# MigratoryData Server

## Configuration Guide

*Version 6.0*
*July 12, 2018*

# Copyright Information

# Contents

# 1. Introduction

This guide describes the configuration of MigratoryData Server. It is recommended to read *MigratoryData Architecture Guide* before reading this document for a more comprehensive background.

## 1.1 Release

This guide is part of the documentation set for MigratoryData Server version 6.0.

## 1.2 Related Documents

- *MigratoryData Architecture Guide*
- *MigratoryData Installation Guide*
- *MigratoryData Performance Benchmarking Guide*
- *Developer's Guide* and *Reference Manual* of any available client library
- *Developer's Guide* and *Reference Manual* of any available extension library

# 2. Configuration File Structure

The configuration file of MigratoryData Server has comments and optional parameters besides required parameters. The optional parameters have default values. An optional parameter that is not present in the configuration file will be used with its default value.

The configuration file supports comments. A line that starts with a # character is interpreted as a comment and is ignored. A parameter can be configured with the following syntax:

```
parameter = value
```

The value of a parameter can be defined on multiple lines by postfixing each line with \. For example

```
LicenseKey = aaaabbbbcccc
```

can be written as follows:

```
LicenseKey = aaaa\
             bbbb\
             cccc
```

The configurable parameters are described in Chapter 3 and Chapter 4.

# 3. Core Parameters

The basic parameters of MigratoryData Server are described below.

## 3.1 LicenseKey

| LicenseKey | |
|---|---|
| Description | A string representing the license key |
| Default value | No default value |
| Required parameter | Required |

The license key consists of a sequence of numbers and letters. License keys are provided by MigratoryData to customers either for evaluation, development, or production usage of MigratoryData Server. To obtain a valid license key please contact MigratoryData at `support@migratorydata.com`.

## 3.2 Memory

| Memory | |
|---|---|
| Description | The maximum memory for the Java Virtual Machine (JVM) process running MigratoryData Server. The memory can be also expressed in gigabytes, megabytes, and kilobytes by using the following postfixes: GB for gigabytes, MB for megabytes, or KB for kilobytes. For example to allocate 512 megabytes for MigratoryData Server, the parameter `Memory` should be configured as follows:<br>`Memory = 512 MB` |
| Default value | No default value |
| Required parameter | Required |

In a production environment is is recommended to use at least 8192 MB memory space or more depending on the load of data on the server and how much simultaneous clients will be connected. Use *MigratoryData Benchmark Kit* to estimate the memory required for your installation.

## 3.3 Listen

| Listen | |
|---|---|
| Description | A comma separated list of addresses to listen for client connections |
| Default value | No default value |
| Required parameter | Optional |

The format of the addresses is `"IP:Port"` (e.g. `192.168.1.1:80`) or `"Name:Port"` (e.g. `push.example.com:80`). IPv6 addresses must be enclosed in square brackets (e.g. `[2001:db8::a00:20ff:fea7:ccea]:80`).

If you specify an address without a port, the default port `80` will be used.

By specifying an IP address, MigratoryData Server will bind only to that IP address. The wildcard address (`*`) will enable MigratoryData Server to bind to all available IP addresses of the machine.

If MigratoryData Server is installed on a machine that has a firewall enabled, then you will need to allow in firewall the access to the addresses and ports configured by this parameter.

> **Note** — For web applications, in a production environment, MigratoryData Server requires a standard web server – such as Apache or Nginx – to serve the static resources like html and css files, images etc. It is recommended that the DNS name of the address provided by the `Listen` parameter to share the same sub-domain with the DNS name of the standard web server.
>
> For example, if your standard web server is accessible at `www.example.com`, then an example of optimal configuration for MigratoryData Server will be:
>
> ```
> Listen = push.example.com:80.
> ```
>
> but the following configuration will not be optimal:
>
> ```
> Listen = push.example.org:80.
> ```
>
> because the sub-domain `example.com` of the web server is different from `example.org`, which is the sub-domain configured for MigratoryData Server in this example.

## 3.4 ListenEncrypted

| ListenEncrypted | |
|---|---|
| Description | A comma separated list of addresses to listen for encrypted client connections |
| Default value | No default value |
| Required parameter | Optional |

The same conventions applies as for the `Listen` parameter described in Section 3.3, except the fact that if you specify an address without a port, the default port 443 will be used.

> **Note** — In a production deployment, it is recommend to use encrypted client connections in order to securely deliver messages to clients but also to prevent certain security solutions to inspect the messaging traffic and block it.

## 3.5 KeyStore

| KeyStore | |
|---|---|
| Description | File with SSL certificates for encrypted connections |
| Default value | No default value |
| Required parameter | Required if at least one of the following parameters is configured:<br>• `ListenEncrypted`<br>• `MonitorJMX.Encryption` is set on `true`<br>• `MonitorHTTP.Encryption` is set on `true` |

The keystore file must be configured using absolute paths. A value example for this parameter is as follows:

| Example of KeyStore | Platform |
|---|---|
| `KeyStore = /some/path/mykeystore.jks` | For Linux/Unix |
| `KeyStore = C:/some/path/mykeystore.jks` | For Windows |

The keystore must contain a SSL certificate for each address used in the configuration of the following parameters:

| | |
|---|---|
| `ListenEncrypted` | |
| `MonitorJMX.Listen` | provided that `MonitorJMX.Encryption` is set on `true` |
| `MonitorHTTP.Listen` | provided that `MonitorHTTP.Encryption` is set on `true` |

Table 3.1: Parameters Used to Configure Encrypted Connections

### 3.5.1 How to add a certificate to the keystore

Suppose the following DNS address `push.example.com` resolves to the IP address `192.168.1.1`, and vice-versa, the IP address `192.168.1.1` resolves to the DNS address `push.example.com`.

