Kofax RPA

Administrator's Guide

Version: 11.5.0

Date: 2023-10-02

Table of Contents

Preface

This guide is intended for system administrators who deploy Kofax RPA in the enterprise environment.

If you are running one of the previous versions of Kofax RPA, see the *Kofax RPA Upgrade Guide* for upgrade procedures.

The guide includes administration information for Kofax RPA including:

- Runtime
- Tomcat Management Console
- Audit Log for Management Console
- SQL Scripts for Kofax RPA Tables

Related Documentation

The documentation set for Kofax RPA is available here:[1]

https://docshield.kofax.com/Portal/Products/RPA/11.5.0-nlfihq5gwr/RPA.htm

The documentation set includes the following resources listed in alphabetical order:

Kofax RPA Administrator's Guide

Describes administrative and management tasks in Kofax RPA.

Kofax RPA Best Practices Guide

Offers recommended methods and techniques to help you optimize performance and ensure success while using Robot Lifecycle Management in your Kofax RPA environment.

Kofax RPA Desktop Automation Service Guide

Describes how to configure and manage the Desktop Automation Service required to use Desktop Automation on a remote computer.

Kofax RPA Developer's Guide

Contains programmer user guides for the Java and the .NET APIs used to execute robots on RoboServer. Also, includes information on the Management Console REST services provided with the product.

[1] You must be connected to the Internet to access the full documentation set online. For access without an Internet connection, see the *Installation Guide*.

Kofax RPA Getting Started with Robot Building Guide

Provides a tutorial that walks you through the process of using Kofax RPA to build a robot.

Kofax RPA Getting Started with Document Transformation Guide

Provides a tutorial that explains how to use Document Transformation functionality in a Kofax RPA environment, including OCR, extraction, field formatting, and validation.

Kofax RPA Help

Describes how to use Kofax RPA. The Help is also available in PDF format and known as *Kofax RPA User's Guide*.

Kofax RPA Installation Guide

Contains instructions on installing Kofax RPA and its components in a development environment.

Kofax RPA Java API documentation

Provides access to the Kofax RPA Java API packages and classes for developers to use with Kofax RPA.

> ℹ The Kofax RPA APIs include extensive references to RoboSuite, the original product name. The RoboSuite name is preserved in the APIs to ensure backward compatibility. In the context of the API documentation, the term RoboSuite has the same meaning as Kofax RPA.

Kofax RPA Release Notes

Contains late-breaking details and other information that is not available in your other Kofax RPA documentation.

Kofax RPA Technical Specifications

Contains information on supported operating systems and other system requirements.

Kofax RPA Upgrade Guide

Contains instructions on upgrading Kofax RPA and its components to a newer version.

Kofax RPA User's Guide

Contains instructions for using Kofax RPA and its components. Includes the *Kofax RPA Help* topics, plus more in depth coverage not available in the *Help*.

System Requirements

For information on supported operating systems and other system requirements, see the *Kofax RPA Technical Specifications* document on the Kofax RPA Product Documentation site: https://docshield.kofax.com/Portal/Products/RPA/11.5.0-nlfihq5gwr/RPA.htm.

Training

Kofax offers both classroom and computer-based training to help you make the most of your Kofax RPA solution. Visit the Kofax Education Portal at https://learn.kofax.com/ for details about the available training options and schedules.

Also, you can visit the Kofax Intelligent Automation SmartHub at https://smarthub.kofax.com/ to explore additional solutions, robots, connectors, and more.

Getting help with Kofax products

The Kofax Knowledge Portal repository contains articles that are updated on a regular basis to keep you informed about Kofax products. We encourage you to use the Knowledge Portal to obtain answers to your product questions.

To access the Kofax Knowledge Portal, go to https://knowledge.kofax.com.

> ⓘ The Kofax Knowledge Portal is optimized for use with Google Chrome, Mozilla Firefox, or Microsoft Edge.

The Kofax Knowledge Portal provides:

- Powerful search capabilities to help you quickly locate the information you need.

  Type your search terms or phrase into the **Search** box, and then click the search icon.
- Product information, configuration details and documentation, including release news.

  To locate articles, go to the Knowledge Portal home page and select the applicable Solution Family for your product, or click the View All Products button.

From the Knowledge Portal home page, you can:

- Access the Kofax Community (for all customers).

  On the Resources menu, click the **Community** link.
- Access the Kofax Customer Portal (for eligible customers).

  Go to the Support Portal Information page and click **Log in to the Customer Portal**.
- Access the Kofax Partner Portal (for eligible partners).

  Go to the Support Portal Information page and click **Log in to the Partner Portal**.
- Access Kofax support commitments, lifecycle policies, electronic fulfillment details, and self-service tools.

  Go to the Support Details page and select the appropriate article.

Chapter 1

Runtime

Kofax RPA offers a number of tools for executing robots you have developed. The following sections describe these tools:

- RoboServer is a server application that enables remote clients to execute robots. It is configured using both the Management Console and the RoboServer Settings application (for advanced configuration, such as security and authentication).
- Management Console helps you to schedule execution of robots, view logs and extracted data. It also provides a centralized place where settings for clusters of RoboServers can be configured.

⚠ After changing the configuration settings for any Kofax RPA component, restart the respective component for the changes to take effect.

Settings added using properties have precedence over the settings configured in the user interface.

ⓘ Timezone definitions are embedded in the bundled JRE. In case there are changes to the definitions since the release date, the JRE can be updated using the *Timezone Updater Tool* provided by Oracle. Refer to the Oracle website for further information.

RoboServer

RoboServer runs robots created in Design Studio. Robots can be started in various ways; either scheduled to run at specific times by a Management Console, called through a REST web service, through the Java or .NET APIs or from a Kapplet.

❗ The minimal Linux installation must include the following packages to be able to run Robots created with Default browser engine.

- `libX11.so.6`
- `libGL.so.1`
- `libXext.so.6`

  Use `yum install` or `sudo apt-get` command to install necessary libraries on a Linux platform.

Also, make sure that the system has all the fonts installed for the Webkit robots to run. This might be necessary in case a headless Linux install is used, because some of the Linux installation packages do not contain fonts.

To install the fonts, see the instructions below:

- Instructions for installing fonts for CentOS / RedHat
- Instructions for installing fonts for Ubuntu

To be able to execute robot, RoboServer must be activated by a Management Console. A RoboServer is active when it belongs to a cluster in a Management Console with a valid license, and sufficient KCUs have been assigned to the cluster. A RoboServer also receives settings from the Management Console where they are configured on the clusters.

The three connection types between the RoboServer and the Management Console are available:

- **Client Connection**: RoboServer creates an HTTP(S) connection to the Management Console and registers to a specified cluster. Use this connection type only with the "Client" type cluster. Usually this connection type is used when Kofax RPA is deployed in the cloud environment. It is not supported in High Availability mode.
- **Socket Service**: RoboServer acts as a service. Management Console initiates a socket connection after RoboServer registers to a specified cluster. Use this connection type only with the "Service" type cluster.
- **Socket SSL Service**: RoboServer acts as a service. Management Console initiates an encrypted socket connection after RoboServer registers to a specified cluster. Use this connection type only with a "Service SSL" type cluster.

Configure this option:

- In the RoboServer Settings application. See RoboServer Configuration.
- In the command line. See Start RoboServer.

See the Management Console chapter in *Kofax RPA Help* for more information on the administration of RoboServers and clusters.

In Kofax RPA version 11.2.0 and later, RoboServer writes logs in UTC time. By default, RoboServers version prior to 11.2.0 write logs in local server time, which can lead to inconsistences in timestamps if both 11.2.0 (or later) and earlier versions of RoboServer log to the same logging database. If you connect RoboServers version prior to 11.2.0 to a Management Console version 11.2.0 or later, you can configure them to write log messages in UTC time instead of local server time by uncommenting the following option in the `RoboServer.conf` file:

```
wrapper_java_additional.41=-DwriteLogdbUtc=true
```

Note that RoboServer must be updated to a fix pack version that supports this parameter. See the corresponding fix pack ReadMe file for details.

Start RoboServer

RoboServer can be started in several different ways:

- By invoking it from the command line.
- By running it as a service. See Start Servers Automatically.
- By running the Docker container. See Docker Tools for Kofax RPA Deployment.
- By clicking the RoboServer program icon (or the Start Management Console program icon from the Start menu that starts both Management Console and RoboServer). For demonstration and testing purposes only.

To invoke a RoboServer from the command line, open a Command Prompt window, navigate to the `bin` folder in the `Kofax RPA` installation folder and type:

```
RoboServer
```

If all required parameters are specified in the configuration file (`C:Users\[User]\AppData\Local\[Version]\Kofax RPA\Configuration\roboserver.settings`), the RoboServer starts.

If any of the necessary parameters is missing, the RoboServer produces an error and displays the usage help and available parameters.

RoboServer Parameters

The command line for starting a RoboServer may include the following parameters:

```
RoboServer [-client]  [-s <service:params>] [-mcUrl <url>] [-ss <MC Shared Secret> [-cl
 <Cluster Name>] [-b <url>] [-p <port number>] [-sslPort <port number>] [-v]
```

RoboServer accepts the parameters in the following table. Note that you can edit all the parameters in the RoboServer Settings application. See RoboServer Configuration for more details. In the Docker environment, you must set the environment variables in the `docker-compose` file.

Parameter	Description
`-mcUrl <arg>`	This required parameter specifies which Management Console to register to in the following format: `http[s]://<hostname>:<port number>` Example: `–mcUrl http://myserver:8080/ManagementConsole` ⓘ When using Document Transformation step with a Callback option in a robot, use the RoboServer host name or IP address in the `–mcUrl` parameter. Do not use 'localhost', because the Document Transformation Service will not be able to reach the Management Console, and the callback robot will not be queued.

`-ss`
`--mcSharedSecret <MC Shared Secret>`
	This required parameter specifies the shared secret used to authenticate RoboServer with Management Console. The shared secret should be copied from the Management Console Service authentication section. For more information, see "Service authentication" in *Kofax RPA Help*.

`-cl`
`--cluster <arg>`
	This required parameter automatically registers a RoboServer with the specified cluster on the Management Console. In the following example the RoboServer registers itself with the *Production* cluster.
	Example: `–cl Production`
	Example: `–mcUrl http://myserver:8080/ManagementConsole -ss <MC Shared Secret> -cl Production`

`-eh`
`--externalHost <port number>`
	Explicitly specifies the name or IP address of the RoboServer host.
	This parameter should be specified when the host address is different from what a RoboServer discovers on the local machine, such as when running with NAT in the cloud, or when you run the RoboServer in a Docker container.
	Example: `–eh 10.10.0.123`

`-ep`
`--externalPort <port number>`
	Explicitly specifies the port number of the RoboServer host.
	This parameter should be specified when the host port is different from what a RoboServer discovers on the local computer, such as when running with NAT in the cloud, or when you run the RoboServer in a Docker container.

`-jmxPass`	Sets the JMX password if you monitor a RoboServer with JMX application and require a password.
`-v` `--verbose`	This optional parameter causes a RoboServer to output status and runtime events.
`-V` `--version`	This optional parameter causes a RoboServer to output the version number, and then exit.
`-h` `--help`	Displays help.

Parameter	Description
`-pauseAfterStartupError`	Pauses if an error occurred at startup.

`-s`
`--service <service-name:service-parameter>`
	This parameter specifies a RQL or JMX service that RoboServer should start. This parameter must be specified at least once, and may be specified multiple times to start multiple services in the same RoboServer. The available services depend on your installation.
	Example: `--service socket:50000`
	Example: `--service jmx:50100`
	See "Available services" in the table below for more information.

`-p`
`--port <port number>`
	This is shorthand for calling `-s socket:<port number>`
	Example: `--port 50000`

`-sslPort <port number>`	This is a shorthand for writing `-s ssl:<port number>`
`-nd` `--NoDoc`	This optional parameter disallows robot documentation requests to this RoboServer.
`-sn` `--serverName`	This optional parameter sets the server name for logging RoboServer statistics, which is later displayed in Kofax Analytics for RPA. If you do not specify the server name, the statistics is collected based on the server IP address.

`-ll`
`--licenseLimit <arg>`
	This parameter specifies the maximum number of license units that a RoboServer may receive.

`-client`	Specifies that RoboServer should run as a client to a Management Console cluster. Not supported in High Availability mode.
Available services	
`--service socket:<portNumber>`	Specifies that RoboServer should run as a service to a Management Console cluster. `<portNumber>`: The port number for the socket-service to monitor.
`--service ssl:<portNumber>`	Specifies that RoboServer should run as a service to a Management Console cluster and creates a secure connection. `<portNumber>`: The port number for the socket-service to monitor.
`--service jmx:<jmx_port_Number>,<jmx_rmi_url>`	`<jmx_port_Number>`: The port number for the JMX service to monitor. `<jmx_rmi_url>`: Optional RMI host and port for the JMX service. Use if you need to connect through a firewall. Example: `--service jmx:example.com:51001`

To set the connection type between the RoboServer and the Management Console, select one of the following parameters:

- `-client`
- `--service socket:<portNumber>`

- --service ssl:<portNumber>

Examples of command lines that connect RoboServer:

- As a client to a Management Console cluster:

  ```
  -client -mcUrl http://localhost:8080/ManagementConsole -cluster <clientCluster> -ss
   <MC Shared Secret>
  ```

  where clientCluster is the name of the cluster of client connection type created in Management Console.
- As a service to a Management Console cluster:

  ```
  -service socket:50000 -mcUrl http://localhost:8080/ManagementConsole -cluster
   <serviceCluster> -ss <MC Shared Secret>
  ```

  where <serviceCluster> is the name of the cluster of service connection type created in Management Console.

To set the shared secret, use the RoboServer Settings application. For more information, see RoboServer Configuration.

⚠ Starting from Kofax RPA version 10, all RoboServers must auto register to the Management Console. Therefore, the URL and shared secret for the Management Console along with the cluster name must be specified when starting a RoboServer (either at the command line as in the following example or using the RoboServer Settings application on the General tab under Register to a Management Console option).

```
RoboServer.exe -mcUrl http://myserver:8080/ManagementConsole -ss <MC Shared
Secret> -cluster Production -service socket:50000.
```

Start Servers Automatically

If your installation includes combined RoboServer and Management Console server functionality, you can configure them so that the servers start automatically.

These two functions can be provided by the same RoboServer server program, depending on the arguments supplied to it when it starts.

The RoboServer Parameters section contains a detailed description of the command-line arguments for the RoboServer program. To enable the RoboServer program to execute robots, specify the –service argument. Similarly, the –MC argument enables the Management Console functionality.

For information about starting RoboServer and other RPA components as services, see Run RPA Components As Services.

Shut Down RoboServer

RoboServer can be shut down using the following command line tool. Run ShutDownRoboServer without arguments to see the various options for how to shut down the server, particularly how to handle any robots currently running on the server.

Production Configuration

RoboServer runs robots created with Design Studio. Robots can be started in various ways; either scheduled to run at specific times by the Management Console, called via a REST web service, through the Java or .NET APIs or from a Kapplet.

In order to get a stable and performing production environment, you may have to tweak some of the default RoboServer parameters. We will look at the following configuration options:

- Number of RoboServer instances
- Number of concurrent robots
- Memory allocation

Number of RoboServer instances

RoboServer runs on Oracle's Java Virtual Machine (JVM), which in turn runs on top of an operating system (OS), which runs on top of your hardware. JVM's and OS's are patched, hardware architecture changes, and each new iteration aims to bring better performance. Although we can give some general guidelines about performance, the only way to make sure you have the optimal configuration is to test it.

As a general rule you get a little more performance by starting two instances of RoboServer. The JVM uses memory management known as garbage collection (GC). On most hardware, only a single CPU core is active during GC, which leaves 75% of the CPU idle on a quad-core CPU. If you start two instances of RoboServer, one instance can still use the full CPU while the other in running GC. However, note that the garbage collector CPU usage depends on the JDK specification that your environment operates on, so multiple CPU cores can be using during the GC process.

Number of concurrent robots

The amount of concurrent robots a RoboServer can run depends on the amount of CPU available, and how fast you can get the data RoboServer needs to process. The number of concurrent robots is configured in the Management Console cluster settings. A robot running against a slow website will use a lot less CPU than a robot running against a website with a fast response time, and here is why. The amount of CPU used by a program can be described with the following formula

```
CPU (core)% = 1 - WaitTime/TotalTime
```

If a robot takes 20 seconds to execute, but 15 seconds are spent waiting for the website, it is only executing for 5 seconds, thus during the 20 seconds it is using an average of 25% (of a CPU core). The steps in a robot are executed in sequence, which means that a single executing robot utilizes only one CPU core at a time. Most modern CPUs have multiple cores, so a robot that executes in 20 seconds, but waits for 15 seconds, in fact only uses about 6% of a quad-core CPU.

By default RoboServer is configured to run 20 robots concurrently. The number of concurrent robots is configured in the Management Console cluster settings. If all your robots use 6% CPU, the CPU is fully utilized when you are running 16-17 robots concurrently. If you start 33 of these 6% robots concurrently, you overload RoboServer; because the amount of CPU available is constant, the result is that each robot takes twice as long to finish. In the real world the CPU utilization of a robot may be anywhere between 5-95% of a CPU core, depending on robot logic and the website it interacts with. As a result it is hard to guess or calculate the correct value for the max concurrent robots. The only way to be sure you have the right value is to do a load test and monitor the RoboServer CPU utilization, as well as the robot runtime as load increases.

Memory allocation

Another parameter that may affect the number of concurrent robots each RoboServer can handle is the amount of memory. The amount of memory used by robots can vary from a few megabytes (MB) to hundreds of MB. By default, RoboServer is configured to use 2048 MB for a 64-bit system. Check Change the RAM Allocation to see how to control memory allocation. To avoid getting an out of memory error, provide enough memory to a RoboServer. To ensure proper memory allocation, monitor memory utilization during your load tests. The JVM does not allocate all of the available memory, but it reserves it from the OS. Once the JVM starts to use the memory, it is not given back to the OS. To find the optimal memory allocation, run a series of load tests that push the CPU to 100%. After each test is complete, check how much of the reserved memory was actually used by the JVM (the java.exe process). If all 2048MB (default) were used, increase (usually double) the memory and run the test again. At some point the JVM does not use all of the reserved memory, and the number of the used memory reflects the actual memory requirement and should be specified for the RoboServer.

Scaling RoboServer use

The default method is load-balanced scaling. The defining RoboServer characteristic for the scaling method is not the amount of CREs, but the amount of free slots for concurrently run robots. And if that is equal, the scaling method checks the queue size.

For on-site deployments, robots are scheduled on each RoboServer using a load-balanced method. In these environments, robots are queued on the RoboServer with the highest number of free execution slots. Work is distributed evenly among each RoboServer.

For a multi-tenant cloud deployment, it may be necessary to scale RoboServer differently. Available only for use in cloud environments, the scalable method queues robots to any RoboServer with empty execution slots, prioritizing the ones that have the least slots available. For a RoboServer that becomes idle faster, this method allows an unused RoboServer to be scaled down. This method optimizes scalability, provided that the maximum number has not been reached and the RoboServer is not in shut-down mode.

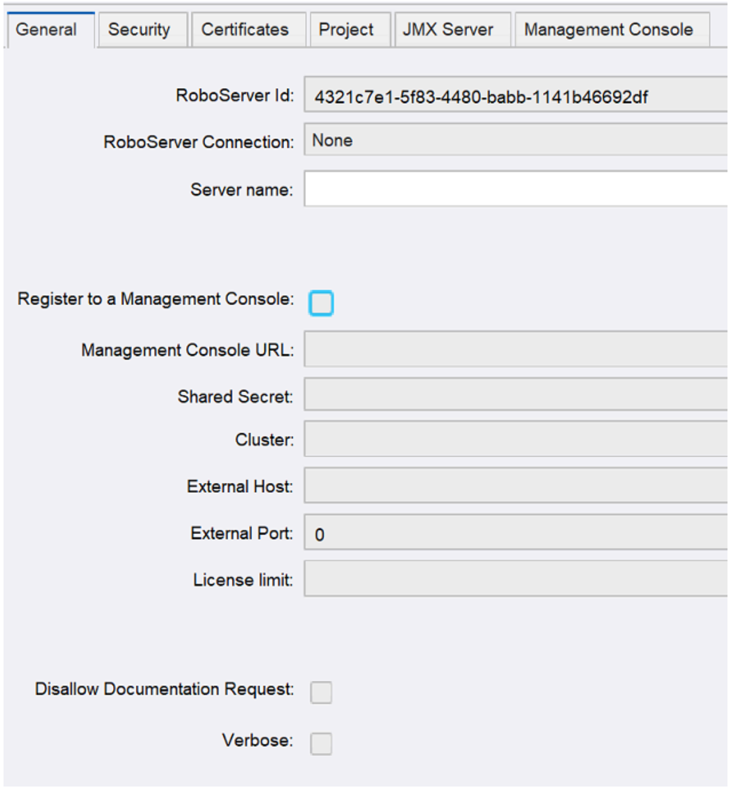
Change the scaling method at start-up. If you change the method while Management Console is running, the change requires a re-start to become effective.

