

Hyland Remote Configuration Server

Installation and Setup Guide

Version: 3.0

Written by: Product Knowledge, R&D
Date: November 2025



Documentation Notice

The information and software described in this document are furnished only under a separate agreement and may only be used or copied according to the terms of such agreement. It is against the law to copy the software except as specifically allowed in such agreement. Complying with all applicable copyright laws is the responsibility of the user. Without limiting the rights under copyright law, no part of this document may be reproduced, stored in or introduced into a retrieval system, or transmitted in any form or by any means (electronic, mechanical, photocopying, recording, or otherwise), or for any purpose, without the express written permission of Hyland Software, Inc. and/ or one of its affiliates.

Hyland, OnBase, Alfresco, Nuxeo, Content Innovation Cloud and other product or brand names are registered and/or unregistered trademarks of Hyland Software, Inc. and its affiliates in the United States and other countries. All other trademarks, service marks, trade names and products of other companies are the property of their respective owners.

Confidential © 2025 Hyland Software, Inc. and its affiliates.

The information in this document may contain technology as defined by the Export Administration Regulations (EAR) and could be subject to the Export Control Laws of the U.S. Government including for the EAR and trade and economic sanctions maintained by the Office of Foreign Assets Control as well as the export controls laws of your entity's local jurisdiction. Transfer of such technology by any means to a foreign person, whether in the United States or abroad, could require export licensing or other approval from the U.S. Government and the export authority of your entity's jurisdiction. You are responsible for ensuring that you have any required approvals prior to export.

DISCLAIMER: This documentation contains available instructions for a specific Hyland product or module. This documentation is not specific to a particular customer or industry. All data, names, and formats used in this document's examples are fictitious unless noted otherwise. This document may reference websites operated by third parties. In such a case, Hyland has no control or liability for the content of such third-party websites. The inclusion of such a link shall not constitute an endorsement or affiliation with such a third-party website; the reference is provided for information purposes only. If you have questions about discrepancies in this document, please contact Hyland. Hyland customers are responsible for making their own independent assessment of the information in this documentation. This documentation: (a) is for informational purposes only, (b) is subject to change without notice, (c) is confidential information of Hyland Software, Inc. and its affiliates and (d) does not create any commitments or assurances by Hyland. This documentation is provided "as is" without representation or warranty of any kind. Hyland expressly disclaims all implied, express, or statutory warranties. Hyland's responsibilities and liabilities to its customers are controlled by the applicable Hyland agreement. This documentation does not modify any agreement between Hyland and its customers.

Table of Contents

Overview	5
Install Hyland Remote Configuration Server	5
Docker Installation	5
<i>Download the Docker Image</i>	5
<i>Configure appsettings.json</i>	6
<i>Create a Docker Secret</i>	6
<i>Create a Docker Compose File</i>	6
Docker Compose File Example	6
<i>Deploy and Start Docker Stack</i>	7
Manual Installation	7
Manual Uninstallation	7
Configuration	8
Kestrel	9
<i>Certificates</i>	9
<i>Endpoints</i>	9
Authentication	9
<i>JwtBearer</i>	9
Authorization	10
Database	11
ConnectionStrings	11
PluginOptions	12
SecretSettings	12
<i>Secrets</i>	12
Host	13
Hyland.Logging	13
<i>Routes</i>	13
CredentialSettings	15
CryptographySettings	15
TransactionSettings	16
RouteConstraintSettings	16
ActiveActiveConfig	17
<i>AMQPConfig</i>	17
<i>Ssl</i>	18

<i>Options</i>	19
Appendix: Add Trusted Root CA Certificates	20

Overview

Hyland Remote Configuration Server (HRCS) manages configuration settings across applications and versions.

The System Administrator has the flexibility to control which settings the Application Administrator has access to so that infrastructure settings are not accidentally changed. The flexibility of this allows each customer environment to be tailored for their IT needs. This access is called configuration visibility and is configured per configuration type.