If an address appears in the configuration of any of the parameters used to define encrypted connections (see Table 3.1), then the keystore file must contain a SSL certificate for that address. If the address is specified by its DNS name, say `push.example.com`, then its certificate entry in the keystore must have as alias the string `push.example.com`. If the address is specified by its IP address, say `192.168.1.1`, then its certificate entry in the keystore must have as alias the string `192.168.1.1`.

To create a certificate for an address in the keystore, there are two possibilities:

1. Use a self-signed certificate for that address
2. Use a certificate signed by a Certificate Authority (CA)

> **Tip** — Self-signed certificates can be used in development. In a production environment always use certificates signed by a Certificate Authority.

In the next two subsections we suppose the keystore file is named `mykeystore.jks` and the address for which a certificate should be added in the keystore is specified either by DNS name as `push.example.com` or by IP address as `192.168.1.1`.

#### 3.5.1.1 Adding a self-signed certificate of an address to the keystore

Enter one of following commands on a single line without line breaks depending on how was specified the address in the configuration file:

| | |
|---|---|
| Address is used in configuration<br>by DNS name as `push.example.com` | `keytool`<br>`-genkeypair`<br>`-dname "CN=`*push.example.com*`"`<br>`-alias` *push.example.com*<br>`-keyalg RSA`<br>`-validity 3600`<br>`-keystore` *mykeystore.jks* |
| Address is used in configuration<br>by IP address as `192.168.1.1` | `keytool`<br>`-genkeypair`<br>`-dname "CN=`*192.168.1.1*`"`<br>`-alias` *192.168.1.1*<br>`-keyalg RSA`<br>`-validity 3600`<br>`-keystore` *mykeystore.jks* |
| Address is used in configuration<br>for encrypted JMX monitoring | `keytool`<br>`-genkeypair`<br>`-dname "CN=`**jmx**`"`<br>`-alias` **jmx**<br>`-keyalg RSA`<br>`-validity 3600`<br>`-keystore` *mykeystore.jks* |
| Address is used in configuration<br>for encrypted HTTP monitoring | `keytool`<br>`-genkeypair`<br>`-dname "CN=`**http**`"`<br>`-alias` **http**<br>`-keyalg RSA`<br>`-validity 3600`<br>`-keystore` *mykeystore.jks* |

You will be asked to set a password for the keystore if the file `mykeystore.jks` does not exist, or to enter the keystore password if the keystore file already exists and contains other certificate entries. This password must be used to configure the parameter `KeyStorePassword` (see Section 3.6).

#### 3.5.1.2 Adding a CA-signed certificate of an address to the keystore

Suppose you obtained the following signed certificate file `push.example.com.crt` from a Certificate Authority for the domain `push.example.com` and suppose `push.example.com.key` is the file containing its corresponding private key.

In order to add this CA-signed certificate to the keystore, follow these steps:

1. If the Certificate Authority provides you an intermediate certificate in addition to the signed certificate `push.example.com.crt`, then you will have to chain your signed certificate with the intermediary certificate. If you didn't receive an intermediary certificate, just skip this step.

   To chain your signed certificate `push.example.com.crt` with the intermediary certificate, you should first append to your signed certificate the intermediary certificate (say `intermediary.crt`), and then append the certificate signing request sent by you to the CA authority (say `push.example.com.csr`). On a Unix-like operating system, use the following command:

   ```
   cat intermediary.crt push.example.com.csr >> push.example.com.crt
   ```

   On Windows, use the command:

   ```
   type intermediary.crt push.example.com.csr >> push.example.com.crt
   ```

2. Convert the CA-signed certificate to the PKCS#12 standard by entering the following command on a single line without line breaks:

   ```
   openssl
      pkcs12
      -export
      -in push.example.com.crt
      -inkey push.example.com.key
      -out push.example.com.pkcs12
   ```

   This will produce the file *push.example.com.pkcs12*; the password for the this new file must be the same like that used for the keystore.

3. Add the pkcs#12 certificate obtained in the previous step to the keystore by entering the following command on a single line without line breaks:

   ```
   keytool
      -importkeystore
      -srckeystore push.example.com.pkcs12
      -srcstoretype PKCS12
      -deststoretype JKS
      -destkeystore mykeystore.jks
   ```

   This will insert a new entry in the keystore file `mykeystore.jks` with the default alias **1**.

4. Rename the default certificate alias **1** in the keystore by entering one of following commands on a single line without line breaks depending on how was specified the address in the configuration file:

| | |
|---|---|
| Address is used in configuration by DNS name as `push.example.com` | `keytool`<br>`-v`<br>`-keystore` *mykeystore.jks*<br>`-changealias`<br>`-alias 1`<br>`-destalias` *push.example.com* |
| Address is used in configuration by IP address as `192.168.1.1` | `keytool`<br>`-v`<br>`-keystore` *mykeystore.jks*<br>`-changealias`<br>`-alias 1`<br>`-destalias` *192.168.1.1* |
| Address is used in configuration for encrypted JMX monitoring | `keytool`<br>`-v`<br>`-keystore` *mykeystore.jks*<br>`-changealias`<br>`-alias 1`<br>`-destalias` **jmx** |
| Address is used in configuration for encrypted HTTP monitoring | `keytool`<br>`-v`<br>`-keystore` *mykeystore.jks*<br>`-changealias`<br>`-alias 1`<br>`-destalias` **http** |

## 3.6 KeyStorePassword

| KeyStorePassword | |
|---|---|
| Description | The password to access the keystore |
| Default value | No default value |
| Required parameter | Required if the parameter `KeyStore` is configured |

Configure this parameter with the password used when you created the keystore file. The keystore file is created when adding the first certificate entry to it as explained in Section 3.5.1.