Configure the scalable method in either one of the following files:

- In a Docker configurator file, specify the `SETTINGS_CLUSTERMANAGER_DISTRIBUTIONSTRATEGY` setting.
- In the `common.xml` file, specify the `<property name="distributionStrategy"="SCALABLE"/>` setting.

RoboServer Configuration

You can configure RoboServer in the RoboServer Settings application that can be started from the Windows Start menu.



RoboServer Id: 4321c7e1-5f83-4480-babb-1141b46692df

RoboServer Connection: None

Server name:

Register to a Management Console: □

Management Console URL:

Shared Secret:

Cluster:

External Host:

External Port: 0

License limit:

Disallow Documentation Request: □

Verbose: □

RoboServer Settings main window

Use this application to configure:

- **General**: RoboServer connection options, Management Console connection options including the shared secret, RoboServer host settings, number of license units, and the Verbose option.
- **Security**: Security settings such as authentication and permissions.
- **Certificates**: The use of certificates.
- **Project**: The location of the default project.
- **JMX Server**: JMX Server Configuration.
- **Management Console**: Embedded Management Console configuration.

After changing any of the settings, click **OK** to store the new settings, and then restart any RoboServers that are running for the changes to take effect.

Starting from Kofax RPA version 10, all RoboServers must auto register to the Management Console. Therefore, the URL and shared secret for the Management Console along with the cluster name must be specified when starting the RoboServer (either at the command line or using the **RoboServer Settings** application).

RoboServer Id is a unique identifier used to identify your RoboServer on the **Management Console** > **Admin** > **RoboServers** > **Server** tab. You can change this ID in the `roboserver.settings` configuration file by editing the `roboserver-id` variable value.

The name or IP address and the port number of the RoboServer host should be specified when those parameters are different from what a RoboServer discovers on the local computer, such as when running with NAT in the cloud, or when you run the RoboServer in a Docker container.

If you need to change the maximum amount of RAM that RoboServer can use, see Change the RAM Allocation.

Start and Enter License in Embedded Management Console

Before you can enter license information into Management Console, you need to start it. If you use an embedded Management Console, start it as follows. See Tomcat Deployment for information about Tomcat Management Console.

Windows

Use the **Start Management Console** item on the Start menu.

To start the Management Console from the command line, run the following command in the `bin` subfolder of the installation folder.

```
RoboServer.exe -p 50000 -MC -mcUrl http://localhost:50080
```

You can also use the command line to start a RoboServer and register it to a Management Console:

`RoboServer.exe -p 50000 -MC -mcUrl http://localhost:50080 -cl "Production"` command starts a RoboServer on port `50000` and registers it to the Management Console at `ServerName:port` under the `Production` cluster.

Linux

Start Management Console from the command line. It is part of the RoboServer program, which is found in the `bin` directory under the installation directory.

```
$./RoboServer -p 50000 -MC -mcUrl http://localhost:50080
```

Auto-start

As an alternative, if you later set up auto-start of the Management Console as described in Start RoboServer, you may select to do that now instead of starting Management Console manually.

After the Management Console is started, open it in a browser. On Windows, click the Management Console item on the Start menu. On all platforms, you can open a browser and go to `http://localhost:50080/`. Log in to the Management Console using the default `admin` user credentials, accept the license terms and enter your license information, including your license keys. If you need to change the license information later, you can do so in **Admin** > **License**.

Embedded Management Console Configuration

The settings are available on the **Management Console** tab of the RoboServer Settings application.

RoboServer contains an embedded web server which runs the Management Console. The web server is part of RoboServer, but is activated only when a RoboServer is started with the `–MC` option. By default, the web server interacts with port 50080, and thus the Management Console web interface is available on:

```
http://host:50080/
```

Protocols and Ports

You can configure the web server to be accessible through HTTP and HTTPS on separate ports. If a protocol is enabled, a port number must be chosen; the defaults are port 50080 (HTTP) and port 50443 (HTTPS).

To enable HTTPS, a server certificate in JKS format must be stored in a file called `webserver.keystore` in the `Certificates/Web` folder, which resides in `C:Users\[User]\AppData\Local\Kofax RPA\[version]`. If a certificate password other than the default (*changeit*) must be used, enter it in the Certificate Password field.

You can also restrict who is allowed to upload JDBC driver to the embedded Management Console (for more information, see "Database drivers" in *Kofax RPA Help)*. Possible choices are "**Not Allowed**", where no one can upload JDBC drivers, "Admin from localhost," which means that the admin user can upload drivers when accessing the Management Console from the local computer; and finally, "Admin from any host," which means the admin user can always upload JDBC drivers.

User Management

Management Console can be accessed not only from the same computer (localhost), but also from others. One of the points of having a Management Console is that it coordinates execution of robots, and thus it typically must be accessible to many clients.

To mitigate the potential security risk of having access to the Management Console from other computers, user management is enabled by default in embedded mode and the default `admin` superuser password is available (user name - `admin`, password - `admin`). You must use these credentials when you access the web interface from a browser. See Predefined User Roles for more information.

Security

On the RoboServer settings **Security** tab, specify RoboServer TLS configuration, general security restrictions, whether authentication is required for accessing the RoboServer, and audit logging preferences.

Allow File System and Command Line Access

Enables RoboServer to create and edit files on the computer where RoboServer runs.

> ❗ When using embedded Derby database, robots can create and edit files on computers when this option is not selected. We recommend using MySQL or another enterprise-class database in your network environment.

Allow the use of Connectors

When running on RoboServer, this setting enables the use of custom Connectors in robots on the computer where RoboServer runs. Use custom Connectors in the Custom Action step in Robots. See *Kofax RPA Help* for details.

Accept JDBC Drivers from Management Console

Distributes JDBC drivers from the Management Console to the RoboServer.

Command Time-out

Specifies how long the RoboServer must wait for a reply from a command on a remote device. This option applies only to automating terminals and browsing web sites in Robots.

A command is an instruction sent to Automation Device, such as click a mouse button, open an application, add a location found guard, and so forth. If a command cannot be completed in a specified time, the service sends a notification and execution of the robot stops.

Note that in case of a Guarded Choice step, this setting applies to invoking the guard in the workflow, but waiting for the guard to be satisfied is not bound to this timeout and can wait forever. Similar situation occurs when using the Move Mouse and Extract steps. The commands must be invoked on the device withing the timeout specified in this field, but the robot waits for up to 240 seconds for the commands to complete.

TLS

To make RPA components work correctly, you need to set the TLS certificates for:

1. RoboServer that applies certificates in four different ways corresponding to the four properties on the **Certificates** tab of the RoboServer Settings application. They refer to the communication between:
   - Management Console and RoboServer
   - RoboServer and automation device
   - RoboServer and API
   - RoboServer and a website

   The "RoboServer - API and "RoboServer - website" communication properties have to do with how robots access web servers as part of their execution.
2. Management Console that needs a certificate for the communication with API that requires Tomcat settings, settings in the Management Console, and in some other related folders.
3. Kapplets Service
4. Robot File System
5. Desktop Automation Service

Prepare Certificates

To prepare for the HTTPS connection, you should create a self-signed certificate. You can generate a self-signed certificate using the Java keytool utility provided along with the JDK package as follows, using a command-line tool. In the command-line, use the path `[JAVA_HOME]\bin` and locate `keytool`. Then follow the example below.

```
keytool -genkeypair -alias mc -keyalg rsa -validity 3650 -keystore mc.p12 -storetype
 pkcs12 -ext san=dns:<host machine name>,ip:127.0.0.1,ip:::1,ip:<IP>
```

> The certificate should include all host names/IP addresses of the Tomcat Management Console computer that you expect to use while connecting to the RPA components. The Common Name (`cn`) parameter must match the host name of the Tomcat Management Console computer.

You need to create two keystores with different names: one keystore for deploying Management Console on Tomcat and another keystore for communicating with other parts of RPA with Management Console (`communication.p12`). The same options can be used for both certificates.

Use the following commands to extract certificates from the keystore:

```
keytool -exportcert -alias mc -keystore mc.p12 -file mc.cer
```

```
keytool -exportcert -alias communication -keystore communication.p12 -file
 communication.cer
```

Management Console TLS Configuration

To ensure stable work with SSL within RPA parts, complete the following steps:

1. Edit the server.xml file in **Tomcat** > **Conf**, as follows:

   ```
   <Connector port="8443" protocol="org.apache.coyote.http11.Http11NioProtocol"
    maxThreads="150" SSLEnabled="true">
       <SSLHostConfig>
           <Certificate certificateKeystoreFile="<path to the certificate/
   mc.p12>" certificateKeystorePassword="<your password>" type="RSA"
    certificateKeyAlias="mc"/>
       </SSLHostConfig>
   </Connector>
   ```

2. If you always need to redirect connection to TLS (that is not to allow the HTTP connection), add the following commands to the web.xml file in **Tomcat** > **Conf**:

   ```
   <security-constraint>
       <web-resource-collection>
           <web-resource-name>Protected Context</web-resource-name>
           <url-pattern>/*</url-pattern>
       </web-resource-collection>
       <user-data-constraint>
           <transport-guarantee>CONFIDENTIAL</transport-guarantee>
       </user-data-constraint>
   </security-constraint>
   ```

   For more information on the SSL Tomcat settings, see this article available on the Apache Tomcat website.

3. Start Management Console: `https://localhost:8443/ManagementConsole/`.

4. Start Management Console and create a new cluster with **Use SSL** selected. Then proceed to TLS configuration to set up the other half of the communication.

5. Optionally, for the encrypted RQL communication with RoboServer, locate the certs.xml file in the `[Tomcat_home]/webapps/[Application name(ManagementConsole)]/WEB_INF` folder, and specify the following:

   ```
   <property name="privateCertificateLocation" value="/WEB-INF/communication.p12"/>
   <property name="privateCertPassword" value="changeit"/>
   ```

RoboServer TLS Configuration

Kofax RPA provides a means for setting up TLS communication between the automation device, RoboServer, Kapplets, Robot File System, or Design Studio. The communication uses certificates for encrypting the communication. The encryption uses a public-private key structure for securing the connection.

In the RoboServer Settings application, on the Security tab, under **TLS Configuration Settings**, you can specify whether to use the built-in set of certificates or specify some other.

- To use the Kofax RPA certificates, select **Use Defaults**.
- To use other certificates, clear **Use Defaults** and specify the paths to private and public keys as well as to the trusted certificates folder in the corresponding fields.

See "Use TLS Communication" in *Kofax RPA Help* for more information.

Establishing SSL connection between RoboServer and the clusters in Management Console

1. Start Management Console and create a new cluster with **Use SSL** selected.
2. In the **RoboServer Settings** application, navigate to **Certificates** > **API** and select **Verify API Client Certificates** to accept connections only from Management Console.
3. Add the communication.p12 certificate to the API Server Certificate.
4. Put the communication.cer certificate from the specified Management Console certs.xml keystore (communication.p12) into:
   - on Linux: `[USER_HOME]/.Kofax RPA/[version]/Certificates/API/TrustedClients`, and
   - on Windows: `[User]\AppData\Local\Kofax RPA\[version]\Certificates\API\TrustedClients`

   The `cn` attribute of the certificate must match the resolved host name of Management Console
5. Add communication.cer to the JRE keystore used by RoboServer:

   ```
   keytool -import -alias communication -keystore "C:\Program Files\[JAVA_HOME]\lib
   \security\cacerts"       -trustcacerts -file C:\<path>\communication.cer
   ```
6. Start RoboServer using the Command Prompt window:

   ```
   RoboServer -service ssl:50001 -mcUrl https:\\<hostname or IP> -ss <MC Shared
    Secret> -cluster SSL
   ```

To enable certificate validation or to let Management Console use a certificate so that RoboServer can verify its validity, change the settings and provide the path to the keystore (communication.p12) in the certs.xml file under `ManagementConsole\WEB-INF`.

Request Authentication

This option is deprecated.

To protect your RoboServer against unauthorized access, you can turn on authentication. This has effect on all RoboServers run from your Kofax RPA installation, including a RoboServer started as a service or from a command line.

To turn on authentication, select the **Require RoboServer Authentication** check box in the RoboServer Settings application. To run robots on a RoboServer with authentication turned on, you have to add users by clicking the add button. You can then fill out the information about the user including the user name that will be shown on the list of users.

A user is configured using the properties in the following table.

User Properties

Property	Description
User name	The user name used by the user when accessing the RoboServer.
Password (Password Hash)	The password used by the user when accessing the RoboServer.
Comments	Here you can write a comment about the user.
Start Robot	Enables the user to start robots on the RoboServer.
Stop Robot	Enables the user to stop robots on the RoboServer.
Shutdown RoboServer	Enables the user to shut down the RoboServer from the Management Console.

Kapplets Service TLS Configuration

To establish the TLS configuration for the connection from Kapplets to Management Console, use the same settings as for deploying Kapplets, with the following differences.

1. Use the SSL path to Management Console in kapplets.xml:

```
<Environment name="kapplets.services.mc.connection.url" value="https://<
 hostname>8443/ManagementConsole/" type="java.lang.String" override="false"/>
```

2. Add the certificate from Management Console into the JRE keystore used by the Kapplet Service Tomcat at `[JAVA_HOME]\lib\security\cacerts`. To do so, use the following command:

```
keytool -import -alias mc -keystore "C:\Program Files\[JAVA_HOME]\lib\security
\cacerts" -trustcacerts -file C:\<path>\mc.cer
```

Robot File System Configuration

Use the following procedure to configure the TLS connection for the Robot File System (RFS).

1. Use the SSL Tomcat configuration as described for Management Console.
2. In the `web.xml` file, configure the following parameters.
   - Set the TLS address for Management Console: `https://<hostmachine>:8443/ManagementConsole`.
   - Configure the RFS shared secret to authenticate with the Management Console. Store the shared secret in a file and set the path to this file in the `shared-secret-file` parameter, or paste the plain text shared secret in the `shared-secret` parameter.

3. Add `mc.cer` to the `cacerts` keystore of Java environment used by Tomcat. For example, `[JAVA_HOME]\lib\security\cacerts`.
4. In the RFS server configuration on the **General** tab of Management Console, set the TLS address for the RFS server: `https://<hostmachine>:8443/rfs`.
5. Add `mc.cer` to the used JRE keystore used by Management Console and RoboServer at `C:\Program Files\Kofax RPA 11.5.0.0\jre\lib\security\cacerts`. To do so, use the following command:

```
keytool -import -alias mc -keystore "Kofax RPA 11.5.0.0\jre\lib\security\cacerts"
 -trustcacerts -file C:\<path>\mc.cer
```

ⓘ If Management Console and RoboServer use different Tomcat servers, generate a keypair for the Robot File System. Otherwise, ensure that the Management Console certificate has been imported to the Java `cacerts`.

With the Robot File System, if you use a self-signed certificate or a certificate signed with a private root CA, configure a CA certificate on every system that runs RoboServer, Design Studio, or the Desktop Automation Service. As a result, the Robot File System will become available. To do so, complete the following steps on each system:

1. Identify the accounts used to run RoboServer, Design Studio and/or the Desktop Automation Service on the system.
2. Copy the `certificate.cer` file to the system.
3. Configure the system to add the environment variable `NODE_EXTRA_CA_CERTS` to the accounts identified in step 1.
   - If the `NODE_EXTRA_CA_CERTS` environment variable is already defined for an account, locate the file it refers to and add the contents of the `certificate.cer` file to this file.
   - If the `NODE_EXTRA_CA_CERTS` environment variable is not defined, add a definition and set its value to the full path of the `certificate.cer` file.
4. Ensure that the accounts identified in step 1 have read access to the file referenced by `NODE_EXTRA_CA_CERTS`.

Desktop Automation Service Configuration

To make the Desktop Automation Service available for all Kofax RPA components, complete the following steps:

1. Use the SSL path Management Console in **Desktop Automation Service** > **Configure Management Console**: `https://<hostname>:8443/ManagementConsole/`.
2. Download the Management Console certificate, `communication.cer`, and add it to the Desktop Automation Service, in the **CA file** field.

There is another way to save a root certificate as a file from the Google Chrome browser to Management Console. To do so, complete the following steps:

1. Right-click the lock icon in the address bar and click **Certificate**.
2. On the **Certificate Path** tab, select the topmost root certificate and click **View Certificate**.

3. On the **Details** tab, click **Copy to File** and follow the wizard to export the root certificate as a base-64 encode X.509 certificate.

JMX Server Configuration

You can use the embedded JMX server to monitor a running RoboServer through tools such as JConsole. Enable JConsole by providing an argument on the RoboServer command line.

Hiding Sensitive Robot Input

The **Show Inputs** option controls whether robot input parameters are shown in the management interface. This makes it possible to hide security sensitive information such as passwords.

JMX Server Access

By default, a JMX server can be accessed by all clients with access to the correct port on the server. By selecting the **Use Password** option, the selected user name and password are required when connecting.

Heartbeat Notifications

If an interval (in seconds) greater than 0 is specified, the JMX server sends out a heartbeat notification with the given interval, as long as a RoboServer is running and responding to queries.

Default RoboServer Project

You can set the location of the default RoboServer project folder on the RoboServer Settings Project tab. By default, the folder is set to the default robot project created during the installation process. See the Design Studio chapter in the *Kofax RPA Help* for more information on robot projects.

The RoboServer default project is used only by the API. When executing a robot using API, any references it has to types, snippets or other resources are resolved by looking in the default project.

Change the RAM Allocation

As installed, each Kofax RPA application is configured with a maximum amount of RAM that it may use. This amount usually is plenty for ordinary work, but if you run many robots in parallel on a RoboServer, or if some robots use much RAM, it may be necessary to increase the allocation.

You can change the allocation for any of the applications by editing its `.conf` file, found in the `bin` subfolder of the installation folder.

For RoboServer, edit: `bin/RoboServer.conf`.

For Design Studio, edit: `bin/DesignStudio.conf`.

To edit the file, perform the following steps.

1. Open the corresponding `.conf` file in a text editor.

2. Find the line containing the `wrapper.java.maxmemory` parameter.
3. Un-comment the line (remove the leading #) and edit its value.
   For example, to permit a RoboServer to use up to 4GB of RAM, enter the following: `wrapper.java.maxmemory=4096`.

   > ⓘ If the `.conf` file does not contain the `wrapper.java.maxmemory` line, add the whole line to the file. The `.conf` file can be edited only by the user who installed Kofax RPA, such as the Windows administrator.

Troubleshoot RoboServer Service Startup

If your service does not start, look for RoboServer messages in the Windows event log. Make sure you have installed the service with the `wrapper.syslog.loglevel=INFO` argument. For more information, see Kofax RPA Initial Configuration in the *Kofax RPA Installation Guide*.

Chapter 2

Tomcat Management Console

For production environment we strongly recommend deploying Management Console as a regular web application on a standalone Tomcat web server.

> ❗ If your setup requires access to the Management Console outside of your corporate intranet, set up TLSv1.0 (or a later version of TLS) to work with your Tomcat server.

The following table lists the differences in the feature set.

Management Console Features and Configuration

Feature	Embedded	Standalone J2SE Web Container
Authentication	Single `admin` superuser and users defined in Management Console. Users and Roles managed by Management Console Administrator.	Users and Roles managed by Management Console Administrator. Role based security through Active Directory or other LDAP provider. Single Sign-On using CA Single Sign-On.
Management Console data store	Embedded Derby database	Container managed Data Source (supported platforms)

> ℹ The derby JDBC driver is not distributed with the Enterprise Management Console. See Apache Derby Web site for Derby JDBC driver download information. We recommend using MySQL or other enterprise-class database with your Enterprise Management Console.

Instructions on configuring an embedded Management Console can be found in the Kofax RPA online help. To start an embedded Management Console, see "Start Management Console" under the "Management Console" section.

Tomcat Deployment

This chapter provides details on how to manually install Management Console on a stand alone J2SE web container. For this guide we have chosen Tomcat. For information on supported Java version for your J2SE web container, see the *Kofax RPA Technical Specifications* document. Visit the Oracle Java SE Downloads site and download the latest Java release.

If your setup requires access to the Management Console outside of your corporate intranet, set up TLSv1.0 (or a later version of TLS) to work with your Tomcat server.

Install Management Console on Tomcat

Prerequisites

- Install the latest Java update from the Oracle website: http://www.oracle.com/technetwork/java/javase/downloads/index.html
- Download Tomcat from the Apache website https://tomcat.apache.org.
  - Install Tomcat and set user password.