Managing a configuration remotely allows you to make configuration changes without the assistance of a System Administrator.

The service supports managing multiple instances of a configuration type and supports multiple services using the same configuration instance. This flexibility allows services, such as **inserver** or **inserverWorkflow**, to use the same inow representation for the database connection. However, if you wanted two instances, such as an E-Mail Broker with different configurations, to distribute the load of your environment, you could set that up in your environment as well.

Install Hyland Remote Configuration Server

The server has two methods of execution, Docker and manual installation. Docker installation uses a Docker containerization tool to manage the server through a Docker Image. Manual installation uses a Windows service to manage the server.

Docker Installation

The following outlines the steps needed to install Hyland Remote Configuration Server using containerization tools:

- Download the Docker image
- Configure appsettings.json
- Create Docker secret
- Create Docker compose file
- Deploy and start Docker stack
- Add trusted root CA certificate

Download the Docker Image

To pull down a docker image and run the Hyland Remote Configuration Server you must first pull the image down and then copy the appsettings.json from the image to a local directory. The **appsettings** file will need to be properly configured before running the docker image.

1. To get the **appsettings.json** out of the image, execute the following command to create a container from the image and then make note of the **ID** that is returned.

```
docker create devrepositoryd00.pvi.com:5006/hyland-remote-configuration-server:<version>
```

2. To copy the **appsettings.json** out of the container execute the following command, where **ID** is the ID you just noted.

```
docker cp <ID>:/app/appsettings.json appsettings.json
```

3. To remove the container, execute the following command:

```
docker rm <ID>
```

Configure appsettings.json

To configure the appsettings.json configuration file, complete the following steps:

1. In a text editor, open the **appsettings.json** file.
2. Update the default **appsettings.json** by following the [Configuration](#) section.
3. Save the **appsettings.json** file and close the text editor.

Create a Docker Secret

To create the Docker Secret from the **appsettings.json** file, execute the following command:

```
docker secret create appsettings_secret ./appsettings.json
```

Create a Docker Compose File

To create a Docker Compose file, complete the following steps:

1. Configure the following database environment variables:
 - Set **AppSettingsFileOverridePath** the path to override where the **appsettings** file is located.
 - ASPNETCORE_URLS:https://+:5001
 - ASPNETCORE_Kestrel__Certificates__Default__Path= /app/config/HRSC_Docker_Cert.pfx
 - ASPNETCORE_Kestrel__Certificates__Default__Password=<password>
2. Configure the ports. The port values should match the **ASPNETCORE_URLS** port values.
3. Configure the volumes.
4. Create a volume for **/app/PluginResources**.
5. Optional. Create a volume for the certificates.
6. To reference the docker secret in your docker compose file, complete the following substeps:
 1. Place the docker secret with the definition of secrets and specify for the service.
 2. Point the **AppSettingsFileOverridePath** environment variable to the secret file location to load the settings at astartup.

Docker Compose File Example

The following is an example of a Docker Compose file.

```
version: "3.3"
services:
  configurationserver:
    image: devrepositoryd00.pvi.com:6000/hyland-remote-configuration-
server:<semantic_version>
    ports:
      - 5001:5001
    secrets:
```

```

    - appsettings_secrets_v4
  volumes:
    - ./logs:/app/logs
    - ./PluginResources:/app/PluginResources
    -
  ./certs/HRCS_HTTP_SSL_CERTIFICATE.pfx:/app/certs/HRCS_HTTP_SSL_CERTIFICATE.pfx
  environment:
    AppSettingsFileOverridePath: /run/secrets/appsettings_secrets_v4
    ASPNETCORE_URLS: https://+:5001
    ASPNETCORE_Kestrel__Certificates__Default__Path:
/app/certs/HRCS_HTTP_SSL_CERTIFICATE.pfx
    ASPNETCORE_Kestrel__Certificates__Default__Password: password_goes_here

secrets:
  appsettings_secrets_v4:
    external: true