## 3.7 Monitor

| Monitor | |
|---|---|
| Description | Monitoring type. Possible values are JMX and HTTP |
| Default value | No default value |
| Required parameter | Optional |

You can configure either JMX monitoring, or HTTP monitoring, or both. Use comma separated values to configure both monitoring types as follows:

```
Monitor = JMX, HTTP
```

## 3.8 MonitorUsername

| MonitorUsername | |
|---|---|
| Description | Username for monitoring access |
| Default value | admin |
| Required parameter | Required if at least one of the following parameters is configured:<br>• `MonitorJMX.Authentication` is set on `true`<br>• `MonitorHTTP.Authentication` is set on `true` |

## 3.9 MonitorPassword

| MonitorPassword | |
|---|---|
| Description | Password for monitoring access |
| Default value | pass |
| Required parameter | Required if at least one of the following parameters is configured:<br>• `MonitorJMX.Authentication` is set on `true`<br>• `MonitorHTTP.Authentication` is set on `true` |

## 3.10 MonitorJMX.Listen

| MonitorJMX.Listen | |
|---|---|
| Description | Address used to listen for JMX compatible clients |
| Default value | No default value |
| Required parameter | Optional |

The format of this addresse is `"IP:Port"` (e.g. `192.168.1.1:3000`) or `"Name:Port"` (e.g. `push.example.com:3000`). IPv6 addresses must be enclosed in square brackets (e.g. `[2001:db8::a00:20ff:fea7:ccea]:3000`).

> **Note** — In order to access the JMX monitoring from a remote machine, please check your firewall settings so that the network traffic from the remote machine is allowed to the address configured by this parameter `MonitorJMX.Listen`.

The `jconsole` utility that is freely available with Oracle's Java Development Kit (JDK) can be used to connect to the JMX monitoring service of MigratoryData Server. Also there are many JMX commercial tools that provide enhanced functionality like dashboards and database persistence that can be used to connect to the JMX monitoring service of MigratoryData Server.

> **Caution** — The `jconsole` utility has a known issue when connecting remotely to a JMX service running on Linux. To avoid this issue, execute the following command on the Linux machine where MigratoryData Server runs:
>
> ```
> hostname -i
> ```

The command above should return the address used in the configuration of the parameter `MonitorJMX.Listen`. If it reports somthing like `127.0.0.1`, `jconsole` would not be able to connect to MigratoryData Server. To fix this issue, edit `/etc/hosts` so that `hostname` resolves to the the address used in `MonitorJMX.Listen`.

## 3.11 MonitorJMX.Authentication

| MonitorJMX.Authentication | |
|---|---|
| Description | Enable or disable authentication for the JMX monitoring service |
| Default value | No default value |
| Required parameter | Optional |

This parameter can have two values: `true` or `false`. If set on `true` then, in order to access the monitoring via JMX, you will need to use the username configured with the parameter `MonitorUsername` and the password configured with the parameter `MonitorPassword`.

## 3.12 MonitorJMX.Encryption

| MonitorJMX.Encryption | |
|---|---|
| Description | Enable or disable encryption for the JMX monitoring service |
| Default value | No default value |
| Required parameter | Optional |

This parameter can have two values: `true` or `false`. If set on `true`, then a JMX client will connect to MigratoryData Server through a SSL/TLS encrypted connection. The use of an encrypted connection is especially recommended for the JMX remote monitoring from an insecure network, including Internet.

If `MonitorJMX.Encryption` is set on `true`, then the address used in the configuration of the parameter `MonitorJMX.Listen` must have an certificate entry in the keystore file defined by the parameter `KeyStore`. The certificate entry in the keystore must be created with the alias **jmx** as explained in Section 3.5.1.

### 3.12.1 Secure JMX monitoring over insecure networks

In this subsection it is assumed that the JMX client used to connect to MigratoryData Server is `jconsole`. For other JMX clients you can also use the steps below, but you may have to adapt the configuration to suit your specific JMX client.

Supposing the file name of the keystore defined by `KeyStore` paramater is `mykeystore.jks`, and supposing the keystore `mykeystore.jks` includes an certificate entry for the address configurated by the parameter `MonitorJMX.Listen` having as alias **jmx** as explained in Section 3.5.1. Then, use the following steps to establish a secure JMX connection:

1. Create a truststore for the JMX client. The truststore is a special keystore that can verify the trusted SSL certificates. The trustsore will be used by the JMX client. Supposing that the file name used for the truststore is `mytruststore.jks`, enter the following two commands each on a single line without line breaks to create the truststore:

   ```
   keytool
      -export
      -alias jmx
      -keystore mykeystore.jks
      -rfc
      -file temp.cer

   keytool
      -import
      -alias jmx
      -file temp.cer
      -keystore mytruststore.jks
   ```

   For the first command above use the password of the keystore as defined by the parameter `KeyStorePassword`. For the second command above use a new password, say `mytruststore-password`.

2. Generate a new keystore for the JMX client and add to it a certificate entry with the alias **jmx**. To do so, supposing the file name for the new keystore is `clientkeystore.jks`, enter the following command on a single line without line breaks:

   ```
   keytool
      -genkeypair
      -alias jmx
      -keyalg RSA
      -validity 3600
      -keystore clientkeystore.jks
   ```

For this command use a new password, say `clientkeystore-password`.