Installation

1. Download full Kofax RPA installer and proceed with the installation. Select the **Management Console WAR** option during the installation.
2. Copy the `ManagementConsole.war` file from the `WebApps` folder in the Kofax RPA installation to the `webapps` folder on the Tomcat server and configure the .war file.
3. Create a `ManagementConsole.xml` Tomcat context file on the Tomcat server at `conf/Catalina/localhost/`. See Create a Tomcat Context File for details.
4. Edit `webapps/manager/WEB-INF/web.xml` file on Tomcat.
   - To upload applications, edit the following.

     ```
     <multipart-config>
     <!-- 150MB max -->
      <max-file-size>152428800</max-file-size>
      <max-request-size>152428800</max-request-size>
      <file-size-threshold>0</file-size-threshold>
     </multipart-config>
     ```

   - If your policies require compulsory use of HTTPS protocols on your network, enable the HTTP Strict Transport Security Header (HSTS Header) on the Tomcat server by using the `org.apache.catalina.filters.HttpHeaderSecurityFilter` class in `web.xml`. See the Apache Tomcat documentation for details.
   - Management Console implements Content Security Policy as an added layer of defense that helps to detect and mitigate certain classes of attacks. A `SecurityHeadersFilter` filter that is mapped to all URLs is declared and configured in `web.xml`. To configure the header's value, set the `contentSecurityPolicy` variable. This is the configuration example of header filtering in the `web.xml` file:

     ```
         <filter>
             <filter-name>SecurityHeadersFilter</filter-name>
             <filter-
     class>com.kapowtech.scheduler.server.servlet.SecurityHeadersFilter</filter-
     class>
             <init-param>
                 <param-name>contentSecurityPolicy</param-name>
                 <param-value>default-src 'self' data: blob: 'unsafe-inline' 'unsafe-
     eval'</param-value>
             </init-param>
         </filter>

         <filter-mapping>
     ```

```
        <filter-name>SecurityHeadersFilter</filter-name>
        <url-pattern>/*</url-pattern>
    </filter-mapping>
```

If you do not require header filtering, either disable or remove the filter code and the corresponding filter mapping. For more information about possible options and header filtering, search the web for Content Security Policy.

5. Start Tomcat.
6. Enter License Information.

Configure ManagementConsole.war

The Management Console application comes in the form of a Web Application Archive (WAR file) named `ManagementConsole.war`, which is located in the `/WebApps` folder in the Kofax RPA installation folder.

The version of `ManagementConsole.war` that ships with Kofax RPA is configured to run embedded inside RoboServer. Before deploying it as a standalone application on Tomcat, you may need to reconfigure it to fit your environment.

A WAR file is compressed using a compressed zip file. To access the configuration files, you must extract the zip file. Once the configuration files are updated, you re-zip and deploy `ManagementConsole.war` to your Tomcat server.

The table below contains a list of the configuration files relative to the root of the unzipped `ManagementConsole.war`.

Configuration Files

File	Configures	Notes
`WEB-INF/Configuration.xml`	Clustering, password encryption, REST-Plugin	If you copy the file from an earlier version, it will automatically be upgraded once you start the Management Console
`WEB-INF/login.xml`	Administrators and users, this is where you integrate with LDAP	
`WEB-INF/classes/log4j2.properties`	Application logging	
`WEB-INF/spring/authentication.xml`	User authentication	
`WEB-INF/roles.xml`	Built-in roles in Management Console.	

Spring Configuration Files

`Configuration.xml`, `login.xml`, `roles.xml`, and `authentication.xml` are all Spring configuration files (www.springsource.org) and share the same general syntax outlined here.

Spring is configured through a series of beans, and each bean has properties that configure a piece of code inside the application. The general syntax:

```
<bean id="id" class="SomeClass">
   <property name="myName" value="myValue"/>
</bean>
```

File	Configures
`id="id"`	The id of the bean is an internal handle, that the application use to refer to the bean. It is also referred to as the bean's name.
`class="SomeClass"`	The class identifies the code component which the bean configures.
`<property name="myName" value="myValue"/>`	Defines a property with the name `myName` and the value `myValue`. This configures a property on the code component defined by the class attribute.

In Kofax RPA, user management is performed using the Users & groups section under the Admin menu in Management Console. In the enterprise version of Kofax RPA, user management is enabled by default. User population from previous installations can be performed using the Kofax RPA backup functionality. For LDAP and SAML integrations, you still need to edit the login.xml file.

Troubleshooting

If you have any problems during the installation, you should check the Tomcat log in the `/logs` folder in your Tomcat installation. During the configuration process it is often easier to run Tomcat from the command line, as it prints error messages directly in the command line window.

Create a New Database

> Management Console and Kapplets require high bandwidth and low latency network between Management Console, Kapplets, and the database containing their data. They need to be located in the same region or in close vicinity. If Management Console and Kapplets are placed far removed from their database and/or each other, they both can potentially run into deadlocks and timeouts. Using parameters to extend timeouts and reduce the frequency of updates may lead to lower service quality.
>
> Stop both Management Console and Kapplets while carrying out maintenance on the database to avoid inconsistencies in the data and locking of Management Console and Kapplets.

We strongly recommend that you create a new database for the tables used by Management Console. There are two requirements to the database.

- Unicode support
- Case-sensitive comparison

Unicode support is needed because non-ASCII characters, like Danish Æ, German ß or Cyrillic Ë may be given as input to robots. This input is stored in the database, and without Unicode support these characters may be stored incorrectly.

Case-sensitive comparison is needed because it is possible to upload a robot named `a.robot` and another named `A.robot`. Without case-sensitive comparison, uploading the latter would override the first.

> ❗ Please create and maintain the Kofax RPA product databases according to the recommendations in the product documentation. If you are considering database modifications or customizations, do not proceed without consulting Kofax; otherwise, the results are unpredictable and the software may become inoperable.

Database servers handle Unicode and case-sensitive comparison very differently. The following list contains recommendations for the supported database systems.

Recommendations for Unicode Support and Case-Sensitive Comparison

Database	Recommendations
MySQL	Create the database with utf8mb4_bin collation. `CREATE DATABASE KAPOW_MC COLLATE utf8mb4_bin`
Oracle	NVARCHAR2, NCLOB types are used for Unicode. For case-sensitive comparison, ensure that NLS_COMP is set to BINARY.
Microsoft SQL Server	NVARCHAR, NTEXT types are used for Unicode. For case-sensitive comparison, create the database with a Case-Sensitive collation such as Latin1_General_100_BIN2: `CREATE DATABASE KAPOW_MC COLLATE Latin1_General_100_BIN2` See "Configure Microsoft SQL Server in Windows Integrated Security" in Create a Tomcat Context File for more information.
PostgreSQL	Create the database using CODESET UTF-8. `CREATE DATABASE KAPOW_MC ENCODING 'UTF8'`

The tables used by Management Console can be grouped into 3 categories: Platform tables, logging tables, and data view tables. The platform tables hold information exclusive to Management Console such as the uploaded robots and their scheduling information, while the logging and data view tables are shared with RoboServers.

For a list of supported databases and versions, see the *Kofax RPA Technical Specifications* document.

User Privileges

When a Management Console starts, it automatically tries to create the required platform tables and logging tables (if not already created by a RoboServer). This means that the user account used to access the database must have the CREATE TABLE and ALTER TABLE privileges as well as the CREATE TEMPORARY TABLES privilege to restore a backup. Oracle users also need the CREATE SEQUENCE privilege. If this is not possible you can ask your Database administrator to create the tables using the scripts below.

Additionally the user must be allowed to SELECT, INSERT, UPDATE, DELETE for the system to work properly.

SQL Scripts for Management Console Tables

SQL scripts are included with your copy of Kofax RPA in the `documentation\sql` directory. See SQL Scripts for Kofax RPA Tables for details.

Management Console uses a 3rd party scheduling component called Quartz. Quartz also requires a number of tables which must reside among the other platform tables. These tables are also created automatically when a Management Console starts, or may be created manually using the scripts in the SQL Scripts for Kofax RPA Tables.

Create a Tomcat Context File

In an enterprise environment, databases are often accessed through a data source. This section shows you how to configure your Tomcat with a data source that connects to a local MySQL database server.

In Tomcat, data sources are defined within the applications context. The context may be declared either embedded or external to the application. When the context is embedded, it is defined in the file `context.xml`, which must be located inside the WAR file in the META-INF folder. When declared externally, the file must be located in the Tomcat's `/conf/Catalina/localhost` folder and the name of the file must be `ManagementConsole.xml` (same name as the deployed WAR file). Although Tomcat recommends deploying with an embedded context, as it provides a single deployment unit, we use an external context definition in this guide, as it makes modifying the file easier. Once you have refined your configuration, you can embed the context file and deploy the WAR file to your production environment.

Add Platform Data Source

Create the file `ManagementConsole.xml` on Tomcat in the `conf/Catalina/localhost` folder and add the following content.

> The following excerpts are provided as an example only, and the actual configuration may contain other settings.

To create a connection to MySQL database:

```
<Context useHttpOnly="true">
         <!-- Default set of monitored resources -->
         <WatchedResource>WEB-INF/web.xml</WatchedResource>

         <Resource name="jdbc/kapow/platform" auth="Container"
             type="javax.sql.DataSource"
             maxTotal="100" maxIdle="30" maxWaitMillis="-1"
             factory="org.apache.tomcat.jdbc.pool.DataSourceFactory"
             validationQuery="/* ping */" testOnBorrow="true"
             username="MyUser" password="MyPassword"
             driverClassName="com.mysql.jdbc.Driver"
             url="jdbc:mysql://localhost:3306/KAPOW_MC?
useUnicode=yes&amp;characterEncoding=UTF-8&amp;rewriteBatchedStatements=true"/>

     </Context>
```

To create a connection to PostgreSQL:

```
<Context useHttpOnly="true">
         <!-- Default set of monitored resources -->
         <WatchedResource>WEB-INF/web.xml</WatchedResource>

         <Resource name="jdbc/kapow/platform" auth="Container"
             type="javax.sql.DataSource"
             maxActive="100" maxIdle="30" maxWait="-1"
             validationQuery="SELECT 1" testOnBorrow="false"
             username="username" password="password"
             driverClassName="org.postgresql.Driver"
             url="jdbc:postgresql://<URL>:<PORT>/<Database>"/>
</Context>
```

To create a connection to Oracle:

```
<Resource name="jdbc/kapow/platform" auth="Container" type="javax.sql.DataSource"
   maxTotal="100" maxIdle="30" maxWaitMillis="10000"
   validationQuery="SELECT 1 FROM DUAL" testOnBorrow="false"
   username="user" password="password" driverClassName="oracle.jdbc.OracleDriver"
   url="jdbc:oracle:thin:@<IP>:<Port>:DB_name"/>
```

The `url` parameter above is a JDBC URL. The username and password attributes are used by Tomcat to create a connection pool used when connecting to the database.

The data sources are defined differently for other databases. For instance, if you are using Microsoft SQL Server, the relevant three lines above should instead be:

```
username="MyUser" password="MyPassword"
            driverClassName="com.microsoft.sqlserver.jdbc.SQLServerDriver"
            validationQuery="SELECT 1" testOnBorrow="true"
            url="jdbc:sqlserver://localhost:1433;DatabaseName=MyDbName"/>
```

The URL `jdbc:mysql://localhost:3306/KAPOW_MC?useUnicode=yes&amp;characterEncoding=UTF-8` refers to a database named KAPOW_MC in your local MySQL. For MySQL it is recommended that you add `?useUnicode=yes&characterEncoding=UTF-8` to all connection strings, otherwise the JDBC driver cannot handle Chinese, Japanese or other 3-byte utf-8 characters correctly, as we cannot have "&" directly inside the context xml file, we must encode it as &amp;

`rewriteBatchedStatements=true` instructs the MySQL JDBC driver batch inserts/updates and should give improved insert performance for kapplet robots.

The `driverClassName` parameter controls which JDBC driver is used; each database vendor provides a JDBC driver for their database, which you have to download. The JDBC driver, typically a single `.jar` file, must be copied into the `/lib` folder on Tomcat.

The `validationQuery` is used by Tomcat to verify that the connection obtained from the connection pool is still valid (as the database server may have closed the connection). The validation query is lightweight and uses very few resources on the database server, this list contains validation queries for the supported databases.

Validation Queries

Database	**Query**
MySQL	/* ping */

Database	Query
Microsoft SQL Server	SELECT 1
Oracle	SELECT 1 FROM DUAL
PostgreSQL	SELECT 1

> ℹ IBM DB2 is not supported as a Management Console database, but you can use it as a user's database type for LogDB and for the databases available for robots.

Note that the MySQL JDBC driver supports a special lightweight /* ping */ 'request', check JConnector manual section 6.1 for details

For more information on context configuration and data sources, see JNDI Resources HOW-TO and JNDI data source HOW-TO.

Configure Microsoft SQL Server in Windows Integrated Security

If you use Microsoft SQL Server and Windows Integrated Security, computers running a Design Studio and a RoboServer must comply with the following conditions:

- Run on Windows
- Run under a user account that has been granted access to the database
- The JDBC driver must be installed manually as described later in this part

To configure Microsoft SQL Server in Windows Integrated Security, perform the following actions:

- Install JDBC drivers to the Tomcat folders: copy JAR file to the `lib` folder, copy DLL file to the `nativelib` folder.
- Do not upload the jar-file to Management Console, because it is not possible for Management Console to distribute the JDBC driver.
- Ensure that you add `;integratedSecurity=true` to the connection URL, both in your `ManagementConsole.xml` (or `context.xml`) and in the Database Type definition in a Management Console as well as any local definitions in Design Studio. See "Add Database Type" in *Kofax RPA Help*.

We are now ready to start the Tomcat server.

Start Tomcat

Start your Tomcat server, wait a couple of seconds for the application to be deployed.

Open the http://localhost:8080/ManagementConsole. You should see the login screen.

Enter `admin` as the username and `admin` as the password and click **Log in**.

Enter License Information

After logging in, the license window is displayed.

Enter the Kofax RPA license information and click **Save**. You should see a dialog box displaying which features are enabled by your license key.

Predefined User Roles

Management Console provides built-in roles that users can have. Roles are mapped to a user or a service. User permissions are calculated based on the roles that are mapped to security groups the user is a member of. You can modify built-in roles or add additional roles. The built-in roles are defined in the `roles.xml` file. See Project Permissions for details.

Built-in Roles

The following are built-in roles provided by Management Console.

Service roles are meant only for use in API applications and should not be used for interactive login to the Management Console in the browser.

- **Project Administrator**: Administrates one or multiple projects and has a right to assign a role to a group for these projects. This role gives rights to view RoboServer and cluster settings without changing them. Project Administrator is not a member of the RPA Administrators group (for more information, see later in this section.)
- **Developer**: Has a right to upload, download, and view all resource types in the repository. This role gives rights to create, edit, and delete schedules, run robots, view run logs and clusters.
- **Viewer**: Can view Schedules, Repository, Data View, Log view, and some Settings. This role gives restricted access under the Admin section and does not give rights to change or run robots.
- **API** (A service role): Gives rights to use the repository API to read from and write to the repository. This role does not permit to run robots using REST but allows running robots using RQL.
- **Service Authentication API** (A service role): Uses the repository API to read from and write to the repository. A user logs in using an OAuth authentication method.
- **RoboServer** (A service role): Can only read from the repository. This role is used by RoboServers when accessing a cluster, retrieving repository items, and requesting passwords from the password store.
- **Kapplet Administrator**: Grants a read/write access to projects in Management Console from Kapplets. In Kapplets, users with this role can manage Kapplets and create and manage Kapplet templates for the projects that contain the robots required for these templates.

  A user with this role cannot access Management Console if this user has no other rights.

  For more information, see "Kapplets user management" and "Users and User Groups" in *Kofax RPA Help*.
- **Kapplet User**: Grants a read-only access to projects in Management Console from Kapplets. In Kapplets, users with this role can only view and run Kapplets for robots belonging to the projects for which they have access.

  A user with this role cannot access Management Console if this user has no other rights.

  For more information, see "Kapplets user management" and "Users and User Groups" in *Kofax RPA Help*.
- **Kapplets Service User** (A service role): Can only read from the repository. This role is used only to retrieve information about available robots, types, snippets, and resources for the given project

and thus is used only for communication purposes between Kapplets and Management Console. This role is automatically applied to all Management Console projects.

- **Password Store Client** (A service role): This add-on role gives permission to access the password store in Management Console. The role is provided on top of other roles, just like the Developer role.
- **DAS Client User** (A service role): A user with this role is created for remote Desktop Automation Service (DAS) clients, and can only access the DAS API. The DAS client user has a right to announce a DAS to Management Console and retrieve DAS configurations.
- **VCS Service User** (A service role): Gives a special set of rights for the Synchronizer. This role grants a right to add, modify, and delete resources. This is the only role that can deploy on behalf of another user to use the "deployer" feature in the version control service.
- **Process Discovery Client** (A service role): This role allows Process Discovery components to interact with Management Console.
- **KTA Client:** (A service role): This role allows KTA components to interact with Management Console.

Built-in admin user

The admin is a superuser who has access to everything. The admin is not a member of the RPA Administrators group and cannot be a member of any group. The default admin user password is available to this user (user name - `admin`, password - `admin`). You can change the admin user password as described in "Users & groups" in *Kofax RPA Help*.

Also, you can change the initial admin user name and password in the `WEB-INF/login.xml` file by configuring the `initialAdminUser` and `initialAdminPassword` properties. The admin user is created if the database is empty or no admin user exists, and the system is configured either to use local authentication or (for LDAP, and so on) the `createAdmin` property is set to `true`. If the database already contains an admin user, the user is not created.

In an LDAP integration setup, the admin group is defined as part of the LDAP configuration. The admin can log in and define which LDAP groups should be mapped to the Developer, Project Administrator, RoboServer, and other roles.

In an internal user setup, the admin user is created at first start and can log in and create Administrators, Developers, and other users.

Built-in admin user special rights

In addition to being the initial user, the admin has special rights:

- In the RoboServers section in Management Console, the admin can click a RoboServer node and request a stack trace from the corresponding RoboServer.
- Only the admin can create and import backups.
- In the password store, the admin can move passwords to another project.

When you restore a Management Console backup, the default `admin` superuser is replaced with a superuser from the backup. Use credentials specified in the restored Management Console.

Built-in group

Users belonging to the RPA Administrators group have all rights for all projects, *excluding* special admin user rights. RPA Administrators users create new administrators and users for any project. To make a user an administrator, add the user to this group.

- The RPA Administrators group is visible when the internal user management is enabled, and it is empty by default.
- When restoring a backup created in version prior to 10.7, users with Administrator role become members of the RPA Administrators group.

Check for login attempts

By default, the check for number of login attempts made by a user and the wait time before the next attempt is disabled.

1. To enable this functionality, edit the code in the `authentication.xml` file.

   This file is located in: `<Tomcat installation folder>\WebApps\Management Console \WEB-INF\spring`. The following is a code sample.

```
   <bean id="loginAttemptService"

 class="com.kapowtech.scheduler.server.spring.security.LoginAttemptService" lazy-
init="true">
       <constructor-arg type="boolean" value="false"/>
       <constructor-arg type="int" value="3"/>
       <constructor-arg type="int" value="10"/>
   </bean>
```

2. Specify the first value as **true**.
3. Specify the second and third values to your preferences.

   The second and third values are for the number of login attempts (3 in the example) and the wait time in minutes before the next attempt (10 in the example), respectively.

Project Permissions

The admin user account used to log in bypasses the normal project permissions applied to regular users, because admin is the superuser. The superuser can not be a member of any group, and has an unrestricted access to all projects.

To change the admin password and create new users and groups, go to Management Console > **Admin** > **Users & groups**. The security model is role-based; after you create a user, you need to add this user to one or more groups associated with specific roles in one or more projects, as shown in the following example procedures.

Create "Developers" Group

1. On the Groups tab, click the plus sign.

   The Create new group dialog box appears.
2. Specify the name "Developers" and enter a description.
3. Click **OK**.

   The group appears in the table.