```

Deploy and Start Docker Stack

To deploy and start the stack, execute the following command:

```
docker stack deploy --compose-file docker-compose.yml configstack
```

Manual Installation

To manually install HRCS, complete the following steps:

1. Extract the artifacts to the desired installation directory. You must also extract the plugins' zips into their own directories. The plugin directories may be sub-directories of the installation directory, but this is not required.
2. Configure **appsettings.json**. For detailed configuration information, see the [Configuration](#) section.
3. To register the service, complete the following substeps:

1. From an elevated PowerShell, execute the following command:

```
New-Service -Name "<SERVICE_NAME>" -BinaryPathName
"<INSTALL_DIR>\Hyland.Remote.Configuration.Server.exe" -StartupType Automatic
```

Where:

<SERVICE_NAME> is a user defined name for the new services. This is typically **Hyland Remote Configuration Server**.

<INSTALL_DIR> is the absolute path to the desired installation directory.

For more details on the New-Service command, see Microsoft's [New-Service Documentation](#)

Manual Uninstallation

To manually uninstall HRCS manually, complete the following steps:

1. Remove the service. The mechanism to remove a Windows service varies depending on the version of PowerShell used.

1. For PowerShell 6 or newer, open an elevated PowerShell and execute the following command:

```
Remove-Service -Name "<SERVICE_NAME>" -Confirm
```

Where **<SERVICE_NAME>** is the name of the registered service.

Verify that PowerShell displays the expected service target and enter 'Y' to remove the service, or 'N' to abort the **Remove-Service** command.

For more details on Remove-Service, see Microsoft's [Remove-Service Documentation](#).

2. For older PowerShell, open an elevated PowerShell and make note of the service name that you want to remove.

Important: Double-check that you have the correct service name - this legacy method does not prompt you for confirmation before removal.

In the elevated PowerShell, execute the following command:

```
sc.exe delete "<SERVICE_NAME>"
```

Where **<SERVICE_NAME>** is the name of the registered service.

For more details on **sc delete**, see Microsoft's [SC Delete Documentation](#)

2. Delete the installation directory. If the plugins are not in a sub-directory of the installation directory, you must also delete the plugin directories.
3. Optional: For a full uninstallation of the entire service, you must delete the HRCS database using your database management programs. The database name is defined by the connection string in the HRCS configuration.

Configuration

HRCS uses an **appsettings.json** file to handle configuration. HRCS does not support other configuration mechanisms.

The **appsettings.json** file controls all modifiable behaviors of your HRCS. When configuring a new HRCS, you must configure the following sections.

- [Kestrel](#)
- [Authentication](#)
- [Authorization](#)
- [Database](#)
- [ConnectionStrings](#)
- [PluginOptions](#)
- [SecretSettings](#)
- [Host](#)

The following sections are optional.

- [Hyland.Logging](#)
- [TransactionSettings](#)
- [CredentialSettings](#)
- [CryptographySettings](#)
- [RouteConstraintSettings](#)
- [ActiveActiveConfig](#)

Kestrel

The **Kestrel** section allows you to directly configure the **ASP.NET Kestrel** settings. For a full set of **Kestrel** settings, please reference Microsoft's [Kestrel Documentation](#).

For HRCS, the **Kestrel** section contains two important sub-sections, **Certificates** and **Endpoints**

Certificates

The **Certificates** settings control the SSL certificate settings for HRCS. To set an SSL certificate for HRCS, specify the **Default Certificate** in the **Certificates** sub-section.

The following table lists the basic **Certificate** settings.

Setting	Description
Path	Specifies the path to the .crt or .cer file for this server's certificate.
KeyPath	Specifies the path to the .key file for this server's certificate.
Password	Specifies the password for the configured certificate.

Endpoints

The **Endpoints** settings control how the server is exposed to callers. At a minimum for HRCS, you need to configure the **Https URL**. This URL may be a full URL, or it may use a wildcard to just specify the port and use the host machines DNS.

