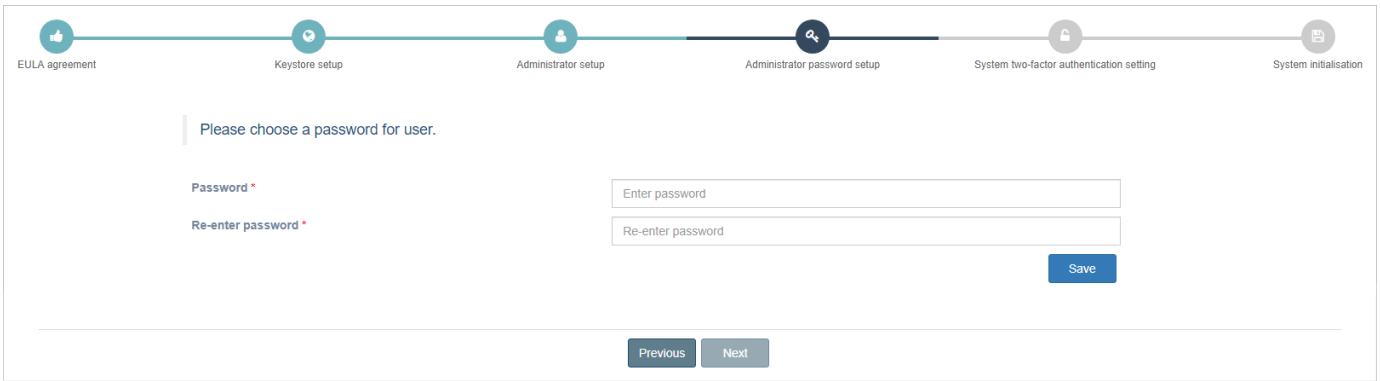
- Select the ***Next*** button to continue.

Administrator setup



- Enter the user details for your Administrator account.
- Select the ***Create*** button to create the account.
- Select the ***Next*** button to continue.

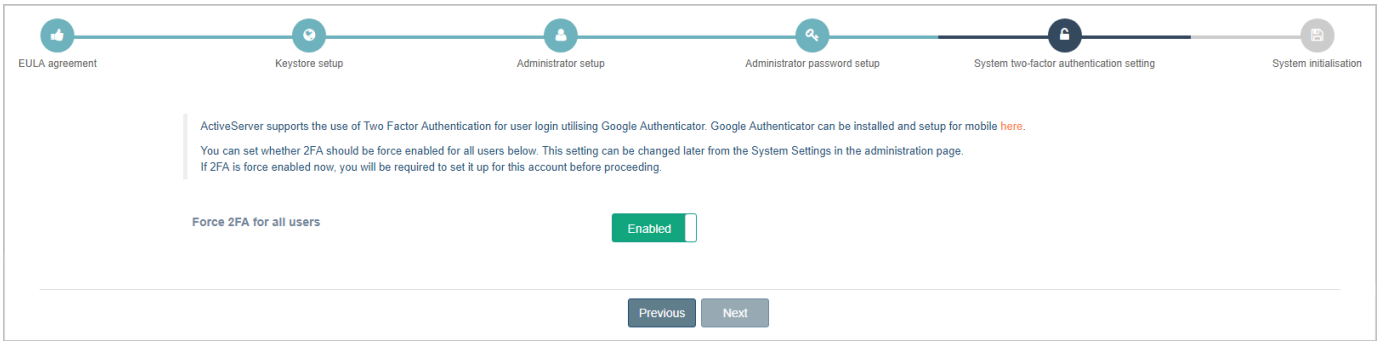
Administrator password setup



- Enter a password for the Administrator account.
- Re-enter the password to confirm it.
- Select the ***Save*** button to create it.

- Select the ***Next*** button to continue.

System two-factor authentication setting



ActiveServer supports two factor authentication for signing into the Administrator UI.