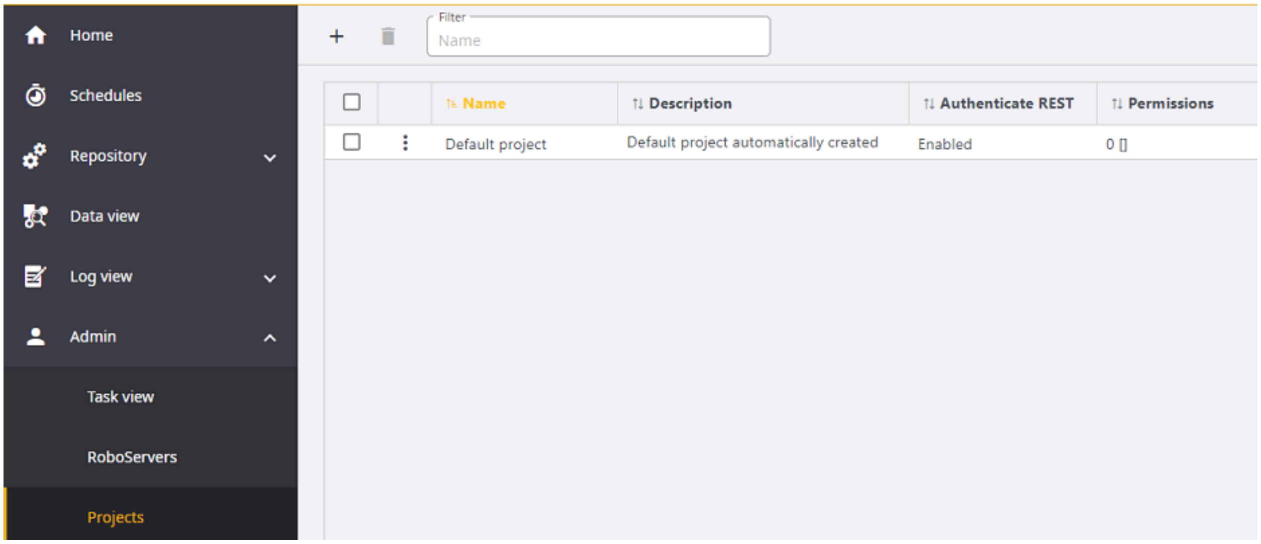
Create "Dev" User

1. On the Users tab, click the plus sign.

   The Create new user dialog box appears.
2. Specify the user name "Dev," password, full name, and email for the user and then select the group "Developers."
3. Click **OK**.

   The user appears in the table.

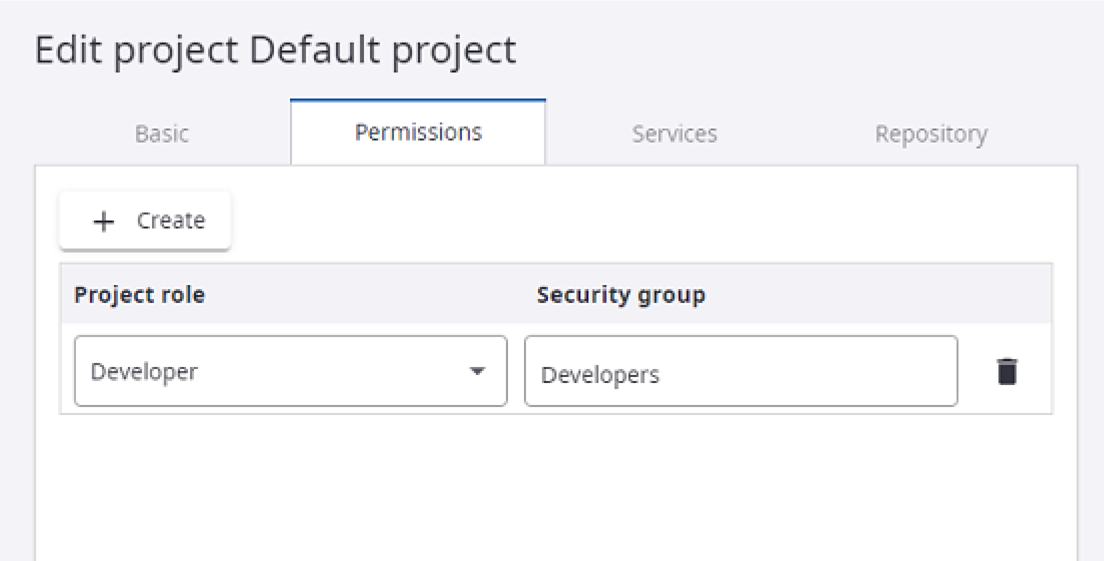
Now you need to assign permissions to the user so the user can log in. Log in as administrator and go to **Admin** > **Projects**. In this section, click the ▥ menu icon on the right and click to display the **Permissions** column. Currently, it displays "0" permissions for this project.



□		Name	Description	Authenticate REST	Permissions
□	⋮	Default project	Default project automatically created	Enabled	0 ▯

Initial Project Permissions

1. Click **Edit** from ⋮ context menu for the default project and go to the **Permissions** tab. There is a grid with the columns **Project role** and **Security group**. A project role determines a set of actions that may be performed inside Management Console, such as uploading robots, creating schedules, viewing logs, and so on. Within a project you assign a project role to a security group. That way, all users of the selected security group will be able to perform the actions allowed by the assigned project role.
2. Click the plus sign to add permissions in this project. This adds a new line to the grid, and inserts a drop-down box allowing us to select a project role. Select the project role **Developer**.
3. Now click the **Security group** column and select the **Developers** security group (of which our "Dev" user is a member). It should look like this:

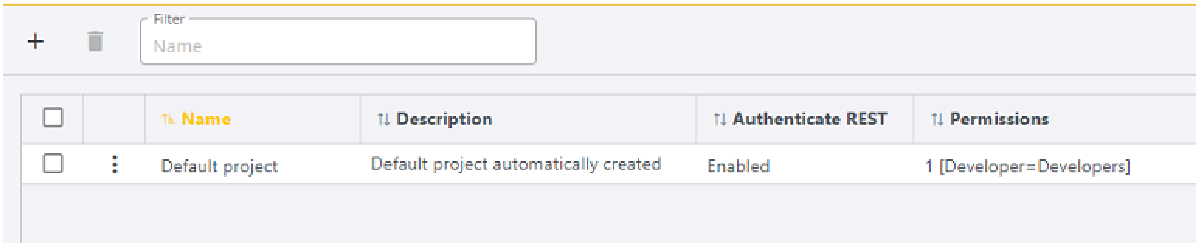


Edit project Default project

Project role	Security group
Developer	Developers

4. Now click **OK**.

   All members of the "Developer" group can now perform the actions allowed by the Developer role. The Permissions column for the default project now shows "1" permission.



□		Name	Description	Authenticate REST	Permissions
□	⋮	Default project	Default project automatically created	Enabled	1 [Developer=Developers]

Then, log in as the "Dev" user and see how the permissions are reflected in the Management Console. You log out by clicking the menu button in the upper right corner, then log in as the dev user. Now go to the **Log view**, select the RoboServer log in the left pane. Notice how the delete button is disabled, and hovering gives a tooltip message that you do not have permissions to delete RoboServer messages.

You can assign multiple roles to the same security group, and you can assign the same role to multiple security groups. If a user holds multiple roles, the user can do anything that at least one of the roles allow. With multiple projects in Management Console, users of different projects can be completely separated by assigning their groups to project-specific roles.

The predefined roles are suggestions, but using the `roles.xml` file, you can add any number of additional roles, or change the existing roles to fit your needs.

Actions that can be performed in the Settings, Backups, and License sections are only available to users that are members of the RPA Administrators group.

For using your LDAP user accounts, see the Advanced Configuration>LDAP Integration topic.

Security

In the WEB-INF/Configuration.xml file, it is possible to configure some additional security settings for the Management Console.

This is an example of security configuration.

```
<bean id="securityConfiguration" class="com.kapowtech.mc.config.SecurityConfiguration">
   <property name="jdbcDriverUpload" value="LOCALHOST"/>
</bean>
```

To change this behavior, modify the jdbcDriverUpload property. The following are the possible values the property can have.

- NONE: Upload of JDBC drivers is not allowed for any users
- LOCALHOST: The default value. The admin user is allowed to upload drivers if accessing the Management Console from the localhost
- ANY_HOST: The admin user is allowed to upload drivers from any host

When deploying Kofax RPA in Docker, configure this setting in the docker-compose file using the CONFIG_SECURITY_JDBCDRIVERUPLOAD environment variable and setting the value to NONE, LOCALHOST, or ANY_HOST. To allow the JDBC driver upload in Docker deployment, set this environment variable to ANY_HOST. See Docker Tools for Kofax RPA Deployment for details.

Telemetry feature

Kofax uses the telemetry feature to analyze and improve its products by collecting the certain metrics.

In Kofax RPA, the telemetry feature is enabled by default and collected data is reported to the Kofax telemetry service every 24 hours.

To disable the telemetry feature, use one of the following methods:

- Edit the WEB-INF/spring/common.xml file located inside the ManagementConsole.war web archive, which must be unpacked.

  Locate the telemetryEnabled property and set its value to false.
- In the Docker environment, set the SETTINGS_TELEMETRY_ENABLED variable to false.

Deployment Checklist

Use the following checklist to ensure that all deployment tasks are completed in the proper order.

Deployment Checklist

Item	Description
Download and install a supported version of Apache Tomcat Server	If your setup requires access to the Management Console outside of your corporate intranet, make sure SSL is set up to work with your Tomcat server.

Item	Description
Download and install Java	Management Console does not initialize correctly if Apache Tomcat is started using any version of Java other than the supported version. For more information, see the *Kofax RPA Technical Specifications* document available on the documentation site: https://docshield.kofax.com/Portal/Products/RPA/11.5.0-nlfihq5gwr/RPA.htm
Ensure that the JAVA_HOME variable is pointed to the Java installation.	
Start up the Apache Tomcat and confirm that the Apache Tomcat server will come online before attempting to deploy the Management Console application.	• You can use the catalina run command from the `\bin` directory to start the server. • Going to http://localhost:8080 in the browser should display the default Apache Tomcat start up page.
From a Kofax RPA installation, locate the `ManagementConsole.war` file that will be deployed under the Apache Tomcat `\webapps` directory	Expect errors on the initial start up of the application at this point, because the server has not been configured yet. We are simply unpacking the .war file so that we can easily gain access to the files we will need to edit to complete the installation and configuration process.
Turn the Apache Tomcat server off.	
Create a new database to be used by the Enterprise Management Console to hold its configuration.	• Select one of the support platforms. • The Kofax RPA installation contains the CREATE and DROP SQL scripts needed to create all the required database tables. See SQL Scripts for Kofax RPA Tables for details. Note that these scripts only need to be used if the user account that the application will use to connect to the database does not have the CREATE privileges on the schema.

Item	Description
Create the DB user account to be used by the application to connect to the database via JDBC.	
Create the Tomcat Context file: `ManagementConsole.xml`	• Validation query to be specified is database specific. The online help has all accepted values for all supported databases. • Do not forget to update the Username, Password, and DatabaseName parameters in the JDBC URL string with the correct values for your database.
Prior to starting up the application, notice that the Management Console database (as specified in the `ManagementConsole.xml`) does not have any tables related to the Management Console process at this time.	• When the application is started, the needed database tables are automatically created if they are not present assuming that the DBUser account has the CREATE privileges. • If the user does not have the CREATE privileges, the CREATE SQL scripts can be found in the `documentation\sql` directory in your Kofax RPA installation directory. See SQL Scripts for Kofax RPA Tables for details.
Do not forget to deploy the necessary JDBC .jar file under the Apache Tomcat installation directory.	• Deploying to `<APACHE_TOMCAT_INSTALL_DIR>\lib` makes it available to ALL apps running under Tomcat. • Deploying to `<APACHE_TOMCAT_INSTALL_DIR>\webapps\ManagementConsole\WEB-INF\lib` makes the JDBC .jar file only accessible to the Management Console application itself.
Restart the Apache Tomcat Server	Confirm the Tomcat main home page loads: http://localhost:8080
Try to login to the Enterprise Management Console as user - admin, password - admin	`http://localhost:8080/ManagementConsole`
Enter the Kofax RPA Software License Keys	See Enter License Information.

Docker Tools for Kofax RPA Deployment

Kofax supplies Docker tools for fast and easy deployment of Kofax RPA in your Linux and Windows environments. Docker tools for Kofax RPA help you build Docker images for the RPA components. Currently, images are available for the following components.

> ⓘ Due to the significantly higher resource consumption of the Windows Docker containers, the Linux version is recommended when available. Components that run only on Windows, support only the Windows Docker container version. Also, running Windows Docker containers is only supported on Windows Server hosts.

The official Kofax RPA images are now provided through Docker Hub. You can either download the required images using the links listed below, or build them using docker-compose files (for more information, see Deploy Kofax RPA using docker-compose files). Note that only Linux images are available.

Component	Linux	Windows
Management Console https://hub.docker.com/r/kofax/rpa-managementconsole	Yes	Yes
RoboServer https://hub.docker.com/r/kofax/rpa-roboserver	Yes	Yes
Robot File System	Yes	Yes
Synchronizer https://hub.docker.com/r/kofax/rpa-synchronizer	Yes	Yes
Kapplets https://hub.docker.com/r/kofax/rpa-kapplets	Yes	Yes
Document Transformation	No	Yes
Kofax Analytics for RPA	No	Yes

This chapter provides details on Docker tools and usage examples. For more information about Docker, see https://www.docker.com. For more information about manual deployment of Kofax RPA on a standalone server, see Tomcat Deployment.

Notes for Windows Docker

For production, we recommend that you run Docker containers under the preferred operating system container specified in the table above. Windows container images do not feature the High Availability feature or health check configurations. If a Windows Docker container is part of your Kofax RPA setup, which is currently only mandatory for Document Transformation and Kofax Analytics for RPA, we recommend that you deploy the setup in a hybrid Kubernetes cluster that includes Linux and Windows Server 2019 (or later) nodes. Previous versions result in significantly larger images.

When running Windows Docker containers, you need to make the following decisions that are specific to this operating system:

- **Isolation level**

  Decide on the runtime isolation mode to use: "process" and "Hyper-V" are available. The Windows Server default is "process," while the Windows 10 default is "Hyper-V." When using the "process" isolation mode, the container base version has to match the operating system version on the host. If you run the containers on a Windows Server 2019 host, the containers must be built based on a servercore version `ltsc2019`.

  When using the servercore version `ltsc2019` or later, some operating system components need to be optimized and added to the RoboServer image for it to work. Check the Dockerfile for RoboServer located at `docker-win\roboserver` and note the commented part where the dxva2.dll script is copied. Move a copy to the build folder or change the COPY command to point to its location as instructed in the comment.

  Also, before running the docker-compose file, ensure that you configure the `SERVERCORE_VERSION` build argument in the required Dockerfile accordingly. For example, if you use Windows Server 2019, set the argument to `ltsc2019`; if you use Windows Server 2022, set the argument to `ltsc2022`.
- **Version**

Decide on the Windows Server version to use. Windows base container images are significantly optimized starting with the Windows Server 2019 version. Using previous versions is not recommended due to the higher resource usage.

However, some Windows containers are not available in 2019 versions yet, such as the Microsoft SQL Server image. The official Microsoft image only supports Linux Docker containers. Other Microsoft provided images are not yet available as 2019 versions. Available options to follow are:

- Run in the "process" isolation mode on an operating system for which all prerequisites are available, such as Windows Server 2016, and build based on those images at an additional runtime resource cost.
- Run in the "Hyper-V" isolation mode with optimal settings for each container with the overhead of running virtual machines.
- Deploy third-party images that are not available in the optimal version (MS SQL Server) on a different computer. In this case, the official Microsoft image (Linux) is recommended.
- Wait for a Microsoft SQL Server version to become available based on "servercore:2019ltsc" or later.
- Wait for Windows Docker to support running Windows and Linux containers at the same time on a computer in the "process" isolation mode.

As running all components on one computer is not intended for production, we recommend that you use a hybrid Kubernetes cluster or a different form of a hybrid setup if Windows Docker containers are needed.

Prerequisites for RoboServer

Before running the docker-compose file with RoboServer included, in the Kofax RPA installation folder, navigate to the `docker-win\roboserver` folder. In the folder, locate and run the `copy_fonts.ps1` script to copy Windows system fonts to the `roboserver\Fonts` folder.

Prerequisites for Kofax Analytics for RPA

Before running the docker-compose file with Kofax Analytics for RPA included, follow these steps:

1. In the Kofax RPA installation folder, navigate to the `docker-win\kafrpa` folder, locate and run the `copy_fonts.ps1` script to copy Windows system fonts to the `docker-win\kafrpa\Insight\Fonts` folder.
2. Rename the Kofax Analytics for RPA project bundle to **kafrpa_bundle.zip** and copy it to the `docker-win\kafrpa\Insight` folder.
3. Rename the Insight installation package to **KofaxInsightSetup.msi** and copy it to the `docker-win\kafrpa\Insight` folder.
4. Obtain the required license file `Altosoft.Insight.License.xml` and copy it to the `docker-win\kafrpa` folder.

Prerequisites for Document Transformation Service

Before running the docker-compose file with Kofax Analytics for RPA included, follow these steps:

1. In the Kofax RPA installation folder, navigate to `docker-win\compose-examples`, open the `docker-compose-dt.yml` file, and then change the following line as applicable.

   ```
   DT_LICENSE_SERVER=put-your-license-server-here.example.com # Kofax license server
   ```

2. Copy the changed `docker-compose-dt.yml` file to `docker-win\documenttransformation`.
3. Copy the NLP bundles `KofaxTransformation-6.3.1.0_NLP-LanguageBundles` (`KofaxTransformation_Salience6.4_LanguageBundle_extended.msi`, `KofaxTransformation_Salience6.4_LanguageBundle_western-default.msi`, and `KofaxTransformation_Salience6.4_LanguageBundle_western-extended.msi`) to `docker-win\documenttransformation`.
4. Copy `KofaxRPADocumentTransformationService-11.5.0.0.msi` to `docker-win\documenttransformation`.

Deploy Kofax RPA using docker-compose files

This topic provides basic steps for deploying Kofax RPA on a standalone server under a Linux-based or Windows-based system. For information on the composition of the example docker-compose files for Windows and Linux supplied by Kofax RPA, see the next section.

> ℹ Currently, Docker Compose Version 2 uses a `docker compose` syntax. However, it still supports `docker-compose` features, and aliasing of `docker-compose` syntax to `docker compose` is enabled by default. You can run Compose V2 by replacing the hyphen (–) with a space, using `docker compose` instead of `docker-compose`. For more information, see the Docker documentation.

To deploy Kofax RPA using a docker-compose file, perform the following steps.

1. Install Kofax RPA on your computer.
2. Download Docker from https://www.docker.com and install it on your computer.
3. *Only applicable to Linux*. Add a new user to the docker group. For example, to add a "Kofax" user and allow the user access to Docker containers, replace `docker:x:<n>` with `docker:x:Kofax` in the `/etc/group` file. Log out and log in or restart the computer to update privileges.
4. In the `compose-examples` folder, select the docker-compose file that is best suited to your needs. Copy the file from the `compose-examples` folder to the root of the Kofax RPA installation folder, renaming the file to `docker-compose.yml` when necessary.

   For example, if you work on Windows, copy the `docker-compose.yml` file from `C:\Program Files\Kofax RPA 11.5.0.0\docker-win\compose-examples` to `C:\Program Files\Kofax RPA 11.5.0.0`.
5. Edit the docker-compose file as applicable to your environment and your needs.
   - Add license information in the `CONFIG_LICENSE_` variables.

     > ℹ Once the Management Console is running, you cannot change the license. To change the license settings, stop the composition and update the license variables in the docker-compose file. If the license settings are not included in the docker-compose file, you can change them in the Management Console without stopping the container.

   - Set the initial admin user name and password.

By default, the following default admin superuser name and password are used to authenticate to the Management Console:

User name: `admin`

Password: `admin`

Configure the `LOGIN_INITIAL_ADMIN_USER` and `LOGIN_INITIAL_ADMIN_PASSWORD` environment variables to use your custom credentials. After the credentials are set, they are stored in the database and cannot be changed again.

The admin user is created if the database is empty or no admin user exists, and the system is configured either to use local authentication or (for LDAP, and so on) the `LOGIN_CREATE_ADMIN_USER` variable is set to `true`. If the database already contains an admin user, the user is not created.

Start the Kofax RPA services. From the product installation folder, run the following command to build an image, start the services, and set the name "kofaxrpa" for the current container.

```
docker-compose -p kofaxrpa up -d
```

The first time you build the image, it may take some time to prepare it.

You can use separate commands to build an image and to start the services as follows.

> ℹ Because the tags on Docker Hub include the build numbers, the docker-compose files use the "latest" image by default.

- To build an image.
  - For Linux: `docker build -f docker/managementconsole/Dockerfile -t managementconsole:11.5.0.0 .`
  - For Windows: `docker build -f docker-win\managementconsole\Dockerfile -t managementconsole:11.5.0.0 .`

  Note the space and period at the end of line. The period refers to the current directory.
- Start the services.

  For both Linux and Windows: `docker-compose –p kofaxrpa up –d`

After you ran the docker-compose file, as soon as it starts the containers, you should be able to open your Docker host, which, by default, is located at `http://localhost`, and see that a Management Console is running with a RoboServer in a separate container. Additionally, you should be able to access other Kofax RPA containerized components that were included in the docker-compose file.