3. To securely connect to the JMX monitoring service of MigratoryData Server, enter the following command on a single line without line breaks:

```
jconsole
    -J-Djavax.net.ssl.keyStore=clientkeystore.jks
    -J-Djavax.net.ssl.keyStorePassword=clientkeystore-password
    -J-Djavax.net.ssl.trustStore=mystruststore.jks
    -J-Djavax.net.ssl.trustStorePassword=mytruststore-password
```

## 3.13 MonitorHTTP.Listen

| MonitorHTTP.Listen | |
|---|---|
| Description | Address used to listen for HTTP monitoring clients |
| Default value | No default value |
| Required parameter | Optional |

The format of this addresse is `"IP:Port"` (e.g. `192.168.1.1:8808`) or `"Name:Port"` (e.g. `push.example.com:8808`). IPv6 addresses must be enclosed in square brackets (e.g. `[2001:db8::a00:20ff:fea7:ccea]:8808`).

> **Note** — In order to access the HTTP monitoring from a remote machine, please check your firewall settings so that the network traffic from the remote machine is allowed to the address configured by this parameter `MonitorHTTP.Listen`.

### 3.13.1 Accessing the HTTP Monitoring Service

Supposing you have the following monitoring related configuration:

```
Monitor = HTTP
MonitorUsername = admin
MonitorPassword = pass
MonitorHTTP.Listen = push.example.com:8808
MonitorHTTP.Authentication = true
MonitorHTTP.Encryption = false
```

Then you can monitor MigratoryData Server by opening the following URL:

`http://`*push.example.com:8808*`/stats?user=`*admin*`&password=`*pass*

Opening the URL above will produce an output with the following format:

```
<fieldname1>:<value1> <fieldname2>:<value2> ...  <fieldnameN>:<valueN>
```

where each field correspond to one of the following **statistics**:

- Average
- Standard Deviation
- Maximum

applied to one of the following **indicators**:

- The number of connected clients
- The number of connections per second
- The number of disconnections per second
- The number of incoming messages per second received from publishers
- The number of outgoing messages per second sent to clients
- The number of incoming bytes per second received from publishers
- The number of outgoing bytes per second sent to clients

for one of the following **period** of time:

- Current
- Since startup
- Last minute, last 5 minutes, and last 15 minutes
- Last hour, last 5 hours, and last 15 hours
- Last day, last 5 days, and last 15 days
- Last month, last 5 months, and last 15 months

#### 3.13.1.1 XML and JSON Output Format

You can also retrieve data in XML and JSON format. Please append a `view` GET parameter to the URL above with the value `xml` or `json`. For example, to retrieve monitoring data in XML format use a URL as follows:

`http://`*push.example.com:8808*`/stats?user=`*user*`&password=`*pass*`&view=xml`

#### 3.13.1.2 Filters

You can filter the monitoring data by *indicator* and/or *statistic* and/or *period* of time. To do so add to the URL above one or more of the GET parameters listed in Table 3.2.

For example to retrieve the maximum number of concurrent clients in the last 15 minutes, use the following URL written on a single line without line breaks:

`http://`*push.example.com:8808*`/stats?`
`user=`*user*`&`
`password=`*pass*`&`
`indicator=ConnectedSessions&`
`statistic=MAX&`
`period=Last.15.Minute`

Another example. To retrieve the average for all indicators for all periods of time available, use a URL as follows:

`http://`*push.example.com:8808*`/stats?user=`*user*`&password=`*pass*`&statistic=AVG`

| GET Parameter | Possible Value | Description |
|---|---|---|
| indicator | ConnectedSessions | The number of connected clients |
| | InBytesPerSecond | The number of incoming bytes per second received from publishers |
| | InPublishMessagesPerSecond | The number of incoming messages per second received from publishers |
| | OutBytesPerSecond | The number of outgoing bytes per second sent to clients |
| | OutPublishMessagesPerSecond | The number of outgoing messages per second sent to clients |
| | SessionConnectionsPerSecond | The number of connections per second |
| | SessionDisconnectionsPerSecond | The number of disconnections per second |
| statistic | AVG | Average |
| | STDEV | Standard Deviation |
| | MAX | Maximum |
| period | Current | Current |
| | Last.1.Minute | Last 1 minute |
| | Last.5.Minute | Last 5 minutes |
| | Last.15.Minute | Last 15 minutes |
| | Last.1.Hour | Last 1 hour |
| | Last.5.Hour | Last 5 hours |
| | Last.15.Hour | Last 15 hours |
| | Last.1.Day | Last 1 day |
| | Last.5.Day | Last 5 days |
| | Last.15.Day | Last 15 days |
| | Last.1.Month | Last 1 month |
| | Last.5.Month | Last 5 months |
| | Last.15.Month | Last 15 months |
| | SinceStartup | Since startup |

Table 3.2: Filters

#### 3.13.1.3 Secure HTTP monitoring over insecure networks

To securely monitor MigratoryData Server over an insecure network such as Internet, configure

```
MonitorHTTP.Encryption = true
```

See Section 3.15 for more details. Then use a URL as follows:

`https://`*push.example.com:8808*`/stats?user=`*user*`&password=`*pass*

## 3.14 MonitorHTTP.Authentication

| MonitorHTTP.Authentication | |
|---|---|
| Description | Enable or disable authentication for the HTTP monitoring service |
| Default value | No default value |
| Required parameter | Optional |

This parameter can have two values: `true` or `false`. If set on `true` then, in order to access the monitoring via HTTP, you will need to use the username configured with the parameter `MonitorUsername` and the password configured with the parameter `MonitorPassword`.