Example

```
"Kestrel": {
  "Certificates": {
    "Default": {
      "Path": "<path to>.<cer or crt>",
      "KeyPath": "<path to>.key"
    }
  },
  "Endpoints": {
    "Https": {
      "Url": "https://*:5001"
    }
  }
},
```

Authentication

The **Authentication** section contains the **JwtBearer** sub-section

JwtBearer

The **JwtBearer** sub-section controls how the server verifies bearer tokens for **Admin** and **User** authentication and allows you to specify the **Admin** and **User** settings in their own sub-sections. For more information on available **JwtBearer** settings, see [Microsoft JwtBearerOptions Class](#).

Setting	Description
---------	-------------

Admin	Specifies how the server verifies tokens for administrative authentication.
User	Specifies how the server verifies tokens for user authentication.

Admin and User

Generally, the **Admin** and **User** sub-sections should contain the settings from the following table.

Setting	Description
Authority	Specifies the Authority to use when making OpenIdConnect calls.
Audience	Specifies the audience for any received OpenIdConnect token.
TokenValidationParameters	Specifies the parameters used to validate identity tokens. The only commonly used parameter is ValidIssuer . When ValidIssuer is not null it overrides what issuer should be validated against. For example, if the valid issuer's server is misconfigured to use http:// instead of https:// , you would need to provide the http:// url as the ValidIssuer . Generally, ValidIssuer should be left as null.

Example

```
"Authentication": {
  "JwtBearer": {
    "Admin": {
      "Authority": "https://<admin-id-domain>",
      "Audience": "<audience>",
      "TokenValidationParameters": {
        "ValidIssuer": null
      }
    },
    "User": {
      "Authority": "https://<user-id-domain>",
      "Audience": "<audience>",
      "TokenValidationParameters": {
        "ValidIssuer": "http://<user-id-domain>"
      }
    }
  }
},
```

Authorization

The **Authorization** section contains the **AllowedSubject** sub-section that controls how the server grants authorization to **Admin** and **User** accounts. The **AllowedSubject** section allows you to specify the **Admin** and **User** settings in their own sections.

The following table lists the settings available in the **Admin** and **User** sub-sections.

Setting	Description
ClaimSubjectOverride	Specifies the subject used to identify an authenticated account.

	The default is name .
AllowedSubjects	Specifies a list of subjects allowed. When AllowAllSubjects is not set to true , any authenticated account must have one of the specified subjects as its ClaimSubjectOverride to be authorized. If AllowedSubjects is empty while AllowAllSubjects is not set to true , then no accounts will be able to obtain authorization.
AllowAllSubjects	Specifies whether all subjects should be allowed. If AllowAllSubjects is set to true , then the AllowedSubjects are ignored, and any authenticated account is authorized. The default is false .

Example

```
"Authorization": {
  "AllowedSubject": {
    "Admin": {
      "ClaimSubjectOverride": null,
      "AllowedSubjects": [ "<allowed user 1>" ],
      "AllowAllSubjects": false
    },
    "User": {
      "ClaimSubjectOverride": "username",
      "AllowedSubjects": [],
      "AllowAllSubjects": false
    }
  }
},
```

Database

The **Database** setting controls which string in **ConnectionStrings** is used for database connections. Valid values are **SqlServer** and **PostgreSql**.

Example

```
"Database": "SqlServer"
```

ConnectionStrings

The **ConnectionStrings** section contains the possible connection strings to select using the **Database** setting.

The following table lists the settings available in the **ConnectionStrings** section.

Setting	Description
SqlServer	Specifies the connection string to use when " Database ": " SqlServer ". The string may only contain settings supported by a Microsoft SQL Server.
AllowedSubjects	Specifies the connection string to use when " Database ": " PostgreSql ". The string may only contain settings supported by a PostgreSQL Server.