In RPA version 11.2 and later RoboServer writes logs in UTC time. RoboServers version prior to 11.2 by default write logs in local server time, which can lead to inconsistences in timestamps if both 11.2 and earlier versions of RoboServer log to the same logging database. If you connect RoboServers version prior to 11.2 to a Management Console version 11.2 or later, you can configure them to write log messages in UTC time instead of local server time by specifying the following option in the `roboserver-service>environment` section of the Docker configuration file:

```
- WRAPPER_JAVA_ADDITIONAL_1=-DwriteLogdbUtc=true
```

Note that the RoboServer must be updated to a fixpack version that supports this parameter. See the corresponding RPA fixpack readme file for details.

The following sections provide detailed information about docker-compose examples and configuration settings.

Docker-compose examples

Kofax RPA includes several docker-compose files with a few simple configurations in the `docker/compose-examples` folder. For Windows, the files are located in `docker-win\compose-examples`.

For Linux, all of the following examples rely on PostgreSQL as the configuration database for Management Console. Although Management Console can run on other databases, PostgreSQL is recommended for Management Console configuration data. Documentation for the PostgreSQL docker images is available at https://hub.docker.com/_/postgres. For Windows, the compose examples rely on Microsoft SQL Server as the configuration database.

For logging and robot data storage, any of the supported databases can be used. See "Supported Platforms" in the *Kofax RPA Installation Guide*.

Docker-compose files for a Linux environment

The following docker-compose files are provided for a Linux environment.

docker-compose-basic.yml

This configuration starts a Management Console, a RoboServer, a PostgreSQL database, and connects them. Before you start this configuration, you might want to enter your license information into the compose configuration to avoid having to type it upon Management Console startup. To scale the amount of RoboServers (in the "Production" cluster), use the following command.

```
docker-compose -p kofaxrpa up -d --scale roboserver-service=2
```

docker-compose-ha.yml

This configuration starts a Management Console, a RoboServer, a PostgreSQL database, and a load balancer based on the lightweight Traefik image.

Management Console is configured to run with High Availability enabled using multicast discovery (requires enterprise license). For more information, see Run on Docker Swarm with Management Console in High Availability.

Multicast discovery requires a network overlay that supports UDP multicast.

For this configuration to work optimally, enter your license information into the docker-compose file before bringing up the services. Also, edit the following line to fit the network assigned to your containers:

```
- CONFIG_CLUSTER_INTERFACE=172.20.0.*
```

You can find further instructions and variables in the `.md` file, which resides in the `docker` folder inside your Kofax RPA installation.

Execute the following steps to find out which network is used.

1. Start the composition.

   ```
   docker-compose -p kofaxrpa up -d
   ```

2. List the started containers.

   ```
   docker container ls
   ```

3. Use the following command to get the host name.

   ```
   docker exec kofaxrpa-managementconsole-service-1 hostname -i
   ```

4. Stop the composition before editing the docker-compose file.

   ```
   docker-compose -p kofaxrpa down
   ```

The `traefik.docker.network` variable value must include the name of the current container.

In our example, `traefik.docker.network=`**`kofaxrpa`**`_net`, as the container name is "kofaxrpa".

Wildcards can be used, so the IP address may be similar to `172.*.*.*`. Note that hazelcast does not accept wildcards instead of all numbers, and therefore `*.*.*.*` is not allowed.

When starting the composition, you can scale the number of running Management Console instances using the following command.

```
docker-compose -p kofaxrpa up -d --scale managementconsole-service=2
```

Running more than two instances of Management Console is possible, but doing so increases the database load. The load balancer is set up to use sticky sessions.

docker-compose-kapplets.yml

This configuration starts a Management Console, a RoboServer, a PostgreSQL database, and also adds Kofax RPA Kapplets and configures them.

docker-compose-ldap.yml

This is an example configuration that uses LDAP. It starts the `OpenLDAP` command in a container, which is normally not needed but is included as an example.

A related file, `ldap_ad_content.ldif`, is also included for testing purposes. Test this composition by running the following command.

```
docker-compose -p kofaxrpa up -d && docker -vv cp ./docker/compose-examples/
ldap_ad_content.ldif kapow_ldap-service_1:/ldap_ad_content.ldif && docker exec
```

```
kapow_ldap-service_1 ldapadd -x -D "cn=admin,dc=example,dc=org" -w admin -f /
ldap_ad_content.ldif
```

docker-compose-rfs.yml

This configuration starts a Management Console, a RoboServer, a PostgreSQL database, and also adds a Robot File System and configures it.

docker-compose-synchronizer.yml

This configuration starts a Management Console, a PostgreSQL, a RoboServer, and also adds Synchronizer and configures it.

If you want to disable the Docker JMX health check, remove the lines in the Dockerfile between:

`# ### start cutting here ###` and `# ### end cutting here ###`

Docker-compose files for a Windows environment

The following docker-compose files are provided for a Windows environment.

docker-compose.yml

This configuration starts all of the available Kofax RPA components, except Document Transformation: Management Console, RoboServer, Synchronizer, Robot File System, Kapplets, Kofax Analytics for RPA, and a MS SQL database.

docker-compose-dt.yml

This compose file is used to document the parameters used by the Document Transformation Windows Docker container. If you require Document Transformation, you can use the example `docker-compose-dt.yml` file as follows. Before running the file, make sure that you followed the prerequisites for Document Transformation Service listed in Notes for Windows Docker.

```
docker-compose -f .\docker-compose-dt.yml up
```

Optimize Docker build context size

The information in this section is applicable to Linux only.

> We do not recommend building your own images. However, if you do so, this section applies to you.

As all Docker images are built with the distribution root folder as the build context, building the images can take up too much time. To optimize the build context size, `dockerignore` files are added to the distribution. As every component has different requirements, a separate `dockerignore` file is provided for every component. The file must be located next to the Dockerfile and follow the same naming convention, except that it must have a `.dockerignore` suffix, such as `Dockerfile.dockerignore`.

To use this feature, the build with BuildKit must be enabled. The standard Docker build currently supports only the global `.dockerfile`. To enable this feature, set `DOCKER_BUILDKIT=1` in the environment, or define it in `/etc/docker/daemon.json`.

If you are building with `docker-compose`, you also need to set `COMPOSE_DOCKER_CLI_BUILD=1` in the environment. If you do not build with BuildKit, you can rename the `Dockerfile.dockerignore` file to `.dockerignore` and move it into the root of your build context. If you build multiple components, such as with `docker-compose`, you need to combine the `.dockerignore` content. However, we recommend a build with BuildKit as it is more efficient.

Use Docker secrets feature for storing passwords

To avoid specifying connection passwords directly in the docker-compose file on both Linux and Windows, you can use the Docker secrets feature to store your passwords in a safe location.

You can use the secrets feature for environment variables.

In the following example, we specify a password to connect to the PostgreSQL database on a Linux-based environment.

1. Create a text file with a password in it. One file must contain one password. For this example, we created a `postgrespassword.txt` file that includes a password to the PostgreSQL database.
2. In the docker-compose file that you want to use, create a section called `secrets` on the first level (no indent), specify a variable name on the second level, and specify a relative or absolute path to the file with a password on the third level of indent as follows.

```
secrets:
  postgrespassword:
    file: postgrespassword.txt
```

3. In the `services` > `postgres-service` > `environment` section of the `.yml` file, substitute the `POSTGRES_PASSWORD` variable with the `POSTGRES_PASSWORD_FILE` variable and refer to the variable specified earlier in the `secrets` section as follows:

```
services:
  postgres-service:
    image: postgres:10
    environment:
      - POSTGRES_DB=postgresdatabase
      - POSTGRES_USER=postgresuser
      - POSTGRES_PASSWORD_FILE=/run/secrets/postgrespassword
    secrets:
      - postgrespassword
```

Using this procedure as an example, you can set up the secrets feature for other passwords in the docker-compose file.

Set up database

For Kofax RPA, we recommend to store your Management Console configuration and repository data in a containerized database, and add external databases for data storage and (audit-) logs.

However, you might want to store the Management Console internal configuration data in an external or corporate database. To change the Management Console database, correct the environment variables for the database context and add the proper JDBC driver to the image/container.

In the Dockerfile for Management Console, which resides in `docker/managementconsole` for Linux and `docker-win\managementconsole` for Windows, the following line adds the current JDBC driver to the Management Console image in the JDBC folder.

Linux: `ADD --chown=${user}:${group} https://jdbc.postgresql.org/download/postgresql-<version>.jar /usr/local/tomcat/lib/jdbc/`

Windows: `ADD https://clojars.org/repo/com/microsoft/sqlserver/sqljdbc4/4.0/sqljdbc4-4.0.jar${CATALINA_HOME}/lib/jdbc/`, where `CATALINA_HOME=${KAPOW_HOME}\tomcat\apache-tomcat-@tomcatNumericVersion@` and `KAPOW_HOME=c:\kapow`

Change this line or add more lines to the Dockerfile to add the correct JDBC driver and the correct version of the database connector for your Java. Use the latest available version of the driver.

After editing the Dockerfile, you need to rebuild the image by running the following command:

Linux: `docker build -f docker/managementconsole/Dockerfile . -t managementconsole:@productVersion@`

Windows: `docker build -f docker-win\managementconsole\Dockerfile . -t managementconsole:@productVersion@`

Or run the simple version:

```
docker-compose -p kofaxrpa up -d
```

Back up and restore

The information in this section is applicable to Linux only. The Management Console image contains two scripts for backing up and restoring repository and configuration information.

backup.sh

To create a backup to be saved in `/kapow/backup`, run the following Docker command:

```
docker exec kapow_managementconsole-service_1 backup.sh [options]
```

The following script usage options are available:

```
backup.sh [options]
```

Options

Option	Description
`-u, --username`	*Required*. The user name to use when calling a Management Console.
`-p, --password`	*Required*. The password to use when calling a Management Console.
`-h, --host`	The hostname for a Management Console (default: localhost).
`-i, --project <Id>`	Project ID to use for backing up only a specific project.

Option	Description
`-c, --configurationOnly <Id>`	An ID to back up only the configuration and no project data.
`-n, --postfix`	Sets the postfix for the created backup file (default is <datetime>).

restore.sh

To restore a backup saved in `/kapow/backup`, run the following docker command:

```
docker exec kapow_managementconsole-service_1 restore.sh [options]
```

The following script usage options are available:

```
restore.sh [options]
```

Options

Option	Description
`-u, --username`	*Required*. The username to use when calling a Management Console.
`-p, --password`	*Required*. The password to use when calling a Management Console.
`-h, --host`	The host name for a Management Console (default: localhost).
`-f, --filename`	*Required*. The name of the backup file to restore.
`-a, --path`	The path to the backup file to restore (default: `/kapow/backup/`).

Pre-start checks

When starting a container, the `ManagementConsole` image, by default, runs the following checks.

1. Checks that the database configuration works by connecting to the database using the provided JDBC URI. It also tries to run the validation query once.
2. Checks that the LDAP configuration works. If LDAP is configured, each of the LDAP directories is checked by trying to connect and bind, and running a query to retrieve all groups. You can expand this test for debugging or validation purposes by adding a test username to use for more lookups.

If a check fails, the image retries for a configurable amount of time. If one of the checks does not succeed within the configured timeout, the container exits and you can review your configuration parameters.

You can bypass a check by setting the timeout for the check to zero (0). See the Environment variables section for more information.

Data folders

Logs, database data, and configuration from the containers are stored in folders listed below. To avoid having your container grow, these folders should be mounted to a volume or to the local file system. See the Docker documentation on volumes.

> ℹ Logs, database, and other data stored on these volumes cannot be reused when upgrading the version of Kofax RPA.

RoboServer

Data, logs, and configuration from the RoboServer container reside in the following folder:

Linux: `/kapow/data`

Windows: `C:\kapow\data`

ManagementConsole

Tomcat logs reside in the following folder:

Linux: `/usr/local/tomcat/logs`

Windows: `C:\kapow\tomcat\apache-tomcat-<tomcatVersion>\logs`

Environment variables

The following table lists some commonly used variables that you need to set up when deploying RPA using Docker. You can find the complete list of environment variables for different Docker containers relative to docker-compose files in the `README.md` file, which resides in the `docker` folder inside your Kofax RPA installation.

Variable	Default value	Description
`ROBOSERVER_CONNECTION_TYPE`	(NoConnectionType)	Specifies the connection type between the RoboServer and the Management Console. Select the required value: • `ClientConnectionType` • `SocketConnectionType` • `SocketSSLConnectionType` See RoboServer for details.
`RFS_MC_SHARED_SECRET`	( )	The shared secret used for authenticating Robot File System with the Management Console.
`ROBOSERVER_MC_SHARED_SECRET_FILE`	( )	The path to a file containing the shared secret to use when registering with the Management Console .
`LOGIN_LDAP_DIRECTORY_CONVERTTOUPPERCASE_N`	(true)	Convert group names to upper-case.
Variables for logging		

Variable	Default value	Description
`LOG4J_LOGGER_ADDITIONAL_COUNT`	(0)	The number of additional properties to add to the log4j.properties file. ⚠ The syntax of the log4j configuration changed. log4j2 is now used. Check the log4j2.properties syntax and specify your parameters accordingly.
`LOG4J_LOGGER_ADDITIONAL_KEY_N`	( )	The key of the additional property N., for example: `LOG4J_LOGGER_ADDITIONAL_KEY_1`
`LOG4J_LOGGER_ADDITIONAL_VALUE_N`	( )	The value of the additional property N. If you want to set logging level to debug, specify DEBUG as the value. See "Example: Values for debug logging" below.

Example: Values for debug logging

The following example sets logging to debug level and writes detailed logging information to a file on a tomcat server. See Apache Log4j 2 on the apache.org website for details.

⚠ The syntax of the log4j configuration changed. log4j2 is now used. Check the log4j2.properties syntax and specify your parameters accordingly.

```
- LOG4J_LOGGER_ADDITIONAL_COUNT=10

- LOG4J_LOGGER_ADDITIONAL_KEY_1=rootLogger.level

- LOG4J_LOGGER_ADDITIONAL_VALUE_1=DEBUG

- LOG4J_LOGGER_ADDITIONAL_KEY_2=logger.spring.level

- LOG4J_LOGGER_ADDITIONAL_VALUE_2=DEBUG

- LOG4J_LOGGER_ADDITIONAL_KEY_3=appenders

- LOG4J_LOGGER_ADDITIONAL_VALUE_3=A, file

- LOG4J_LOGGER_ADDITIONAL_KEY_4=appender.file.type

- LOG4J_LOGGER_ADDITIONAL_VALUE_4=File

- LOG4J_LOGGER_ADDITIONAL_KEY_5=appender.file.name

- LOG4J_LOGGER_ADDITIONAL_VALUE_5=file

- LOG4J_LOGGER_ADDITIONAL_KEY_6=appender.file.fileName

- LOG4J_LOGGER_ADDITIONAL_VALUE_6=/usr/local/tomcat/logs/output.log
- LOG4J_LOGGER_ADDITIONAL_KEY_7=appender.file.layout.type
- LOG4J_LOGGER_ADDITIONAL_VALUE_7=PatternLayout
- LOG4J_LOGGER_ADDITIONAL_KEY_8=appender.file.layout.pattern
- LOG4J_LOGGER_ADDITIONAL_VALUE_8=%d{HH:mm:ss,SSS} %-5p %c %equals{%x}{[]}{} - %m%n
- LOG4J_LOGGER_ADDITIONAL_KEY_9=rootLogger.appenderRefs
- LOG4J_LOGGER_ADDITIONAL_VALUE_9=A, file
- LOG4J_LOGGER_ADDITIONAL_KEY_10=rootLogger.appenderRef.file.ref
- LOG4J_LOGGER_ADDITIONAL_VALUE_10=file
```

Run on Docker Swarm with Management Console in High Availability

> The following procedure and setup are intended only as an example and should not be considered a recommendation.

The example in this section demonstrates how to set up Kofax RPA on a Docker swarm with more than one instance of Management Console (with High Availability mode enabled).

This section contains an example of setting up a Docker swarm that consists of two nodes, manager and worker, which provide a higher level of fault tolerance than using only one node. The example uses PostgreSQL.

The Management Console can run as several instances in a cluster, sharing configuration, log and repository data through a database, and sharing the cluster volatile state through the Hazelcast platform. This platform requires each instance of the Management Console to be able to discover and connect to the other instances in the cluster. The two discovery methods that are currently implemented are multicast and TCP. In this example procedure, we use the multicast (UDP) discovery method. The TCP discovery method was also successfully tested.

To make multicast UDP work in a Docker swarm, the Weave Net plugin 2.5.1 is used in this procedure.

Set up a minimal Docker swarm with Management Console

Before proceeding, ensure that you have two Docker hosts running with Linux kernel 3.8 or higher. In this procedure, the hosts are referred to as `host1` and `host2`.

1. Set up your Docker swarm cluster.

   a. Create a manager node on `host1`.

   ```
   host1$ docker swarm init --advertise-addr <host1_IP>
   ```

   b. Join `host2` into the swarm as worker node.

   ```
   host2$ docker swarm join --token <token> --advertise-addr <host2_IP>
    <host1_IP>:<port>
   ```

   Replace `<token>` with the token retrieved from the first command.

2. Install the Weave Net plugin on the two hosts and enable the multicast feature as follows.

```
host1$ docker plugin install store/weaveworks/net-plugin:2.5.2 --grant-all-
permissions --disable
host1$ docker plugin set store/weaveworks/net-plugin:2.5.2 WEAVE_MULTICAST=1
host1$ docker plugin enable store/weaveworks/net-plugin:2.5.2
```

```
host2$ docker plugin install store/weaveworks/net-plugin:2.5.2 --grant-all-
permissions --disable
host2$ docker plugin set store/weaveworks/net-plugin:2.5.2 WEAVE_MULTICAST=1
host2$ docker plugin enable store/weaveworks/net-plugin:2.5.2
```

3. On your manager node, create a Docker network using the Weave Net plugin as the driver as follows.

```
host1$ docker network create --driver=store/weaveworks/net-plugin:2.5.2 --
attachable weavenet
```

4. Build the Management Console and RoboServer Docker images on all of the nodes in your Docker swarm. With a Docker repository, you can push the images to it. For information on building the Docker images, see Docker Tools for Kofax RPA Deployment.

5. Create a file called **docker-compose.yml**, adapting the following lines to your environment.

```
version: '3.2'
    networks:
      net:
        external:
          name: weavenet
    services:
      loadbalancer:
        image: traefik
        command: --docker --docker.swarmmode --docker.watch --web --loglevel=DEBUG
        ports:
          - 80:80
        volumes:
          - /var/run/docker.sock:/var/run/docker.sock
        networks:
          - net
        deploy:
          mode: global
          placement:
            constraints: [node.role == manager]
      postgres-service:
        image: postgres:10
        environment:
          - POSTGRES_USER=scheduler
          - POSTGRES_PASSWORD=schedulerpassword
          - POSTGRES_DB=scheduler
        networks:
          - net
      managementconsole-service:
        image: managementconsole:@productVersion@
        depends_on:
          - postgres-service
        environment:
          - CONTEXT_RESOURCE_VALIDATIONQUERY=SELECT 1
          - CONTEXT_RESOURCE_USERNAME=scheduler
          - CONTEXT_RESOURCE_PASSWORD=schedulerpassword
          - CONTEXT_RESOURCE_DRIVERCLASSNAME=org.postgresql.Driver
          - CONTEXT_RESOURCE_URL=jdbc:postgresql://postgres-service:5432/scheduler
          # enter your license here, or type it through the GUI in first login
          - CONFIG_LICENSE_NAME=
          - CONFIG_LICENSE_EMAIL=
          - CONFIG_LICENSE_COMPANY=@licenseCompany@
          - CONFIG_LICENSE_PRODUCTIONKEY=@licenseProduction@
          - CONFIG_LICENSE_NONPRODUCTIONKEY=@licenseNonProduction@
            # change to use your own, match with the Roboserver setting
 ROBOSERVER_MC_SHARED_SECRET
          -
 SERVICE_AUTHENTICATION_ROBOSERVER_SHARED_SECRET=@RoboserverSharedSecret@
          - CONFIG_CLUSTER_JOINCONFIG=multicastCluster
          # change to fit your network
          - CONFIG_CLUSTER_INTERFACE=10.*.*.*
          - CONFIG_CLUSTER_MULTICAST_GROUP=224.2.2.3
          - CONFIG_CLUSTER_MULTICAST_PORT=54327
          - LOG4J_LOGGER_COM_HAZELCAST=ERROR, A
        deploy:
          replicas: 2
          labels:
            traefik.docker.network: weavenet
            traefik.port: 8080
            traefik.backend.loadbalancer.sticky: "true"
            traefik.frontend.rule: PathPrefix:/
        networks:
          - net
      roboserver-service:
        image: roboserver:@productVersion@
        depends_on:
          - postgres-service
        networks:
          - net
        environment:
          - ROBOSERVER_ENABLE_MC_REGISTRATION=true
          - ROBOSERVER_MC_URL=http://loadbalancer/
          - ROBOSERVER_MC_CLUSTER=Production
          # change to use your own, match with the Management Console setting
 SERVICE_AUTHENTICATION_ROBOSERVER_SHARED_SECRET
          - ROBOSERVER_MC_SHARED_SECRET=@RoboserverSharedSecret@
          - ROBOSERVER_CONNECTION_TYPE=SocketConnectionType
          - WRAPPER_MAX_MEMORY=2048
```

ⓘ If you are copying the file, make sure to keep the original layout. Incorrect layout may result in an invalid file.