## 3.15 MonitorHTTP.Encryption

| MonitorHTTP.Encryption | |
|---|---|
| Description | Decide whether to use SSL/TLS encryption when connecting to the HTTP monitoring service |
| Default value | No default value |
| Required parameter | Optional |

This parameter can have two values: `true` or `false`. If set on `true`, then a HTTP monitoring client will connect to MigratoryData Server through a SSL/TLS encrypted connection. The use of an encrypted connection is especially recommended for the HTTP remote monitoring from an insecure network, including Internet.

If `MonitorHTTP.Encryption` is set on `true`, then the address used in the configuration of the parameter `MonitorHTTP.Listen` must have an certificate entry in the keystore file defined by the parameter `KeyStore`. See Section 3.5.1 to learn how to add a certificate to the keystore.

## 3.16 LogFolder

| LogFolder | |
|---|---|
| Description | The folder where the logs will be written |
| Default value | `logs` |
| Required parameter | Optional |

If not configured, MigratoryData Server will use the default folder `logs` relative to the directory path used to start MigratoryData Server.

The log folder can be configured using absolute paths. A value example for this parameter is as follows:

| Example of log folder | Platform |
|---|---|
| `LogFolder = /some/path/mylogs` | For Linux/Unix |
| `LogFolder = C:/some/path/mylogs` | For Windows |

## 3.17 LogLevel

| LogLevel | |
|---|---|
| Description | The level of verbosity of the messages recorded in the logs |
| Default value | INFO |
| Required parameter | Optional |

The following levels are available:

- `TRACE` (most verbose)
- `DEBUG`
- `INFO` (recommended for production use)
- `WARN`
- `ERROR` (least verbose)

## 3.18 LogRotateLimit

| LogRotateLimit | |
|---|---|
| Description | The maximum capacity of a log file in kilobytes (KB), megabytes (MB), or gigabytes (GB) |
| Default value | 10 MB |
| Required parameter | Optional |

If the log file reaches the capacity provided by this parameter, then MigratoryData Server will automatically create a new log file. The previous log files are preserved on disk up to the number of log files defined by the parameter `LogRotateFileCount`.

## 3.19 LogRotateTime

| LogRotateTime | |
|---|---|
| Description | The time interval at which a new log file will be created in minutes (m), hours (h), days (D), weeks (W), months (M), or years (Y). |
| Default value | No default value |
| Required parameter | Optional |

For example, in order to record the logs in a separate file every day use:

```
LogRotateTime = 1 D
```

To record the logs in separate file every 4 hours use:

```
LogRotateTime = 4 h
```

The previous log files are preserved on disk up to the number of log files defined by the parameter `LogRotateFileCount`.

> **Note** — This parameter takes precedence over the parameter `LogRotateLimit`. Therefore, if the parameter `LogRotateTime` is configured, then the configuration of the parameter `LogRotateLimit` is ignored.

## 3.20 LogRotateFileCount

| LogRotateFileCount | |
|---|---|
| Description | Limit the number of historical log files created by log rotation |
| Default value | 100 |
| Required parameter | Optional |

If the number of log files produced by log rotation defined by the parameters `LogRotateTime` or `LogRotateLimit` reaches the value of this parameter, then the oldest log file is removed whenever a new log file is created such that the total number of logs files will not exceed the value of this parameter.

## 3.21 Audit.Access

| Audit.Access | |
|---|---|
| Description | Enable/disable audit access events |
| Default value | false |
| Required parameter | Optional |

Set this parameter on `true` to receive audit access events (such as connect/disconnect events and subscribe/unsubscribe events) into your custom extension built with MigratoryData's Extension Audit API and plugged into the MigratoryData server.

## 3.22 Audit.Publish

| Audit.Publish | |
|---|---|
| Description | Enable/disable audit message events |
| Default value | false |
| Required parameter | Optional |

Set this parameter on `true` to receive audit message events into your custom extension built with MigratoryData's Extension Audit API and plugged into the MigratoryData server. A message event is triggered whenever a new message is accepted by the server.

## 3.23 Audit.Cache

| Audit.Cache | |
|---|---|
| Description | Enable/disable audit cache events |
| Default value | false |
| Required parameter | Optional |

Set this parameter on `true` to receive audit cache events into your custom extension built with MigratoryData's Extension Audit API and plugged into the MigratoryData server. A cache event is triggered whenever a message is added to or accessed or deleted from the cache.

## 3.24 Audit.Stats

| Audit.Stats | |
|---|---|
| Description | Enable/disable audit stats events |
| Default value | false |
| Required parameter | Optional |

Set this parameter on `true` to receive audit stats events (such as connection/disconnection rate, number of clients, incoming / outgoing message throughput) into your custom extension built with MigratoryData's Extension Audit API and plugged into the MigratoryData server.

## 3.25 DocumentRoot

| DocumentRoot | |
|---|---|
| Description | The folder from which MigratoryData Server will serve files |
| Default value | `html` |
| Required parameter | Optional |

This parameter is optional, if not supplied the default folder `html` relative to the directory path used to start MigratoryData Server is used.

**Note** — In a production environment, the parameter `DocumentRoot` should be disabled.

In a production environment, the files of a web application built with MigratoryData APIs should be installed in a standard web server such as Apache or Nginx. MigratoryData Server has a limited capability to serve web pages, which is only offered to facilitate the development process. The purpose of MigratoryData Server is to push real-time data. Thus, use a web server to provide any static resources such as html or css files, and images.

## 3.26 DocumentRoot.Listen

| DocumentRoot.Listen | |
|---|---|
| Description | A comma separated list of addresses to listen for clients requesting static resources of the folder defined by the parameter DocumentRoot. |
| Default value | `html` |
| Required parameter | Optional |

The format of the addresses is `"IP:Port"` (e.g. `192.168.1.1:8800`) or `"Name:Port"` (e.g. `push.example.com:8800`). IPv6 addresses must be enclosed in square brackets (e.g. `[2001:db8::a00:20ff:fea7:ccea]:8800`).