Example

```
"ConnectionStrings": {
```

```

"SqlServer":
"Server=(localdb)\\mssqllocaldb;Database=HylandRemoteConfigurationDB;Trusted_Connection=True;MultipleActiveResultSets=true;Encrypt=true;TrustServerCertificate=false",
"PostgreSql":
"Host=localhost;Database=HylandRemoteConfigurationDB;Username=;Password=;SSLMode=VerifyFull"
},

```

PluginOptions

The **PluginOptions** section controls which configuration plugins are loaded by this server using the list specified by **PluginPaths**. Each plugin should be in its own directory. Each path may be a relative or full path. The server must have at least one plugin specified before it can run. **Note** that when running as a Windows Service with relative plugin paths, the paths are evaluated relative to **C:\Windows\System32**.

Example

```

"PluginOptions": {
  "PluginPaths": [
    "./PluginResources/<Plugin1>",
    "./PluginResources/<Plugin2>"
  ]
},

```

SecretSettings

The **SecretSettings** section controls how the server encrypts its managed configurations. Any active instances that share a database must have the same **SecretSettings**. The **DefaultSecret** setting specifies which secret to use from the **Secrets** sub-section. The **DefaultSecret** must be one of the secrets specified in the **Secrets** sub-section.

The **DefaultSecret** is required for the server to start. At startup, the server verifies that all secrets match the hashes in the database. The server will fail to start if any are missing or mismatched.

Key rotation is performed by adding a new secret and then changing the **DefaultSecret** to reference the new secret.

Secrets

The **Secrets** sub-section contains the secrets used by this server. Each element of the **Secrets** sub-section is a **<user-defined key>**. Any secret stored within the **Secrets** can never be removed or modified since they may be needed to unencrypt stored configurations.

You must specify either the **Passphrase_Base64** or the **Passphrase_File**. Failing to specify only one passphrase results in a configuration failure.

The following table lists the available **Secrets** settings.

Setting	Description
Passphrase_Base64	Specifies the base64 encoded passphrase used by this secret. The passphrase must already be encoded, and it must be 16 , 24 , or 32 bytes in length.
Passphrase_File	Specifies the path to a file containing the base64 encoded passphrase used by this secret. The passphrase must already be encoded, and it must be 16 , 24 , or 32 bytes in length.

Example

The following is an example **SecretSettings** section for your appsettings.json

```

"SecretSettings": {
  "DefaultSecret": "<user-defined-key2>",
  "Secrets": {
    "<user-defined-key1>": {
      "Passphrase_Base64": "YURlZmFlbHRQYXNzd29yZA=="
    },
    "<user-defined-key2>": {
      "Passphrase_File": "./<key2 secret file>"
    }
  }
},

```

Where **<user-defined-key1>** is an older, rotated key, and **<user-defined-key2>** is the current key.

The following are some example commands to generate random base64 passphrases using OpenSSL.

- 128-bit AESGCM (16 bytes)
`openssl rand -base64 16`
- 192-bit AESGCM (24 bytes)
`openssl rand -base64 24`
- 256-bit AESGCM (32 bytes)
`openssl rand -base64 32`

Host

The **Host** section controls the available CORS policy options for the server.

The following table lists the available **Host** settings.

Setting	Description
AllowedOrigins	Specifies the origins allowed to make requests to the server using a semicolon, <code>;</code> , delimited list. This setting must include your HRCC domain. If the domain is missing, then the HRCC will not be able to authenticate or use any of the endpoints.

Example

```

"Host": {
  "AllowedOrigins": "https:<user-client-domain>;https:<admin-client-domain>"
}

```

Hyland.Logging

The **Hyland.Logging** section controls how the server performs logging.

The following settings detail how to set up basic logging routes.

Routes

The **Routes** section is a subsection of **Hyland.Logging** and specifies the configuration information for each configured route.

<NameOfRoute>

The <**NameOfRoute**> section is a subsection of the **Routes** section and specifies the settings for each configured route. The <**NameofRoute**> name is user-defined.

The following table lists some of the common settings for the <**NameofRoute**> section.