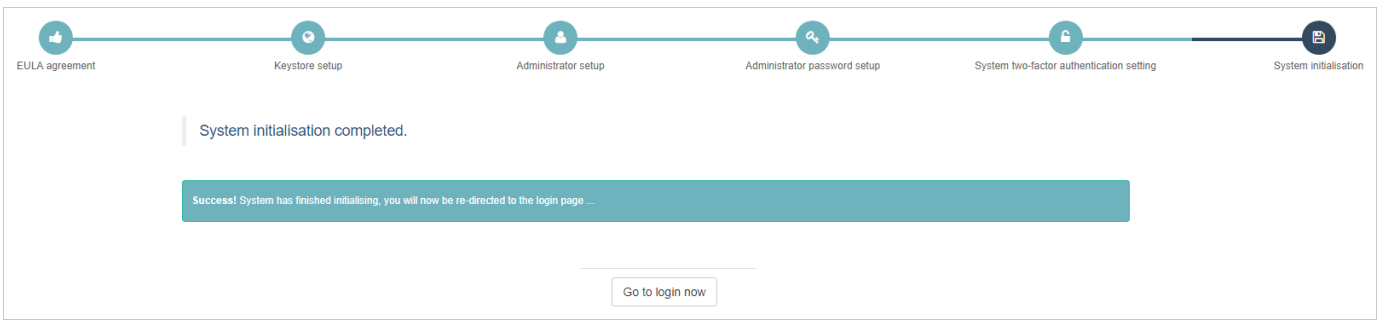
> **Note**
>
> To use this feature, you must have Google Authenticator installed on a mobile device.
>
> Refer to Install Google Authenticator for setup instructions for Google Authenticator.

By default, ActiveServer does not force users to use two factor authentication.

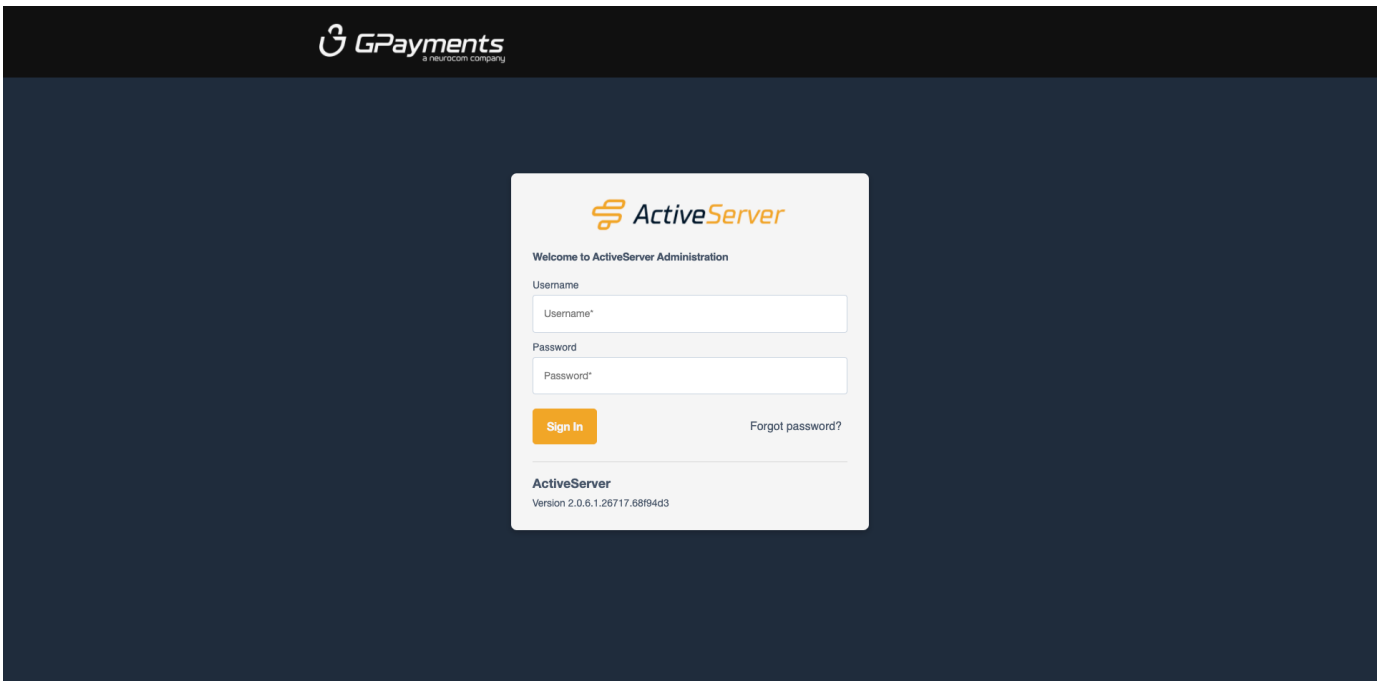
To mandate two factor authentication for all users:

- ***Enable*** the toggle, adjacent to **Force 2FA for all users**.
- Select the ***Next*** button to continue.

System initialisation



The Setup Wizard will inform you that system initialisation is complete and you will be redirected to the ActiveServer login page.



What next?

After this, you can:

- Activate ActiveServer
- Configure system settings
- Start integrating with the ActiveServer API