If you specify an address without a port, the default port 8800 will be used. You can also use the wildcard address (*) to listen on all network interfaces of the machine (e.g. `*:8800`).

**Note** — As mentioned in the note of the parameter DocumentRoot, the functionality provided by DocumentRoot and this parameter should be used only for testing and development purposes. In production, this parameter should be commented out.

## 3.27 ClusterPassword

| ClusterPassword | |
|---|---|
| Description | Each MigratoryData server in a cluster communicates with the other MigratoryData servers in the cluster, including with itself. This parameter defines the password used by each cluster member for connecting to the other cluster members, including to itself. |
| Default value | No default value |
| Required parameter | Required |

This password is used only by the MigratoryData servers that form a cluster to connect each other. This password is not used for clients. In order to allow or deny subscriptions and publications for clients, you can enable authorization via the parameters `Authorization.Type` and `Authorization.Password`.

## 3.28 ClusterDeliveryMode

| ClusterDeliveryMode | |
|---|---|
| Description | Choose between Standard Message Delivery and<br>Guaranteed Message Delivery |
| Default value | Standard |
| Required parameter | Optional |

Define the quality-of-service level for message delivery that your cluster of MigratoryData servers will use. The possible values are: "Standard" (Standard Message Delivery) and "Guaranteed" (Guaranteed Message Delivery)

Using Standard Message Delivery, in the case of a failover reconnection, the client will get the latest (most recent) message available at the moment of the reconnection for each of its subscribed subjects.

Using Guaranteed Message Delivery, in the case of a failover reconnection, the client will get not only the latest (most recent) message for each of its subscribed subjects, but it will also get all the potential messages published during the failover period for each of its subscribed subjects.

See *MigratoryData Architecture Guide*, chapter *Guaranteed Message Delivery*, for a complete discussion.

## 3.29 ClusterMemberListen

| ClusterMemberListen | |
|---|---|
| Description | Each MigratoryData server in a cluster communicates with the other MigratoryData servers in the cluster, including with itself. This parameter represents the address used to listen for incoming connections from the other cluster members, including from itself. |
| Default value | No default value |
| Required parameter | Required |

The format of this listen address is: "IP_Address:Port" (e.g. 192.168.0.1:8801) or "DNS_Name:Port" (e.g. push.example.com:8801) IPv6 addresses must be enclosed in square brackets. For example [2001:db8::a00:20ff:fea7:ccea]:8801.

> **Note** – The port defined by this parameter must be allowed by the firewall for incoming connections from all MigratoryData servers of the cluster.
>
> In addition, the four consecutive port numbers starting with the port number defined by this parameter must be allowed by the firewall for incoming connections from all MigratoryData servers of the cluster.
>
> For example, supposing that this parameter is configured as follows:
>
> ClusterMemberListen = push.example.com:8801
>
> Then the ports 8801, 8802, 8803, 8804, 8805 must be allowed by the firewall for the internal cluster communication between its members.

## 3.30 ClusterMembers

| ClusterMembers | |
|---|---|
| Description | Define the cluster by specifying its members. |
| Default value | No default value |
| Required parameter | Required |

Each cluster member is specified by the listen address defined by its parameter `ClusterMemberListen`.

For developing and testing purposes, you can deploy a cluster of MigratoryData servers where its members run on the same machine. For example, to define a cluster of three MigratoryData servers all running on the same machine, use

```
ClusterMembers = 192.168.1.1:7701, 192.168.1.1:8801, 192.168.1.1:9901
```

For production purposes, you should deploy a cluster of MigratoryData servers where each member runs on a different machine. For example, to define a production cluster of three MigratoryData servers, use something like:

```
ClusterMembers = 192.168.1.1:8801, 192.168.1.2:8801, 192.168.1.3:8801
```

> **Note** — The cluster definition must be **identical** in all cluster members (the order of the listen addresses must be preserved in the configuration of each cluster member)

## 3.31 ClusterMemberCoordinationFolder

| ClusterMemberCoordinationFolder | |
|---|---|
| Description | Folder used to store information related to cluster coordination |
| Default value | `coordination_logs` |
| Required parameter | Optional |

If not configured, MigratoryData Server will use the default folder `coordination_logs` relative to the directory path used to start the MigratoryData server.

The coordination log folder can be configured using absolute paths. A value example for this parameter is as follows:

| Example of transactions folder | Platform |
|---|---|
| `ClusterMemberCoordinationFolder = /some/path/coordination_logs` | For Linux/Unix |
| `ClusterMemberCoordinationFolder = C:/some/path/coordination_logs` | For Windows |

> **Note** — It is recommended to configure this folder on a local disk instead of a network-attached disk

## 3.32 Authorization.Type

| Authorization.Type | |
|---|---|
| Description | Authorization type. The possible values are `None`, `Basic`, and `Custom` |
| Default value | `Basic` |
| Required parameter | Optional |

- **None authorization** allows any client to connect, subscribe to any topic, and publish messages with any topic.
- **Basic authorization** allows any client to connect and subscribe to any topic, however, publication is allowed only from the clients which authenticate with the password defined by the parameter `Authorization.Password`.
- **Custom authorization** allows you to define your own authorization rules using a simple extension API. To learn how to build and deploy an authorization extension to be plugged into MigratoryData Server, please refer to the documentation of *MigratoryData Authorization Extension API*.

  In addition, the parameter `PublishAllowFromAddressList` can be used to allow publishing clients only from a set of addresses.