Setting	Description
Console	Specifies logging to the console. Leave this value empty. Omit to disable console logging.
File	Specifies the path and name of the log file to where the logger writes. Omit to disable file logging. Note that when running as a Windows Service with a relative path, the path is evaluated relative to C:\Windows\System32\ .
Splunk	Specifies the address of the Splunk server. Omit to disable Splunk logging.
SplunkToken	Specifies the authentication token the system gets from the Splunk server
exclude-profiles	Specifies which profiles are not written to this route. Profiles not listed are written to the route. The include-profiles setting overrides this setting. The default is Empty List .
include-profiles	Specifies which profiles are written on this route. Profiles not listed are not written. The default is Empty List .
minimum-level	Specifies the minimum level of logging, Trace, Debug, Information, Warning, Error, Critical , or None , you want the system to log.
maximum-level	Specifies the maximum level of logging, Trace, Debug, Information, Warning, Error, Critical , or None , you want the system to log.
FileRollInterval	Specifies how often the logger should trigger a rollover event. Valid values are Minute, Hour, Day, Month, Year , and Infinite . No time-based rolling occurs when Infinite is selected. The default is Infinite
FileByteLimit	Specifies the maximum size a log file may reach before the logger stops writing to the file. If FileRollOnSize is not set to true , then the logger will not write to the file until another log rollover event occurs, or the file is deleted. The default is 1 GB .
FileRollOnSize	Specifies if the logger should rollover based on size. The default is false .
FileCountLimit	Specifies the number of rollover log files to keep. Settings FileCountLimit to "" (empty string) specifies that the logger should keep an "infinite" number of log files. The default is 31 .

OutputFormat	<p>Specifies the output format to use when logging. Valid values are json, text, and minimal.</p> <p>The default is json.</p> <p>Note that the minimal format does not include logging context data, so some logging information may be missing.</p>
--------------	--

Example

```
"Hyland.Logging": {
  "Routes": {
    "<user-defined-Console-log>": {
      "Console": "",
      "minimum-level": "Information",
      "maximum-level": "Critical"
    },
    "<user-defined-File-Log>": {
      "File": "<user-defined-path>/hracs.all.log",
      "minimum-level": "Information",
      "maximum-level": "Critical"
    }
  }
},
```

CredentialSettings

The **CredentialSettings** section controls how the server generated credentials for its clients. This setting should only be modified by someone familiar with authentication.

The following table lists the available **CredentialsSettings**.

Setting	Description
ClientCredentialSizeInBytes	<p>Specifies the number of bytes used to generate a client credential.</p> <p>The default is 64.</p>

Example

```
"CredentialSettings": {
  "ClientCredentialSizeInBytes": 64
},
```

CryptographySettings

The **CryptographySettings** section controls how the server performs cryptography. This setting should only be modified by someone familiar with cryptography.

The following table lists the available **CryptographySettings**.

Setting	Description
HashIterations	<p>Specifies the number of hash iterations performed by the server during cryptography operations. HashIterations cannot ever be decreased.</p> <p>The default is 100000.</p>

Example

```
"CryptographySettings": {
  "HashIterations": 100000
},
```

TransactionSettings

The **TransactionSettings** section controls how the server interacts with its database's transactions. These settings can have a significant impact on performance and should only be adjusted if the database experiences an unusual number of transaction failures, such as under high concurrent modification loads.

The following table lists the available **TransactionSettings**.

Setting	Description
TransactionRetryCount	Specifies the number of times to retry a database transaction using an unsigned integer. The default is 3 .
TransactionRetryDelay	Specifies the amount of time to wait before retrying a transaction. The default is 1 second, 00:00:01 .

Example

```
"TransactionSettings": {
  "TransactionRetryCount": 3,
  "TransactionRetryDelay": "00:00:01"
}
```

RouteConstraintSettings

The **RouteConstraintSettings** section allows you to specify a path prefix for **Admin**, **User**, and **Client** type endpoints to support endpoint filtering. Any filtering must be handled by external tools. All prefixes are optional. If you do not have a specific filtering rule that you need to enforce, you should leave the prefixes as empty, "", since your clients and services must be reconfigured to match any non-empty settings.