Also note:

- Replace **@productVersion@**, **@licenseCompany**, **@licenseProduction@**, **@licenseNonProduction@**, and **@RoboserverSharedSecret@** with your appropriate values.
- **CONFIG_CLUSTER_INTERFACE** must fit the Weave Net subnet in your Docker swarm. You can find the subnet using the following command on your manager node: `host1$ docker network inspect weavenet`
- The common network **net** refers to the external network **weavenet** that you created before.
- **Traefik** is used as the load balancer with sticky sessions.
- This setup runs a PostgreSQL database on a temporary volume. In a real production setup, the database also needs to be run clustered and with a permanent data volume.
- To run multiple instances of the RoboServer, you can add deployment constraints to **roboserver-service**.
- All services must use **endpoint_mode: dnsrr** (the default setting **endpoint_mode: vip** is not supported by the Weave Net plugin)

6. Deploy the service stack on the manager node as follows.

   ```
   host1$ docker stack deploy -c docker-compose.yml rpa
   ```

   > Starting **managementconsole-service** with two replicas on an empty database may lead to a race condition. If you experience this situation, change the line **replicas: 2** to **replicas: 1** and run the preceding command. Wait until the services are started, change the line back, and then run the preceding command again.

   To stop the services, use the following command.

   ```
   host1$ docker stack down rpa
   ```

7. After deploying the stack, the Management Console can be accessed at the following URL.

   `http://host1/`

Advanced Configuration

LDAP and CA Single Sign-On Integration

This topic describes how to use LDAP and CA Single Sign-On authentication.

Also, Kofax RPA supports LDAPS (LDAP over SSL). See "Secure LDAPS" and "Checklist for solving SSL connection errors when using LDAPS" later in this section.

User origin and authentication

A user that logs to a Management Console is identified by a user name and origin. The User origin field on the **Users & groups** page of the Management Console contains information about the user creation method as in the following table.

User origin	Description
`unknown`	The user was created after restoring a backup.
`internal`	The user was created manually on the Users & groups page.
`saml`	The user was created after logging via SAML.
`siteminder`	The user was created after logging via SiteMinder.
`ldap#{ldapDirectoryIdentifier}`	The user was created after logging via LDAP.

Note the following for user authentication.

- If a user logs in and there is another user with the same name and `unknown` origin, the origin is changed to the one based on the current log in and a new user is not created.
- If you do not use any of the external identity providers, such as SAML, LDAP, or SiteMinder, you can change `unknown` origin to `internal` by clicking **Set internal origin** for the selected user on the **Users & groups** page in the Management Console.
- `ldapDirectoryIdentifier` is a required property in `login.xml`.

- `LdapDirectory` is optional and defaults to 0 (zero) if not explicitly assigned.
- The Management Console does not start if there are more than one LDAP Directory with the same `ldapDirectoryIdentifier`.
- You can use LDAP and SAML authentication at the same time by specifying `true` for the `useLdap` and `useSaml` options in the `authenticationConfiguration` bean in `login.xml`.
- When using both LDAP and SAML authentication, SAML is used for SSO in the Management Console and LDAP is used for services authentication, such as connecting to Design Studio and so forth.

LDAP Integration

To use LDAP for authentication, enable the LDAP authentication and edit the LDAP definition in the `login.xml` file.

To enable the LDAP authentication, set the `useLdap` property to `true` as follows:

```
<bean id="authenticationConfiguration"
 class="com.kapowtech.scheduler.server.spring.security.AuthenticationConfiguration">
        <property name="useLdap" value="true"/>
        <property name="useSiteMinder" value="false"/>
</bean>
```

In `login.xml`, you can find the following definition:

```
<bean id="ldapDirectories" class="com.kapowtech.mc.config.LdapDirectories" lazy-
init="true">
        <property name="directories">
            <list>
                <bean class="com.kapowtech.mc.config.LdapDirectory">
                  <!-- Property defining unique ldap directory name, used as part of
 user's origin field.
                  Must be different for each LdapDirectory -->
                  <property name="ldapDirectoryIdentifier" value="0"/>
                    <!-- List of security groups which will be application
 administrators.
                    Users in these groups will have all permissions. Only users in
 these groups can
                    access the backup tab and create and restore backups -->
                    <property name="adminGroups">
                        <list>
                            <value>KAPOWADMIN</value>
                            <value>ENGINEERING/value>

                   <property name="administratorGroups">
                    <list>
                        <value>RPAADMINISTRATORS</value>
                    </list>
                   </property>
                    </property>
                    <property name="ldapServerURL" value="ldap://
ldap.kapowdemo.com:389"/>
                    <property name="userDn" value="CN=LDAP
 test,CN=Users,DC=kapowdemo,DC=local"/>
                    <property name="password" value="change-me"/>
                    <property name="userSearchBase"
 value="OU=Users,OU=TheEnterprise,DC=kapowdemo,DC=local"/>
                    <property name="userSearchFilter"
 value="(userPrincipalName={0}@kapowdemo.local)"/>
                    <property name="userSearchSubtree" value="true"/>
                    <property name="groupSearchBase" value="OU=Security
 Groups,OU=TheEnterprise,DC=kapowdemo,DC=local"/>
                    <property name="groupSearchFilter" value="(member={0})"/>
                    <property name="groupRoleAttribute" value="cn"/>
                    <property name="groupSearchSubtree" value="true"/>
                    <property name="allGroupsFilter" value="(cn=*)"/>
                    <property name="fullNameAttribute" value="displayName"/>
                    <property name="emailAttribute" value="userPrincipalName"/>
                    <property name="referral" value="follow"/>
                </bean>
            </list>
        </property>
    </bean>
```

This defines a list of `ldapDirectory` beans named `ldap` and represents a list of connections to LDAP servers. Kofax RPA supports multi-forest LDAP integration, so you can specify several connections to LDAP directories. Each bean defines a number of properties that controls the LDAP integration. If you are familiar with the way Tomcat integrates to LDAP, this should be quite familiar as well.

Group names must be unique across all LDAP servers in the list.

Secure LDAP

Kofax RPA supports LDAPS (LDAP over SSL) with Management Console. To use LDAPS, the `ldapServerURL` property must be set as follows:

```
<property name="ldapServerURL" value="ldaps://<hostname>:<port>"/>
```

By default, the LDAPS port is 636.

Deployment Checklist

Property	Description
`ldapDirectoryIdentifier`	An LDAP directory name, used as part of user's origin field in a Management Console. This name must be unique for each LDAP Directory.
`adminGroups`	List of LDAP groups mapped to the admin superuser in Management Console who has access to everything.
`administratorGroups`	List of LDAP groups mapped to RPA Administrators in Management Console. Users belonging to these LDAP groups have all rights for all projects (excluding special admin user rights), such as mapping of LDAP groups to roles in Management Console.
`ldapServerURL`	URL to the LDAP server. This uses either the ldap:// or ldaps:// protocol.
`userDn`	DN (distinguished name) used to log in to LDAP to authenticate other users.
`password`	Password for the userDN account. As the password will be stored in clear text in this file you should use an account that only has "read" access.
`userSearchBase`	Subdirectory in the LDAP tree where users can be found.
`userSearchFilter`	Filter that is applied to find the username.
`userSearchSubtree`	Set to true if users may be located in the subdirectory of the userSearchBase.
`groupSearchBase`	Subdirectory in the LDAP tree where groups can be found.

Property	Description
`groupSearchFilter`	Filter that is applied to identify the users in this group.
`groupRoleAttribute`	Attribute that holds the group name.
`groupSearchSubtree`	Set to true if groups may be located in the subdirectory of the groupSearchBase
`convertToUpperCase`	Should the group names be converted to upper case, true by default.
`allGroupsFilter`	Optional. Controls which groups are displayed when creating project permissions, see below.
`fullNameAttribute`	Attribute to fetch the full name of the user.
`emailAttribute`	Attribute to fetch the email of the user.
`referral`	Set to "follow" to allow redirection to sub nodes in the LDAP tree.

To use an LDAP account to administer a Management Console, you must add one of the groups that you are a member of to the `administratorGroups` bean in `login.xml`, as described in Project Permissions. Note that anyone who is a member of a group listed in `administratorGroups` is the Management Console administrator, so you may want to create a new LDAP group for this purpose. Use the upper case group name if `convertToUpperCase` is true.

When you select a project permission in the Management Console, you can see that all the group names are pulled from LDAP to populate the list. The groups are located by using the `groupRoleAttribute` to construct a filter to fetch all groups. Sometimes you do not want all LDAP groups displayed here, in which case override this behavior by providing your own filter. This is done by adding an additional property to the `LdapLogin.`

`<property name="allGroupsFilter" value="(cn=*)"/>`: Finds all group names, if the group name is in the cn attribute (this is the default).

If you only want to find groups starting with the letter 'e' you can use the following code

```
<property name="allGroupsFilter" value="(cn=E*)"/>
```

The filter uses basic LDAP queries. See LDAP documentation for more complex queries.

Checklist for solving SSL connection errors when using LDAPS

If you experience errors connecting using LDAPS, check the following:

- LDAPS requires that the certificate presented by the LDAP server is trusted by the java running the tomcat. Import the public certificate to the Java keystore that your application uses.
- Make sure certificates are imported into the correct truststore, such as if you have multiple instances of JRE or JDK.
- Make sure the correct truststore is in use. If `-Djavax.net.ssl.trustStore` is configured, it overrides the location of the default truststore.
- If connecting to a mail server, such as Exchange, ensure that authentication allows plain text.
- Verify that the target server is configured to serve SSL correctly. This can be done with the SSL Server Test tool.
- Check if your anti virus tool has "SSL Scanning" that blocks SSL and TLS. Disable this feature or set exceptions for the target addresses.

CA Single Sign-On Integration

Management Console supports pre-authentication using CA Single Sign-On. With CA Single Sign-On the identity of the user is established before accessing a Management Console, and the user's identity is communicated through an HTTP header. The identity of the user must be in the form of an LDAP distinguished name for a Management Console to resolve the user's LDAP group memberships.

CA Single Sign-On integration is disabled by default. You can enable it by setting the `useSiteMinder` property to `true` in the `login.xml` file as follows:

```
    <bean
class="com.kapowtech.scheduler.server.spring.security.AuthenticationConfiguration"
id="authenticationConfiguration">
        <property name="useLdap" value="false"/>
        <property name="useSiteMinder" value="true"/>
    </bean>
```

```
    <bean class="com.kapowtech.mc.config.SiteMinderConfiguration"
id="siteMinderConfiguration">
        <property name="headerName" value="sm_userdn"/>
        <property name="accountAttribute" value="sAMAccountName"/>
        <property name="accountAttributePattern" value="(.*)"/>
    </bean>
```

After you enable the CA Single Sign-On integration, specify the name of the HTTP header that contains the user's distinguished name. The `accountAttribute` property identifies which of the user's LDAP attributes is used as an account name (The default uses the `sAMAccountName`, which is the user's Windows login name). The `accountAttributePattern` property specifies how the account name is parsed from the attribute value, which must be a regular expression (with a single set of parentheses identifying the account name), and the `(.*)` value in the `accountAttributePattern` property means everything in the attribute. To extract the account name from the user's email address in the configuration, you can specify the following:

```
    <bean class="com.kapowtech.mc.config.SiteMinderConfiguration"
id="siteMinderConfiguration">
        <property name="headerName" value="sm_userdn"/>
        <property name="accountAttribute" value="userPrincipalName"/>
        <property name="accountAttributePattern" value="([^@]*)@.*"/>
    </bean>
```

`([^@]*)@.*` will match an email address and extract everything before `@` as the account name.

As CA Single Sign-On uses part of the LDAP login configuration, you need to add a user group to the `administratorGroups` bean, before you can start configuring the Management Console.

It is not possible to log out of the system, as the presence of the CA Single Sign-On header means that you are always authenticated. To log out, close your browser.

Limitations

CA Single Sign-On integration only works when a Management Console is accessed through a browser. However, if a Management Console is accessed by applications that are not browsers, the CA Single Sign-On authentication mechanism is not used and such services require a set of credentials (username and password) set in the Management Console. These clients include, but are not limited to, the following:

Design Studio

Robot developers need to access Management Console to obtain a developer seat license, to upload robots, and get database configurations and other settings stored in the Management Console. This requires access to the following URLs (relative to the context path where Management Console is deployed).

- /License/*
- /secure/*
- /IDESettings/*
- /rest/*
- /ws/*

Access to the URLs above is protected by basic authentication with passwords set in the Management Console.

REST services

REST services are protected by basic authentication by default, but robots can be exposed without authentication as REST services. Such services are typically invoked by external applications. REST uses /rest/* URL.

RoboServer

RoboServer authenticates using a shared secret for registering to a cluster and for resource requests.

Desktop Automation Service

Desktop Automation Service authenticates using a shared secret or Management Console for registering and status updates.

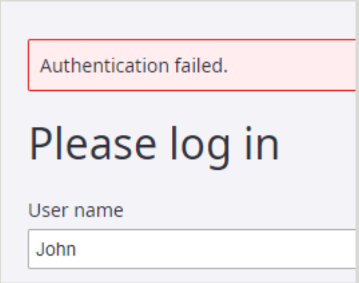
Any application based on Kofax RPA Java or .Net API

Any application based on Kofax RPA Java or .Net API is protected by basic authentication when accessing a Management Console.

Example: Usage

- **Jane** is the designated Management Console administrator. Because she is added to the **Administrators** group in the Active Directory (AD), this group is mentioned in `login.xml`, and once she authenticates her browser with CA Single Sign-On, she is immediately logged in to the Management Console.

- **John** is a robot developer who is a member of the group **Users** in the AD and currently he is declined when trying to log in to the Management Console.

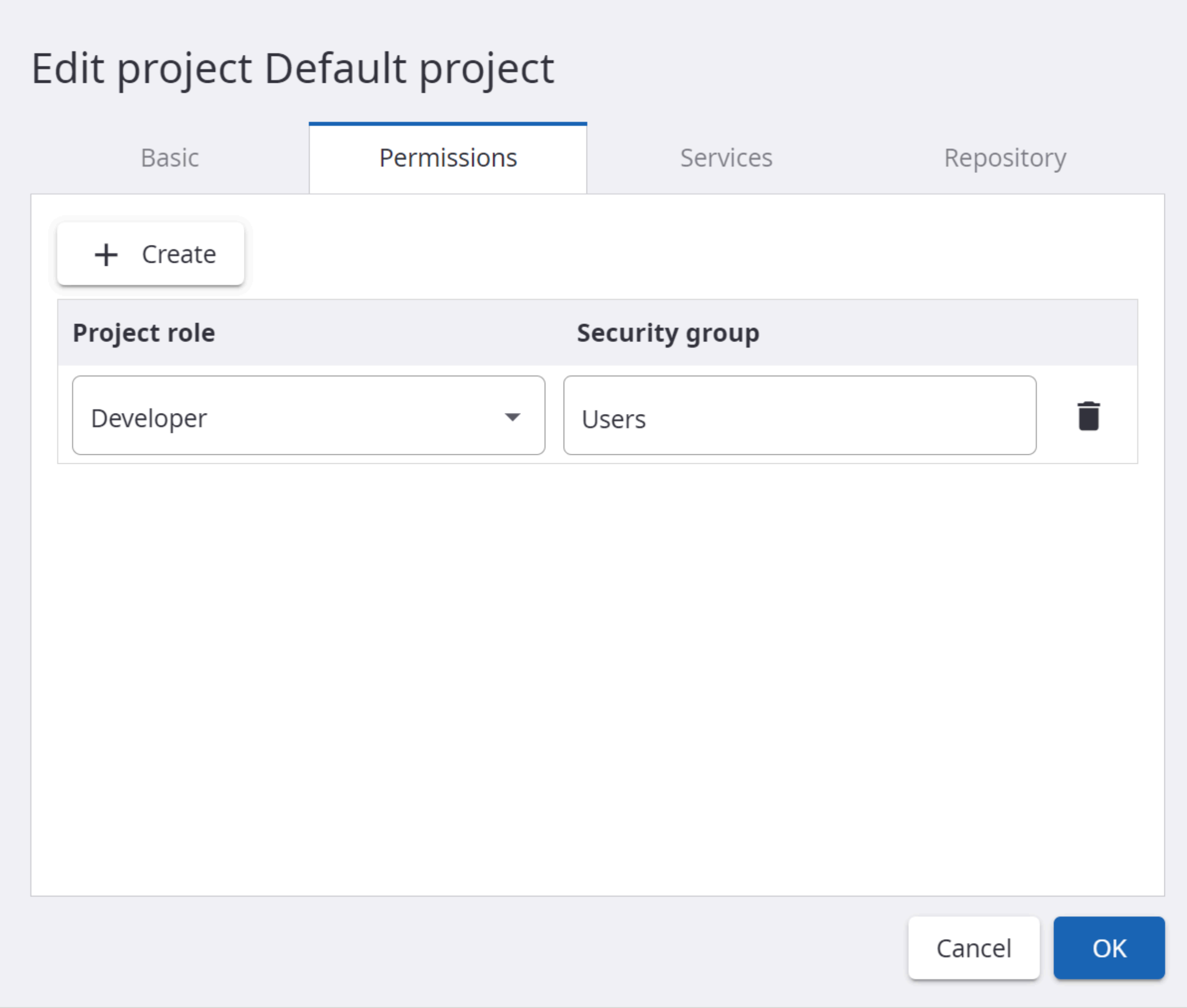


Authentication failed.

Please log in

User name: John

Jane needs to assign privileges to the **Users** group in the Management Console > **Admin** > **Projects** for the selected project before John can log in. For example, the Users group can be assigned a Developer role as shown here.



Edit project Default project

Project role	Security group
Developer	Users

Now John can log in to the Management Console using CA Single Sign-On.

When John starts Design Studio, he needs to enter the required URL in the **Enter License Information** dialog box and authenticate to the Management Console in the opened browser to acquire a license.

Jane can also create local service users that are never allowed to log in to the Management Console, such as for the RoboServer service or Desktop Automation Service. Note the following:

- The groups you select for service users must originate only from Active Directory.
- The group must have the appropriate privileges set in the Management Console.

SAML Single Sign-On Integration

Management Console supports user pre-authentication using SAML Single Sign-On.

The following procedure is written with the assumption that you have already created and configured a SAML application in your identity provider. For an example configuration procedure, see Example OneLogin Configuration.

Management Console with SAML integration cannot start without a license. Install a valid license before you can use the application.

Integration of the Management Console with SAML Single Sign-On is disabled by default. To enable it, in the `login.xml` file, locate and set the `useSaml` property to `true`.

You can use LDAP and SAML authentication at the same time by specifying `true` for the `useLdap` and `useSaml` options in the `authenticationConfiguration` bean in `login.xml`. See User origin and authentication for details.

```
 <bean
class="com.kapowtech.scheduler.server.spring.security.AuthenticationConfiguration"
id="authenticationConfiguration">
   <property name="useLdap" value="false"/>
   <property name="useSiteMinder" value="false"/>
  <property name="useSaml" value="true"/>
```

After you enable the integration in `login.xml`, you need to configure the identity provider and the service provider settings. In the `saml.xml` file located in the folder `\WEB-INF\spring`, modify the following properties to match your setup.

After changing the `login.xml` and `saml.xml` files, restart Tomcat.

Property	Description
`assignGroups`	When set to `false`, the administrator manually assigns users to the groups, so that they have access to Management Console. Default is `true`.
`forceAuthN`	When set to `true`, the identity provider re-authenticates users and does not rely on previous authentication events. Default is `false`.