## 3.33 Authorization.Password

| Authorization.Password | |
|---|---|
| Description | Password for client authorization |
| Default value | No default value |
| Required parameter | Required if the `Authorization.Type` is `Basic` |

If the `Authorization.Type` is `Basic`, then only the clients which authenticate with the password defined by this parameter are allowed to publish messages.

If the `Authorization.Type` type is `Custom`, then this parameter is ignored, and the authorization is performed according the authorization rules defined by your extension built with *MigratoryData Authorization Extension API*.

## 3.34 RunAsUser

| RunAsUser | |
|---|---|
| Description | Run MigratoryData Server as non-root (normal user) |
| Default value | No default value |
| Required parameter | Optional |

**Note** — This parameter is available only for Linux.

Supposing "migratorydata" is an existing normal user. Configure MigratoryData Server as follows:

```
RunAsUser = migratorydata
```

Now, start the push server as root (this is necessary to be able to bind on the privileged ports 80 or 443). Please note that while running as root, MigratoryData Server will not accept any client connections. Then, MigratoryData Server will drop the root privileges (using the system call `setuid`) and will automatically switch to the normal user "migratorydata". Only at this time, MigratoryData Server will start to accept client connections.

## 3.35 PublishAllowFromAddressList

| PublishAllowFromAddressList | |
|---|---|
| Description | Define the list of IP addresses allowed for message publication. |
| Default value | No default value |
| Required parameter | Optional |

If this parameter is configured, then the MigratoryData server will accept message publications only from clients running on any of the IP addresses defined by this parameter.

If this parameter is not set, message publication will be allowed from any client provided however that the client is allowed by the authorization mechanism you defined with the parameter `Authorization.Type`.

**Note** — Use only dotted-decimal notation for the IP addresses (no DNS names) to specify the list of IP addresses. For example:

```
PublishAllowFromAddressList = 192.168.1.100, 192.168.1.101
```

# 4. Advanced Parameters

The advanced parameters of MigratoryData Server are described below.

## 4.1 ClusterCommunication.IoThreads

| ClusterCommunication.IoThreads | |
|---|---|
| Description | Define the number of threads for inter-cluster communication |
| Default value | 2 |
| Required parameter | Optional |

This parameter defined the number of threads for inter-cluster communication. These threads dedicated for messaging among the cluster member and not shared with messaging with clients.

## 4.2 Native.Ssl

| Native.Ssl | |
|---|---|
| Description | Use OpenSSL rather than Java SSL for encrypted connections |
| Default value | false |
| Required parameter | Optional |

The requirements for this option is to use Linux and have installed OpenSSL and Apache Portable Runtime (APR) libraries. To install these libraries you can use for example:

- RedHat/Centos: `yum install apr openssl`
- Debian/Ubuntu: `sudo apt-get install libapr1 openssl`

## 4.3 MaxCachedMessagesPerSubject

| MaxCachedMessagesPerSubject | |
|---|---|
| Description | The maximum number of messages to be cached for each topic |
| Default value | 1000 |
| Required parameter | Optional |

The maximum number of most recent messages to be stored in memory for each topic.

> **Note** — This parameter applies only if the *Guaranteed Message Delivery* feature is enabled (see parameter `ClusterDeliveryMode`).

## 4.4 CacheExpireTime

| CacheExpireTime | |
|---|---|
| Description | The time in seconds before a cached message expires |
| Default value | 180 |
| Required parameter | Optional |

Messages are removed continuously from the cache of each topic, however each message is held in the cache at least the number of seconds defined by this parameter.

> **Note** — This parameter applies only if the *Guaranteed Message Delivery* feature is enabled (see parameter `ClusterDeliveryMode`).

## 4.5 Workgroups

| Workgroups | |
|---|---|
| Description | The number of groups of clients |
| Default value | The number of total CPU cores available |
| Required parameter | optional |

In order to better scale on multiprocessor servers the incoming users are separated in groups. This parameter configures the number of groups (every group has a dedicated thread). If not supplied the total CPU cores available is the default value. In most situations it is not recommended to modify the default value.

## 4.6 IoThreads

| IoThreads | |
|---|---|
| Description | The number of threads used for I/O processing |
| Default value | The number of total CPU cores available |
| Required parameter | Optional |

If not supplied the number of total CPU cores available is the default value. In most situations it is not recommended to modify the default value.

## 4.7 Stats.LogInterval

| Stats.LogInterval | |
|---|---|
| Description | Time interval in seconds to produce monitoring stats |
| Default value | 60 |
| Required parameter | Optional |

The minimum value of this parameter is 5 (seconds).

## 4.8 MaxBatchingSpace

| MaxBatchingSpace | |
|---|---|
| Description | The maximum size of the batching in bytes |
| Default value | 0 |
| Required parameter | Optional |

If this parameter is not configured or configured with the default `0` value, it means space-based batching is disabled.

*Batching* is the process of collecting messages together for a period of time or until a total size is reached before sending them in a single I/O operation to a client.

To enable the Batching feature, a pre-configured time period and/or a pre-configured size should be configured with the parameters `MaxBatchingTime` and respectively `MaxBatchingSpace`. Once enabled, MigratoryData Server will not send individually every message to the client, instead it will send messages in batches, thus MigratoryData Server will perform a single I/O network operation for a single batch (that might contain several messages).

Depending on your use case, batching can help optimize network I/O, bandwidth usage, and paradoxically even latency as explained in the following.

Batching implies that individual messages will have to wait until the batching time expires or until the batching size is reached (whichever comes first, if both parameters are enabled) and then the whole bunch of messages grouped together in a batch will be sent on the network having as final destination the client.