The following table lists the available endpoint types.

Setting	Description
Admin	Specifies the prefix to use for endpoints used by administrator Hyland Remote Configuration Clients. The default is "".
Client	Specifies the prefix to use for endpoints used by services consuming remote configurations. The default is "".
User	Specifies the prefix to use for endpoints used by user Hyland Remote Configuration Clients. The default is "".

Example

```
"RouteConstraintSettings": {  
  "Admin": "<user-defined-admin-route>",  
  "Client": "<user-defined-client-route>",  
  "User": "<user-defined-user-route>",  
},
```

ActiveActiveConfig

The **ActiveActiveConfig** section controls if the server runs in active-active mode. To run in active-active mode, include the **AMQPConfig** sub-section. To run in stand-alone mode, omit the **ActiveActiveConfig** section or the **AMQPConfig** sub-section. The **AMQPConfig** sub-section controls how the HRCS connects to an AMQP server to support active-active mode.

While running in active-active mode, the server will create multiple Exchanges on the server specified by the **AMQPConfig** to facilitate active-active communication. The server creates the Exchanges as needed, so they will not appear immediately.

These following are the created Exchanges.

- HRCSType-ChangedConfigRevisionEventArgs
- HRCSType-RemovedClientAccessEventArgs
- HRCSType-RemovedClientEventArgs
- HRCSType-RemovedConfigInstanceEventArgs
- HRCSType-RemovedConfigVersionEventArgs
- HRCSType-RemovedUserVisibilityEventArgs
- HRCSType-RevokedClientCredentialsEventArgs

AMQPConfig

The **AMQPConfig** section contains settings necessary to connect to an AMQP server. All settings in this section are optional aside from **Options** and its **ActiveActiveAccount** subsection for authentication. Any omitted settings will use the default behavior as described by RabbitMQ's API documentation. Note that if you want to use RabbitMQ's API default authentication, you must configure the **ActiveActiveAccount** to use **NoAuth**. AMQP servers in production environments should not use **NoAuth** and should have SSL enabled. The authorized account must have read and write permissions and must be allowed to create exchanges. Depending on your AMQP server, you may also need to give the user explicit permission to use the configured **VirtualHost**.

The following settings are available for the **AMQPConfig** section.

Setting	Description
HostName	Specifies the name of the host used by this AMQP server.
VirtualHost	Specifies the virtual host used by this AMQP server.
Port	Specifies the port number exposed by your RabbitMQ server. Type -1 to use RabbitMQ's default port number.
OperationTimeout	Specifies the number of milliseconds past the expected operation durations the server should wait for RabbitMQ operations. This setting must be between 0 and 60000 milliseconds (1 minute) or it must be completely omitted. If the setting value is 0 or omitted, then the server only waits for the expected duration of each operation. Should match the <code>[-][d'].][hh':][mm':][ss']'.ffffff]</code> timespan format.
Ssl	Specifies the SSL section contains the settings that control the SSL options for this server. For more information on the available settings, see the SSL table below.
Options	Specifies the connections option available with this server. This server only supports an ActiveActiveAccount option to control the AMQP servers auth settings. For more information on the available settings, see the Options table below.

Ssl

The **Ssl** section is a subsection of the **AMQPConfig** section and specifies the settings that control the SSL options for the configured AMQP server.

The following settings are available for the **Ssl** section.

Setting	Description
Enabled	Controls if this server should use the SSL settings during AMQP connections. If Enabled is false, then the server will not use SSL to connect to the AMQP server.
CertPath	Specifies the file path to the SSL certificate used by this server.
CertPassphrase	Specifies the password required to access the SSL certificate found at the CertPath . If your cert is not password protected, then you may omit this setting.
ServerName	Specifies the name of the server used to generate the certificate found at the CertPath .
AcceptablePolicyErrors	Specifies the flag for controlling any allowed SSL Policy errors. If none of the flags are selected, then the server will not allow any certificate errors. The following flags are available for the Errors section, RemoteCertificateNotAvailable , RemoteCertificateNameMismatch , and RemoteCertificateChainErrors .