Property	Description
`groupsAttributes`	Group attribute. User access to Management Console is managed by means of groups that a particular user belongs to. Specify a name or a list of names matching attribute names in SAML Assertion message that contains the list of groups assigned to a user in the identity provider.
`adminGroups`	Admin groups. List of identity provider groups mapped to the admin superuser in Management Console who has access to everything.
`administratorGroups`	Administrator groups. The users in this group get administrative rights to all of the RPA projects. Also, it is possible to create custom administrator groups using SAML. For more information on the user roles and groups, see "Predefined User Roles" in the *Kofax RPA Administrator's Guide.*
`email`	This is the name of the attribute from the SAML Response that contains the user's email.
`firstname`	This is the name of the attribute from the SAML Response that contains the user's first name.
`lastname`	This is the name of the attribute from the SAML Response that contains the user's last name.
`idpName`	Name of the identity provider. Possible values are `OKTA`, `ONELOGIN`, and `AZURE`. Otherwise, set to `DEFAULT`.
`idpGroupNameSeparator`	Delimiter to use to separate identity provider group names. Takes effect only if the identity provider specified with `idpName` is OneLogin. Otherwise, this property is ignored.
`idpUserNameRegex`	Update the existing parameter by adding the `^[\p{L}0-9 ]*$` pattern if some of the user names contain special characters.
`entityId`	URL of a Management Console, plus `saml/login`. You can get this URL from your SAML application. Example: `http://localhost:8080/ManagementConsole/saml/login`
`entityBaseURL`	Base URL of a Management Console. Example: `http://localhost:8080/ManagementConsole`
`maxAuthenticationAge`	Validity of single sign-on in seconds. Time window allowed for users to sign in after they are initially authenticated with the identity provider. By default, it is 86400 seconds, which is 24 hours. Use this property to change the default.
`responseSkew`	Inaccuracy tolerance in seconds for comparing clocks between the identity provider server and the computer where Management Console is deployed. As the clock synchronization may not be fully accurate, a tolerance of 60 seconds is applied by default. Use this property to change the default.

Property	Description
`maxAssertionTime`	Validity of assertions processed during the single sign-on. If the assertion time reaches the configured limit, the authentication becomes invalid. By default, it is limited to 6000 seconds, which is 100 minutes. Use this property to change the default.
`java.lang.String` of `HTTPMetadataProvider` bean	URL to the identity provider metadata. You can get this URL from your SAML application. Example: `http://example.okta.com/saml/metadata/222670a0-2b96-48ef-975a-b7267446d09e` Some identity providers, such as Microsoft Azure, do not require this property.
`java.io.File` of `FilesystemMetadataProvider` bean	Path to the XML file containing the identity provider metadata. Use this property if your identity provider does not allow reading of metadata in real time. Some identity providers, such as OneLogin, do not require this property.
`useSamlSingleLogout`	Default is `false`. If set to `false`, after logging out from Management Console, the Management Console session is destroyed and a message appears confirming successful logout. If set to `true`, the user logs out of the SAML application in your identity provider when clicking **Log out** in Management Console.

The following snippets from the example `saml.xml` file contain the properties you need to configure.

```
<bean id="samlEntryPoint" class="org.springframework.security.saml.SAMLEntryPoint"
 lazy-init="true">
    <property name="defaultProfileOptions">
        <bean class="org.springframework.security.saml.websso.WebSSOProfileOptions">
            <property name="includeScoping" value="false"/>
            <property name="forceAuthN" value="false"/>
            <property name="passive" value="false"/>
        </bean>
    </property>
    <property name="filterProcessesUrl" value="/saml/entry"/>
</bean>
```

```
<bean id="samlAuthenticationProvider" class="com.kapowtech.scheduler.server.spring.sec
urity.GroupProvidingSAMLAuthenticationProvider" lazy-init="true"> <constructor-arg ref
="platformEMF"/>
    <property name="internalAuthenticationProvider"
 ref="internalAuthenticationProvider"/>
    <property name="customerNameMapper" value="false"/>
    <property name="customerNameMapperIdentifier" value=""/>
    <property name="groupsAttributes">
        <list>
            <value>groups</value>
        </list>
    </property>
    <property name="adminGroups">
        <list>
            <value>KapowAdmins</value>
        </list>
    </property>
    <property name="assignGroups" value="true"/>
    <property name="consumer" ref="webSSOprofileConsumer"/>

    <property name="email" value="email"/>
    <property name="firstname" value="firstname"/>
    <property name="lastname" value="lastname"/>
    <property name="idpName" value="ONELOGIN"/>
    <property name="idpGroupNameSeparator" value=";"/>
    <property name="idpUserNameRegex" value="^[a-zA-Z0-9]*$"/>
    <property name="idpEmailRegex" value="^[A-Z0-9._%+-]+@[A-Z0-9.-]+\\.[A-Z]{2,6}$"/>
</bean>
```

```
<bean id="useSamlSingleLogout" class="java.lang.Boolean">
      <constructor-arg value="false"/>
</bean>
```

```
<bean id="metadataGeneratorFilter"
 class="org.springframework.security.saml.metadata.MetadataGeneratorFilter">
    <constructor-arg>
        <bean class="org.springframework.security.saml.metadata.MetadataGenerator">
            <property name="entityId" value="http://localhost:8080/ManagementConsole/
saml/login"/>
            <property name="requestSigned" value="false"/>
            <property name="entityBaseURL" value="http://localhost:8080/
ManagementConsole"/>
               <property name="extendedMetadata">
                <bean
 class="org.springframework.security.saml.metadata.ExtendedMetadata">
                    <property name="idpDiscoveryEnabled" value="false"/>
                    <property name="signMetadata" value="false"/>
                </bean>
            </property>
        </bean>
    </constructor-arg>
</bean>
```

```
<bean id="webSSOprofileConsumer"
 class="org.springframework.security.saml.websso.WebSSOProfileConsumerImpl">
        <property name="maxAuthenticationAge" value="86400"/>
        <property name="responseSkew" value="600"/>
        <property name="maxAssertionTime" value="6000"/>
    </bean>
```

```
<bean id="metadata"
 class="org.springframework.security.saml.metadata.CachingMetadataManager">
    <constructor-arg>
        <list>
           <bean class="org.opensaml.saml2.metadata.provider.HTTPMetadataProvider"
 lazy-init="true">
                <constructor-arg>
                    <value type="java.lang.String">http://example.okta.com/saml/
metadata/222670a0-2b96-48ef-975a-b7267446d09e</value>
                </constructor-arg>
                <constructor-arg>
                    <value type="int">10000</value>
                </constructor-arg>
                <property name="parserPool" ref="parserPool"/>
             </bean>
```

```
            <bean
 class="org.opensaml.saml2.metadata.provider.FilesystemMetadataProvider">
                      <constructor-arg>
                          <value type="java.io.File">classpath:security/idp.xml</value>
                      </constructor-arg>
                      <property name="parserPool" ref="parserPool"/>
                  </bean>
        </list>
    </constructor-arg>
</bean>
```

Example OneLogin Configuration

This section provides an example procedure on how to configure a SAML application in the OneLogin identity provider for use with Kofax RPA.

1. On the OneLogin website, create an account.
2. When the account is created, open the page {your-domain-name}.onelogin.com and open the Administration page.
3. On the menu, click **APPS** and select the option to add a new application. Select the following application type: **SAML Test Connector (Advanced)**.
4. On the **Configuration** tab, set the application parameters as shown in this table. Leave the other parameters as they are.

Parameter	Value
RelayState	http://<IP-address>:8080/ManagementConsole/saml/login
Audience	http://<IP-address>:8080/ManagementConsole/saml/login
Recipient	http://<IP-address>:8080/ManagementConsole/saml/login
ACS (Consumer) URL Validator	http://<IP-address>:8080/ManagementConsole/saml/login
ACS (Consumer) URL	http://<IP-address>:8080/ManagementConsole/
Single Logout URL	http://<IP-address>:8080/ManagementConsole/saml/SingleLogout
Login URL	http://<IP-address>:8080/ManagementConsole/saml/login
SAML Initiator	OneLogin
nameID format	Email
issuer type	Specific
signature element	Response
encryption method	TRIPLEDES-CBC

5. On the **Parameters** tab, set the application parameters as shown in this table.

Parameter	Value
NameID	Email name part
email	Email
firstname	First Name
group	User Roles
lastname	Last Name

Also, select the option **Include in SAML assertion**.

6. The **SSO** tab contains the issuer URL required to configure the Kofax RPA `saml.xml` file.

   Make sure that **X.509 Certificate** is set to **Standard Strength Certificate (2048-bit)** and **SAML Signature Algorithm** is set to **SHA-1**.

   Copy the **Issuer URL** property value and use it in the "IDP Metadata configuration" section in your `saml.xml` file. For an example, see below.

7. On the menu, click **USERS** > **Roles** and select the option to add a new role.

   Add roles that correspond to your Management Console groups and assign to them the application created before. The **KapowAdmins** role is mandatory in Kofax RPA, so ensure that you add it.

8. On the menu, click **USERS** > **All users** and select the required roles for the user.

   You have now configured a OneLogin application.

9. Now you need to configure group attributes in `saml.xml` located in the folder `\WEB-INF\spring` to match the OneLogin application you have created.

   After changing the file, restart Tomcat.

   Example

```
        <property name="groupsAttributes">
            <list>
                <value>group</value>
            </list>
        </property>
        <property name="adminGroups">
            <list>
                <value>KapowAdmins</value>
            </list>
        </property>
...

<property name="idpName" value="ONELOGIN"/>

...

    <bean id="metadataGeneratorFilter"
 class="org.springframework.security.saml.metadata.MetadataGeneratorFilter" lazy-
init="true">
        <constructor-arg>
            <bean
 class="org.springframework.security.saml.metadata.MetadataGenerator">
                <property name="entityId" value="http://<IP_address>:8080/
ManagementConsole/saml/login"/>
                <property name="requestSigned" value="false"/>
                <property name="entityBaseURL" value="http://<IP_address>:8080/
ManagementConsole"/>

                <property name="extendedMetadata">
                    <bean
 class="org.springframework.security.saml.metadata.ExtendedMetadata">
                        <property name="idpDiscoveryEnabled" value="false"/>
                        <property name="signMetadata" value="false"/>
                    </bean>
                </property>
            </bean>
        </constructor-arg>
    </bean>

...

    <!-- IDP Metadata configuration - paths to metadata of IDPs in circle of trust
 is provided here
    -->
    <bean
 class="org.springframework.security.saml.metadata.CachingMetadataManager"
 id="metadata" lazy-init="true">
        <constructor-arg>
            <list>

                 <!--  <bean
 class="org.opensaml.saml2.metadata.provider.FilesystemMetadataProvider">
                      <constructor-arg>
                          <value type="java.io.File">classpath:security/idp.xml</
value>
                      </constructor-arg>
                      <property name="parserPool" ref="parserPool"/>
                  </bean> -->
                <bean
 class="org.opensaml.saml2.metadata.provider.HTTPMetadataProvider" lazy-
init="true">
                        <constructor-arg>
                            <value type="java.lang.String">https://
app.onelogin.com/saml/metadata/d237b5c5-b110-4f42-a646-7678ae08feae</value>
                        </constructor-arg>
                        <constructor-arg>
                            <value type="int">10000</value>
                        </constructor-arg>
                        <property name="parserPool" ref="parserPool"/>
                    </bean>
                 <!--  <bean
 class="org.opensaml.saml2.metadata.provider.FilesystemMetadataProvider">
                      <constructor-arg>
                          <value type="java.io.File">classpath:security/idp.xml</
value>
                      </constructor-arg>
                      <property name="parserPool" ref="parserPool"/>
                  </bean> -->
            </list>
         </constructor-arg>
    </bean>
```

High Availability

If high availability (failover) is required, you can configure multiple Management Console instances to work together as a cluster. The following components must be clustered to achieve full failover.

Cluster Components

Component	Description
Load balancer	An HTTP load balancer is required to distribute requests between multiple Tomcat servers. ❗ When configuring high availability mode, all other services, such as RoboServers and Kapplets must be configured to access Management Console via the load balancer instead of the direct connection to Management Console.
Clustered platform database	The Management Console stores schedules, robots, and other in the platform database. In a failover scenario, the platform database should run on a clustered DBMS to avoid a single point of failure.
Tomcat session replication	Although the Management Console does not store any data directly in the user's session (except during Import/Export), the session holds the user's authentication information. If session replication is not enabled, the user will have to login again if the Tomcat he is currently connected to crashes.
Apache Tomcat Connector	mod_jk is an Apache module used to connect the Tomcat servlet container with web servers such as Apache, iPlanet, Sun ONE (formerly Netscape) and even IIS using the Apache JServ Protocol.
Hazelcast	Hazelcast (www.hazelcast.com) is used to cluster data structures over multiple JVMs. Inside the Management Console this is used to provide clustering of vital data structures, and to provide intercommunication between application instances. Here is a example: When you run a robot on a RoboServer, a thread is required to process the status messages returned by the RoboServer. This thread runs inside a specific Tomcat instance. In a clustered environment, a user trying to stop the robot may in fact be generating the stop request on another Tomcat instance than the instance running the robot. In that case the stop request is broadcast through Hazelcast to all instances and the instance running the robot will receive it and act to stop the robot.

Multiple Management Console Instances

You should have two or more identical Tomcat installations, and deploy the same version of `ManagementConsole.war` on them all. Make sure the `web.xml`, `Configuration.xml`, `login.xml`, and `roles.xml` files are the same across all the instances.

Install and Configure Components

This topic describes how to install and configure necessary components for high availability configuration using multicast clustering. In this configuration we will set up two host computers: one host (host1) contains Tomcat server and a database, another host (host2) contains Tomcat server and Apache server as a load balancer.

Step-by-Step Procedure

The following procedure helps you to install components for high availability configuration.

1. Set up a database on "host1" computer.
2. Download Tomcat from the Apache website: https://tomcat.apache.org
3. Install Tomcat on both hosts and set user password. See Tomcat Deployment for more information.
4. Install Management Console on Tomcat on both hosts.
5. Start Tomcat application on both computers and make sure they go online. You should have two identical Tomcat installations, and deploy the same version of ManagementConsole.war on them all. Make sure the `web.xml`, `Configuration.xml`, `login.xml`, and `roles.xml` files are the same on both installations.
6. Shut down Tomcat servers.
7. Download Apache server from the Apache website: http://httpd.apache.org/download.cgi#apache24. Install the Apache server on "host2" and start the Apache service. See Apache doc for details: http://httpd.apache.org/docs/
8. Download Apache `mod_jk` connector from the Apache website: https://tomcat.apache.org/download-connectors.cgi.
   - Unzip the files to a directory on your disk.
   - Copy `mod_jk.so` file to the `<apache>\module` directory.
   - Edit `<apache>\conf\httpd.conf` as follows:

     ```
     LoadModule jk_module modules/mod_jk.so
     <IfModule mod_jk.c>
      JkWorkersFile "<apache>\conf\workers.properties"
      JkLogFile "<apache>\logs\mod_jk.log"
      JkLogLevel error
      JkLogStampFormat "[%a %b @d %H:%M:%S %Y] "
      JkRequestLogFormat "%w %V %T"
     </IfModule>
     JkMount /ManagementConsole/* loadbalancer
     JkMount /ManagementConsole loadbalancer
     ```
   - Create a `<apache>\conf\workers.properties` file with the following content:

     ```
     worker.list=host1, host2, loadbalancer

     worker.host1.host=<ip address or host name>
     worker.host1.port=8009
     worker.host1.type=ajp13
     worker.host1.lbfactor=1
     worker.host2.host=<ip address or host name>
     worker.host2.port=8009
     worker.host2.type=ajp13
     worker.host2.lbfactor=1
     worker.loadbalancer.type=lb
     worker.loadbalancer.balance_workers=host1, host2
     ```

     Where `<ip address or host name>` is the address or the host name of the host computers running Tomcat.
9. For both Tomcat servers enter the following lines to the `conf\server.xml` file.

   ```
   <Connector protocol="AJP/1.3"
   address=<host name or IP address>
   ```

```
port="8009"
redirectPort="8443"
secretRequired="false" />
<Engine name="Catalina" defaultHost="localhost" jvmRoute="<host number from
 workers.properties>">
<Cluster className="org.apache.catalina.ha.tcp.SimpleTcpCluster"/>
```

For more information, see Tomcat Session Replication.

10. On each Tomcat server, edit `webapps\ManagementConsole\WEB-INF\Configuration.xml` file as follows. Note that you must specify valid IP addresses of the host computers on your network.

```
<!-- Cluster configuration -->
<bean id="cluster" class="com.kapowtech.mc.config.ClusterConfig" >
<property name="port" value="5701"/>
<!-- In "<mask for the IP address>"/>, insert the values, such as "192.168.1.*" or
 "10.10.10.*"-->
<property name="interface" value="<mask for the IP address>"/>
<!-- Uncomment the line below to enable clustering via multicast. Your license
must support High Availability for this to work -->
<!--property name="joinConfig" ref="multicastCluster"/> -->
<!-- or uncomment this line to enable clustering via TCP-IP. Your license must
support High Availability for this to work-->
<property name="joinConfig" ref="tcpCluster"/>
<property name="managementCenterUrl" value=""/>
</bean>
 <!-- definition for a TCP cluster. You need to add peers to this list, so each
 client can locate at least one other functioning cluster member -->
 <bean id="tcpCluster" class="com.kapowtech.mc.config.TcpJoinConfig" lazy-
init="true">
 <property name="peers">
 <list>
 <bean class="com.kapowtech.mc.config.TcpPeer">
 <property name="host" value="<ip address or host name>"/>
 <!-- port is only needed if the other machine is not using the same port as this
 instance-->
 <!--property name="port" value="5701"/-->
 </bean>
 <bean class="com.kapowtech.mc.config.TcpPeer">
 <property name="host" value="<ip address or host name>"/>
 <!-- port is only needed if the other machine is not using the same port as this
 instance-->
 <!--property name="port" value="5701"/-->
 </bean>
 </list>
 </property>
 </bean>
```

For more information, see Hazelcast Replication.

Now you can log in to the Management Console on the load balancer by navigating to `<host2>:80/ManagementConsole`, where "host2" is the name of the computer running Apache server. Once you log in, go to **Admin** > **High availability nodes**, and you should see two nodes with correct IP addresses.

ⓘ When configuring high availability mode, all other services, such as RoboServer and Kapplets must be configured to access Management Console via the load balancer instead of the direct connection to Management Console.

Load Balancer Startup

This section describes how to determine if the application started correctly.

If the `ManagementConsole.xml` (context configuration) or `web.xml` files are invalid, the application cannot be deployed on Tomcat, and requests normally return error code 404 (as it hits the Tomcat's ROOT application which does not have anything deployed on `/ManagementConsole/`).

Any other errors encountered during application startup are shown to the user when the application loads. This way you do not always have to check the log to figure out why the application did not load correctly. This is, however, a bit impractical as the application returns 200 OK even if there are errors during startup. Also, if authentication is enabled, you have to login before you can see the error messages.

To make it easier for load balancers to see if the application started correctly, you can make a request to the URL `/ManagementConsole/Ping`. This either returns HTTP status code 200 if the application loaded correctly, or 500 with a stack trace of the error.

Tomcat Session Replication

Session replication is configured in `/conf/server.xml`. Here is an example that uses multicast for instance discovery on Tomcat.

```
<Cluster className="org.apache.catalina.cluster.tcp.SimpleTcpCluster"
          managerClassName="org.apache.catalina.cluster.session.DeltaManager"
          expireSessionsOnShutdown="false"
          useDirtyFlag="true"
          notifyListenersOnReplication="true"
          printToScreen="true">

         <Membership
             className="org.apache.catalina.cluster.mcast.McastService"
             mcastAddr="228.0.0.4"
             mcastPort="45564"
             mcastFrequency="500"
             mcastDropTime="3000"/>

         <Receiver
             className="org.apache.catalina.cluster.tcp.ReplicationListener"
             tcpListenAddress="auto"
             tcpListenPort="4002"
             tcpSelectorTimeout="100"
             tcpThreadCount="6"/>

         <Sender
             className="org.apache.catalina.cluster.tcp.ReplicationTransmitter"
             replicationMode="pooled"
             ackTimeout="150000"
             waitForAck="true"/>

         <Valve className="org.apache.catalina.cluster.tcp.ReplicationValve"
                filter=".*\.gif;.*\.js;.*\.jpg;.*\.png;.*\.htm;.*\.html;.*\.css;.*
\.txt;"/>

         <Deployer className="org.apache.catalina.cluster.deploy.FarmWarDeployer"
                   tempDir="/tmp/war-temp/"
                   deployDir="/tmp/war-deploy/"
                   watchDir="/tmp/war-listen/"
```

```
                  watchEnabled="false"/>

        <ClusterListener
className="org.apache.catalina.cluster.session.ClusterSessionListener"/>
    </Cluster>
```

You also have to set the jvmRoute attribute on the <Engine> element in `server.xml`:

```
<Engine jvmRoute="tomcat2" name="Catalina" defaultHost="MyHost">
```

If you are using mod_jk as a poor man's load balancer, the value of the jvmRoute has to match the name listed in the `workers.properties` file references by the mod_jk configuration.

See your Tomcat documentation for details.