When the throughput of messages is high, the time spent on network I/O without batching increases significantly and the message latency without batching is actually worse than the latency with batching.

## 4.9 MaxBatchingTime

| MaxBatchingTime | |
|---|---|
| Description | The maximum time of the batching in milliseconds |
| Default value | 0 |
| Required parameter | Optional |

If this parameter is not configured or configured with the default `0` value, it means time-based batching is disabled.

See the parameter `MaxBatchingSpace` to learn more about batching.

## 4.10 CipherListEnabled

| CipherListEnabled | |
|---|---|
| Description | Enable one or more SSL ciphers in addition to the default JVM ciphers. |
| Default value | No default value |
| Required parameter | Optional |

The JVM supports a number of ciphers as listed in:

https://docs.oracle.com/javase/8/docs/technotes/guides/security/StandardNames.html

Some of these ciphers are enabled by default. Use this parameter to enable one or more ciphers not enabled by default. For example, to enable the ciphers `TLS_DHE_RSA_WITH_AES_128_CBC_SHA` and `TLS_DHE_DSS_WITH_AES_128_CBC_SHA`, configure this parameter as follows:

```
CipherListEnabled = TLS_DHE_RSA_WITH_AES_128_CBC_SHA, TLS_DHE_DSS_WITH_AES_128_CBC_SHA
```

## 4.11 CipherListExcluded

| CipherListExcluded | |
|---|---|
| Description | Exclude one or more SSL ciphers from the default JVM ciphers. |
| Default value | No default value |
| Required parameter | Optional |

The JVM supports a number of ciphers as listed in:

https://docs.oracle.com/javase/8/docs/technotes/guides/security/StandardNames.html

Some of these ciphers are enabled by default. Use this parameter to disable one or more ciphers enabled by default. For example, to disable the ciphers `TLS_DHE_RSA_WITH_AES_128_CBC_SHA` and `TLS_DHE_DSS_WITH_AES_128_CBC_SHA`, configure this parameter as follows:

```
CipherListExcluded = TLS_DHE_RSA_WITH_AES_128_CBC_SHA, TLS_DHE_DSS_WITH_AES_128_CBC_SHA
```

## 4.12 SocketBufferLimit

| SocketBufferLimit | |
|---|---|
| Description | The maximum number of bytes of a client request |
| Default value | 65536 (64 KB) |
| Required parameter | Optional |

The value of this parameter determines the maximum size (in bytes) of a client requests that MigratoryData Server will accept.

## 4.13 BufferLimit.Send

| BufferLimit.Send | |
|---|---|
| Description | The default initial memory size (in bytes) for outgoing messages |
| Default value | 8192 |
| Required parameter | Optional |

The default memory size used by MigratoryData Server to create an outgoing message to be sent to the clients is given by the value of this parameter. If the actual size of the message is larger than this default value, then MigratoryData Server will reallocate the necessary memory size.

The default value of `BufferLimit.Send` is 8192 bytes. You could adjust the value of this parameter based on the average size of your messages to either avoid memory reallocation or optimize memory consumption.

## 4.14 BufferLimit.Receive

| BufferLimit.Receive | |
|---|---|
| Description | The default initial memory size (in bytes) for incoming messages |
| Default value | 8192 |
| Required parameter | Optional |

The default memory size used by MigratoryData Server to create memory space for an incoming message is given by the value of this parameter. If the actual size of the message is larger than this default value, then MigratoryData Server will reallocate the necessary memory size.

The default value of `BufferLimit.Receive` is 8192 bytes. You could adjust the value of this parameter based on the average size of your messages to either avoid memory reallocation or optimize memory consumption.

## 4.15 Proxy.Type

| Proxy.Type | |
|---|---|
| Description | The type of proxy used to connect to the MigratoryData license servers (supported types are SOCKS or HTTP) |
| Default value | No default value |
| Required parameter | Optional |

If MigratoryData Server is deployed behind a proxy, configure this parameter to be able to access the MigratoryData licensing servers to validate your evaluation license key.

> **Note** — This parameter only apply for evaluation / trial license keys, remote license verification is not used for purchased license keys.

## 4.16 Proxy.Host

| Proxy.Host | |
|---|---|
| Description | The address of your proxy used to connect to the MigratoryData license servers |
| Default value | No default value |
| Required parameter | Optional |

If MigratoryData Server is deployed behind a proxy, please configure the address of your proxy in order to be able to access the MigratoryData licensing servers to validate your evaluation license key.

The format of the proxy address is "ip_address:port" (e.g. 192.168.0.1:3128) or "dns_name:port" (e.g. push.example.com:3128).

**Note** — This parameter only apply for evaluation / trial license keys, remote license verification is not used for purchased license keys.

## 4.17 Proxy.Username

| Proxy.Username | |
|---|---|
| Description | The username to authenticate into your proxy to connect to the MigratoryData license servers |
| Default value | No default value |
| Required parameter | Optional |

If MigratoryData Server is deployed behind a proxy, please configure the username to authenticate into the proxy in order to be able to access the MigratoryData licensing servers to validate your evaluation license key.

**Note** — This parameter only apply for evaluation / trial license keys, remote license verification is not used for purchased license keys.

## 4.18 Proxy.Password

| Proxy.Password | |
|---|---|
| Description | The password to authenticate into your proxy to connect to the MigratoryData license servers |
| Default value | No default value |
| Required parameter | Optional |

If MigratoryData Server is deployed behind a proxy, please configure the password to authenticate into the proxy in order to be able to access the MigratoryData licensing servers to validate your evaluation license key.

**Note** — This parameter only apply for evaluation / trial license keys, remote license verification is not used for purchased license keys.