Setting	Description												
Protocols	<p>Specifies the flags for controlling the allowed SSL protocol versions. If none of the flags are selected, then the server will not be able to connect using SSL.</p> <p>The following table lists the available flags for the Protocols section.</p> <table border="1"> <thead> <tr> <th>Setting</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>Ssl2</td> <td>Specifies the SSL 2.0 protocol. SSL 2.0 has been superseded by the TLS protocol and is provided for backward compatibility only.</td> </tr> <tr> <td>Ssl3</td> <td>Specifies the SSL 3.0 protocol. SSL 3.0 has been superseded by the TLS protocol and is provided for backward compatibility only.</td> </tr> <tr> <td>Tls</td> <td>Specifies the TLS 1.0 security protocol. The TLS protocol is defined in IETF RFC 2246.</td> </tr> <tr> <td>Tls11</td> <td>Specifies the TLS 1.1 security protocol. The TLS protocol is defined in IETF RFC 4346.</td> </tr> <tr> <td>Tls12</td> <td>Specifies the TLS 1.2 security protocol. The TLS protocol is defined in IETF RFC 5246.</td> </tr> </tbody> </table>	Setting	Description	Ssl2	Specifies the SSL 2.0 protocol. SSL 2.0 has been superseded by the TLS protocol and is provided for backward compatibility only.	Ssl3	Specifies the SSL 3.0 protocol. SSL 3.0 has been superseded by the TLS protocol and is provided for backward compatibility only.	Tls	Specifies the TLS 1.0 security protocol. The TLS protocol is defined in IETF RFC 2246.	Tls11	Specifies the TLS 1.1 security protocol. The TLS protocol is defined in IETF RFC 4346.	Tls12	Specifies the TLS 1.2 security protocol. The TLS protocol is defined in IETF RFC 5246.
Setting	Description												
Ssl2	Specifies the SSL 2.0 protocol. SSL 2.0 has been superseded by the TLS protocol and is provided for backward compatibility only.												
Ssl3	Specifies the SSL 3.0 protocol. SSL 3.0 has been superseded by the TLS protocol and is provided for backward compatibility only.												
Tls	Specifies the TLS 1.0 security protocol. The TLS protocol is defined in IETF RFC 2246.												
Tls11	Specifies the TLS 1.1 security protocol. The TLS protocol is defined in IETF RFC 4346.												
Tls12	Specifies the TLS 1.2 security protocol. The TLS protocol is defined in IETF RFC 5246.												

Options

The **Options** section contains the **ActiveActiveAccount**, which controls the authentication settings to the AMQP server.

The **ActiveActiveAccount** section contains the following.

Settings Key	Description						
Auth	<p>The authentication used by this connection option. This section contains either a NoAuth section or a BasicAuth section.</p> <p>The following table lists the settings for the Auth section.</p> <table border="1"> <thead> <tr> <th>Setting</th> <th>Description</th> </tr> </thead> <tbody> <tr> <td>NoAuth</td> <td>Represents a connection option that uses RabbitMQ's default authentication.</td> </tr> <tr> <td>BasicAuth</td> <td>Represents a connection option that uses basic authentication and includes the settings Username and Password.</td> </tr> </tbody> </table>	Setting	Description	NoAuth	Represents a connection option that uses RabbitMQ's default authentication.	BasicAuth	Represents a connection option that uses basic authentication and includes the settings Username and Password .
Setting	Description						
NoAuth	Represents a connection option that uses RabbitMQ's default authentication.						
BasicAuth	Represents a connection option that uses basic authentication and includes the settings Username and Password .						

Example

```
"ActiveActiveConfig": {
  "AMQPConfig": {
    "HostName": "<domain>",
    "VirtualHost": "",
    "Port": 5672,
    "OperationTimeout": "00:01:00",
```