Hazelcast Replication

The most basic Hazelcast settings can be edited in `Configuration.xml`, while more advanced settings such as SSL encryption must be configured in `/WEB-INF/Hazelcast.xml`

When a Management Console starts, it creates a Hazelcast node on port 5701 (or the next available port if 5701 is unavailable). By default this Hazelcast node binds to IP address 127.0.0.1. You have to change the bind address to a public IP/host name before it can participate in a cluster. This is done by modifying the interface property of the cluster bean in `Configuration.xml`. It might look like this:

```
    <bean id="cluster" class="com.kapowtech.mc.config.ClusterConfig" >
        <property name="port" value="26000"/>
        <property name="interface" value="10.0.0.*"/>
    .......
    </bean>
```

The * is used as a wildcard, in this case the application will try bind to the 'first' interface that has an IP address starting with 10.0.0. It is possible, but not recommended to use *.*.*.* as you may end up binding to 127.0.0.1, or another virtual interface.

When you start additional instances of Management Console, their Hazelcast instances will try to find any existing Hazelcast node and join the cluster. This discovery can be done through multicast or through TCP/IP.

To use multicast discovery you must modify the cluster bean in `Configuration.xml`. This is done my un-commenting the following line:

```
<property name="joinConfig" ref="multicastCluster"/>
```

multicastCluster is a reference to the multicastCluster bean, which defines the multicast group and port. You may change it to fit your network topology.

If your network does not allow multicast you will have to use the tcpCluster. That is done by un-commenting this line instead:

```
<property name="joinConfig" ref="tcpCluster"/>
```

The tcpCluster bean contains a list of TcpPeer, one for each other Hazelcast node. If you use the same TCP port for all Hazelcast nodes you do not need to specify a port number (each node will assume that its peers are running on the same port as itself). If you have two nodes configured in a TCP cluster it could look like this:

```
    <bean id="tcpCluster" class="com.kapowtech.mc.config.TcpJoinConfig">
         <property name="peers">
             <list>
                 <bean class="com.kapowtech.mc.config.TcpPeer">
                     <property name="host" value="10.0.0.25"/>
                 </bean>
                 <bean class="com.kapowtech.mc.config.TcpPeer">
                     <property name="host" value="10.0.0.26"/>
                 </bean>
             </list>
         </property>
     </bean>
```

Notice that both nodes are in the list. This means that regardless which node starts first it will be able to find its peer. It also allows you to use identical `Configuration.xml` files in both applications. Also, TCP ports numbers are not defined, so each peer will try to connect to the other one on the same port as it is listening on itself.

Application Nodes

You can verify that the application is properly clustered by going to **Admin** > **High Availability Nodes**.

The **Interface** column lists the IP/host and port that Hazelcast is using for inter-cluster communication. The **Connected to** column shows which of the two nodes you are connected to at the moment. If you shut down the server you are currently connected to, you will automatically be re-routed to another live instance by the load balancer.

From the context menu for a node, you can request a thread dump, which may be useful for debugging purposes.

URI Encoding

If you plan to upload robots with names that contain non-ASCII characters, like Danish ÆØÅ or German ß to the repository, you have to configure the URI Encoding on your web container to UTF-8.

On Tomcat this is done on the <connector> definition found in `server.xml` file inside the `/conf` folder. Here you add the attribute URIEncoding="UTF-8" like this:

```
<Connector port="8080" URIEncoding="UTF-8"....../>
```

Password Encryption

Management Console uses certificate based (public-, private-key) encryption when storing passwords. When you import from a previous version password will automatically be re-encrypted using the new certificate based algorithm.

The certificate and the matching private key is stored in a Java keystore, Management Console ships with a keystore that contains a default certificate and private key. As all customers get the same keystore we recommend that you create your own keystore, otherwise anyone will be able to load your exports and potentially get your passwords.

Create Your Own Keystore

If you have already started a Management Console, you need to upgrade the certificate. The keystore must be in pkcs12 format, and can be created using the keytool application that comes with the Java SDK (which can be downloaded from Oracle.com). The following command creates a new pkcs12 keystore with a certificate that is valid for 365 days.

```
keytool -genkey -alias mc -keyalg RSA -validity 3650 -keystore mc.p12 -
storetype pkcs12
```

You will be prompted for password, and the information that will be stored in the X.509 private key. The command will create a file mc.p12 (the value from the -keystore argument) in the current directory. -validity 3650 means the certificate will be valid for 10 years.

We do not recommend using a certificate issued by a certificate authority (CA) as pkcs12 holds both the private key and the public certificate, and the password to the private key will be written in clear text as part of the application configuration.

To instruct Management Console to use the new certificate, change the `Configuration.xml` file. The file is located inside the `ManagementConsole.war` web archive, which must be unpacked, see Deploying into Tomcat for details. Inside `Configuration.xml` you will find the following entry:

```
        <bean id="keyStore" class="com.kapowtech.mc.config.KeyStoreConfig" >
              <property name="location" value="/WEB-INF/mc.p12"/>
              <property name="password" value="changeit"/>
              <property name="alias" value="mc"/>
        </bean>
```

Here you must specify the location, password and alias of the keystore. If you copy the keystore into `ManagementConsole.war` the location must be relative to the root of the application. If you want to refer to a keystore stored in the file system, the location must start with file://, and must be an absolute reference to the keystore location.

Upgrading the Keystore

The first time Management Console starts, it creates a checksum using the private key from the keystore, this allows it to detect when the keystore has been replaced, and verify that passwords can in fact be decrypted with the provided certificate. If you have already started a Management Console before installing your own keystore, you have to configure it to perform a password conversion.

To upgrade the keystore, copy the current keystore file into a new location, such as your users home folder, then modify `Configuration.xml` to create a password converter with a reference to the old keystore:

```
    <bean id="oldKeyStore" class="com.kapowtech.mc.config.KeyStoreConfig" >
       <property name="location" value="file:///home/roboserver/mc.p12"/>
       <property name="password" value="changeit"/>
```

```
         <property name="alias" value="mc"/>
      </bean>

      <bean id="passwordConverter"
 class="com.kapowtech.scheduler.server.service.PasswordConverter">
         <constructor-arg ref="oldKeyStore"/>
      </bean>
```

This configures a password converter to use the previous certificate to decrypt any existing passwords and checksum (you will have to provide correct location, alias and password for the old keystore), and use the new private key (as configured above) to re-encrypt passwords and create a new checksum. The conversion occurs the next time the Management Console is started, the conversion occurs while the application is starting and may take some time if there are many schedules. You do not have to remove the oldKeyStore and passwordConverter beans from `Configuration.xml`, as the password conversion is only triggered when the checksum and keystore is out-of-sync, and after the conversion the checksum matches the new keystore).

SSL Endpoint Verification

When you create a new Cluster you can select that you want the communication with the RoboServers to be SSL encrypted, this prevents anyone from "listening" to the network and extracting critical information exchanged between the two parties.

In addition to encryption, SSL also offer endpoint validation. This is to ensure that you do not exchange critical information with a third party, either due to misconfiguration, or because you DNS has been hacked. For this to work you need to configure RoboServer to trust your Management Console and configure Management Console to trust your RoboServers.

This requires you to edit files inside `ManagementConsole.war`, so make sure you Tomcat server is not running when you perform this modification.

Certificates

You will need to create two certificates, one for Management Console and one for RoboServer, each certificate contains a private and a public key. Creating a certificate and exporting the public key is described here, in general it is a good idea to read the entire section of the help the discusses certificates, especially the section on API Client/Server certificates.

Endpoint verification can be separated into two parts, making RoboServer trust Management Console and making Management Console trust RoboServer, each of these are configured individually, and you do not have to configure both.

Make RoboServer Trust Management Console

You now have to configure a Management Console to use the private key when creating the SSL connection to a RoboServer. This is done by modifying `/WEB-INF/certs.xml` found inside the WAR file. Provide the location, and the password for the certificate, which could look like this:

```
<bean id="sslCertificationConfiguration"
 class="com.kapowtech.mc.config.SSLVerificationConfiguration">
                   ...
     <property name="privateCertificateLocation" value="file:///home/roboserver/
client.p12"/>
     <property name="privateCertPassword" value="changeit"/>
```

```
</bean>
```

Management Console is now using a private key when establishing SSL connections. Once the Management Console public key is deployed in the RoboServer `/TrustedClients` folder, the RoboServer can verify that it is connected to the right Management Console. Remember to enable **Verify API Client Certificates** in RoboServer Settings, and deploy the public key on all RoboServers in the cluster.

Make Management Console Trust RoboServer

RoboServer already comes with an API certificate installed, therefore you have to create a new certificate and replace the pre-installed one. First create the certificate as described above, then start RoboServer Settings and go to the **Certificates** tab. Click the change button, select the certificate, and enter the password when prompted. RoboServer now uses the new certificate when creating SSL connection with a Management Console (and other API clients).

Now you need to configure the Management Console to only trust SSL connections from RoboServers with the correct certificate. Like the Management Console client certificate this is (partly) configured in `/WEB-INF/certs.xml`, using the following two options:

```
<bean id="sslCertificationConfiguration"
class="com.kapowtech.mc.config.SSLVerificationConfiguration">
    <property name="verifyRoboServerCert" value="true"/>
    <property name="checkHostName" value="true"/>
    ...
</bean>
```

The option for verifying RoboServer certificates is a simple boolean flag (true/false), this is because you have to import the RoboServer public key into the JRE's default keystore. The JRE's default keystore is a file named `cacerts` located at `/jre/lib/security/`.

To import the RoboServer public key into `cacerts`, use the following command:

```
keytool -import -alias RoboServer -keystore cacerts -trustcacerts -file
server.pub.cer
```

You will be prompted for a password, which is `changeit` unless you have changed it. The alias has to be unique, so if you created a separate certificate for each RoboServer, add a suffix. Also note that the references to cacerts and server.pub.cer are relative in this example.

The checkHostName option ensures that Management Console only communicates with a RoboServer if it presents the correct certificate and is contacted using the hostname written inside the RoboServer certificate. Note that localhost and 127.0.0.1 is not considered the same host when the hostname is checked.

Troubleshooting

Troubleshooting can be quite hard as there is virtually no information available if SSL connections cannot be established, but it is important to know the following.

- Management Console does not start if it cannot find the certificate, or if the password is wrong.
- When you change the RoboServer certificate in RoboServer Settings, it checks that the password is correct before storing the certificate.

If a Management Console cannot connect to a RoboServer, the following may help you troubleshoot:

- Is RoboServer running? Try to telnet to the socket to be sure.
- Is the RoboServer host name correct (if checkHostName is enabled)?
- Is the v public key imported into cacerts? Use keytool -list -v -keystore cacerts -alias RoboServer if you give -alias it lists all certificates.
- Was the Management Console public certificate copied to the RoboServer `/TrustedClients` folder?
- Check expiration date. The public key contains the expiration data of the private key, and can be opened/viewed in both Windows and Linux.

Simultaneous Sessions for a User Account

By default, the system allows a single user account to be authenticated simultaneously from multiple locations. To restrict the possibility of concurrent sessions for a single user account, adjust the settings in the `authentication.xml` file that resides in `WEB-INF/spring`.

In `authentication.xml`, locate the following section and remove the comment tags (marked in bold here):

```
<!--
<bean class="com.kapowtech.scheduler.server.spring.security.KapowConcurrentSes
sionControlAuthenticationStrategy" lazy-init="true"> <constructor-arg ref="ses
sionRegistry"/> <constructor-arg ref="platformEMF"/> <property name="maximumSessi
ons" value="1"/> <property name="exceptionIfMaximumExceeded" value="true"/> </bea
n>
-->
```

Also, to define the timeout to automatically end the session if the user does not perform any actions, configure the `session-timeout` property in the `web.xml` file that resides in `WEB-INF`. Be default, the timeout is 30 minutes.

Use Microsoft SQL Server with integrated security

If you want to run Kofax RPA Management Console with Microsoft SQL Server database that uses integrated security, as well as store data in such database, perform the following steps to set up the environment. The JDBC driver cannot be stored in the Management Console, therefore both JAR and DLL files must be placed in the specified folders.

On Tomcat server

- Copy the JAR file of the Microsoft JDBC Driver for SQL Server to the **lib** folder of the Tomcat installation folder.
- Copy the DLL file of the Microsoft JDBC Driver for SQL Server to the **bin** folder of the Tomcat installation folder.

On developer computers for Design Studio users

- Copy the JAR file of the Microsoft JDBC Driver for SQL Server to the **lib** folder of the Design Studio installation folder.

- Copy the DLL file of the Microsoft JDBC Driver for SQL Server to the **jre\bin** folder of the Design Studio installation folder.

On RoboServer computers

- Copy the JAR file of the Microsoft JDBC Driver for SQL Server to the **lib** folder of the RoboServer installation folder.
- Copy the DLL file of the Microsoft JDBC Driver for SQL Server to the **jre\bin** folder of the RoboServer installation folder.

> ⓘ Users running Design Studio and RoboServers must have access rights to the database and must run on Windows.

Set up Robot File System server

Robot File System (RFS) server provides shared storage for RoboServers, Design Studio instances, and Desktop Automation agents. To set up an RFS server on a Tomcat server, perform the following steps.

The maximum size of the file that can be uploaded to or downloaded from the Robot File System is limited by available memory.

1. Locate the `rfs.war` file in the `WebApps` folder in your Kofax RPA installation folder.

   For example, on a Windows system, the folder resides in: `C:\Program Files\Kofax RPA 11.5.0.0\WebApps`
2. Copy `rfs.war` to the `webapps` folder in your Apache Tomcat installation folder.
   For example, on a Windows system, the folder resides in: `C:\apache-tomcat\webapps`
3. Restart the Tomcat server.

   After you restart the Tomcat server, the `webapps` folder in the Tomcat installation folder should contain the `rfs` subfolder.
4. In a text editor, open the `web.xml` file located in the `webapps\rfs\WEB-INF` folder in the Tomcat installation folder.
5. Locate the `mc-path` parameter and specify the Management Console URL in `param-value`. For example, `http://localhost:50080`.

   > ⓘ Accessing the Robot File System over HTTPS with a self-signed certificate is not supported.
6. Open the Management Console and go to **Admin** > **Service authentication**. Click the context menu for **RFS**, select **Show shared secret**, and copy the shared secret to clipboard.

   Store the shared secret in a file, locate the `shared-secret-file` parameter and specify the path to this file in `param-value`. Another option is to paste the plain text shared secret directly in the `shared-secret` parameter (only used when `shared-secret-file` is empty or the file cannot be read).
7. Locate and set the `allow-absolute-paths` parameter to `true` or `false`.

   If `allow-absolute-paths` is set to `true`, you can create RFS file shares with paths such as `c:\files`, `z:\data`, and other paths that the RFS service user can access. If `allow-absolute-`

`paths` is set to `false` and `data-path` is set to a specific folder, such as `/data` on Linux, the service only allows access to shares with a path within the specified folder, such as `/data`.

8. Specify a folder to store temporary robot run data in `data-path`. For example, `C:/RFSData`. Note that you can specify the absolute path only if `allow-absolute-paths` is set to `true`.

   Temporary shares are created and deleted as subfolders of the specified folder.
9. Leave other settings as they are and restart the Tomcat server.
10. Open the Management Console and go to **Settings** > **General** > **Robot File System server**.

    Select **Use Robot File System server** and specify the URL to the Tomcat server where you set up the RFS server. For example, `http://myserver.mydomain:8080/rfs`.

Now you can use file systems configured to share and store data used and/or produced by robots. To add a configuration for a file system, see "Robot File System" in *Kofax RPA Help*.

Example: Map folder to Robot File System

This example provides general steps on how to map a folder on Windows to a Robot File System in Management Console and add a step to a robot in Design Studio to write data to this Robot File System.

Before following this example, we recommend that you read the procedure "Set up Robot File System server" above and "Robot File System" in *Kofax RPA Help* as they provide detailed information on configuration and usage of the Robot File System functionality.

This procedure is written with the assumption that you have completed steps 1-8 from "Set up Robot File System server."

1. In the `web.xml` file, in the `data-path` parameter, specify the path to a folder to store temporary robot run data. For example, `c:/rfs`.

   ```
   <init-param>
               <!-- the path to where local data is stored -->
               <param-name>data-path</param-name>
               <param-value>c:/rfs</param-value>
        </init-param>
   ```

   Restart the Tomcat server to apply the settings.
2. In **Management Console** > **Settings** > **General** > **Robot File System server**, select to use the Robot File System server.
3. In **Management Console** > **Repository** > **Robot File System**, add a configuration for your file system.

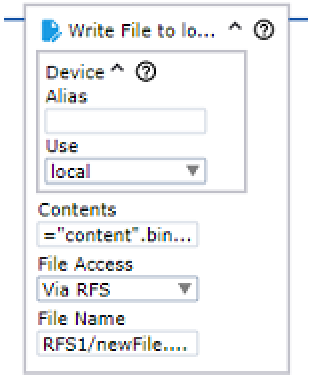
   a. On the **General information** tab, specify all required parameters.

      In the **File system name** parameter, specify **RFS1**. In the **Path** parameter, specify: **rfs1_folder**.

      In this case, `rfs1_folder` is the root folder for **RFS1**, so the absolute path would be `c:/rfs/rfs1_folder`. The `rfs1_folder` will be created automatically if not created manually.

   b. On the **Authorized access tokens** tab, paste the robot access token to ensure that only the current version of the robot has access to the file system (the robot must be uploaded to Management Console). To copy the token, go to **Management Console** > **Repository** > **Robots**, open the context menu for the required robot, and click **Get resource access token**.

4. Save the configuration.
5. In Design Studio, in a Robot, create a Write File step with the following properties.

   Write File to lo...

   Device

   Alias

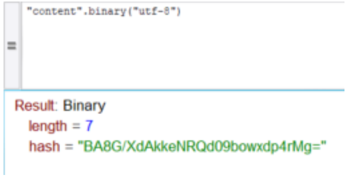
   Use: local

   Contents: ="content".bin...

   File Access: Via RFS

   File Name: RFS1/newFile....

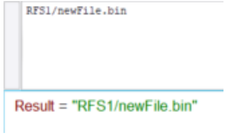
   Contents: Specify a binary variable containing data to write to the Robot File System. In this example, "content" is the string written in the file. The file content must be binary.

   ```
   "content".binary("utf-8")
   ```

   ```
   Result: Binary
     length = 7
     hash = "BA8G/XdAkkeNRQd09bowxdp4rMg="
   ```

   File Name: Specify the path to the file to which the data is written. The Robot File System name is case-sensitive.

   ```
   RFS1/newFile.bin
   ```

   ```
   Result = "RFS1/newFile.bin"
   ```

6. After you execute the step, the newFile.bin file will contain the data from "content". The file will be saved to `C:/rfs/rfs1_folder/`.

Configure temporary RFS session storage

During the execution process, robot can save and read files on the Robot File System. For this purpose, robot creates temporary RFS session storage, which is used only by the robot.

For example, temporary RFS session storage might be useful for robots to save files with similar names and different content. This option is available in several steps, such as Read File and Write File.

To map a folder to the temporary RFS session storage, after setting up RFS server, configure the required robot step. In the **File Name** property, specify the path, such as `robot/newFile.bin`.

The file will be saved to a folder, such as `C:/rfs/8c8a89ca-fd06-4580-99a7-b7ffb7288080`.

This folder is unique to each robot execution and is deleted after the execution is complete.

Chapter 3

Run RPA Components As Services

This chapter describes how to run different Kofax RPA components as services using the `ServiceInstaller.exe` program.

ServiceInstaller.exe explained

To run a Kofax RPA component as a service, you need to install it first using the `ServiceInstaller.exe` program. The following is a general example outlining the command-line arguments to the "RPAComponent" program (although displayed on multiple lines here, this is a one-line command):

```
ServiceInstaller.exe -i RPAComponent.conf wrapper.ntservice.account=Account
 wrapper.ntservice.password.prompt=true wrapper.ntservice.name=Service-
name wrapper.ntservice.starttype=Start-method wrapper.syslog.loglevel=INFO
 wrapper.app.parameter.1="First-Argument" wrapper.app.parameter.2="Second-argument"
```

wrapper.ntservice.account

The account of the user that has to run an "RPAComponent". Kofax RPA stores configuration in the user's directory and it is important to choose a user that has the correct configuration.

To run "RPAComponent" as a domain user, enter the account in the form `domain\account`

To run "RPAComponent" as a regular user, enter the account in the form `.\account`

> ⓘ For security reasons, do not use the `LocalSystem` account for the RoboServer service's login. If `LocalSystem` is used, the following error occurs when WebKit (default) robots run: "Could not establish connection to WebKitBrowser. Failed to connect to bus."

wrapper.ntservice.password.prompt

The value `true` prompts the user for the account password. If you prefer to enter the password in the command line, use `wrapper.ntservice.password=<your-password>`.

wrapper.ntservice.name

The name of the service to install. Note that the name of the service can not contain spaces.

wrapper.ntservice.starttype

Specify the following values.

- **AUTO_START**: Starts the service automatically when the system is restarted.
- **DELAY_START**: Starts the service after a short delay.
- **DEMAND_START**: Starts the service manually.

wrapper.syslog.loglevel

Redirect the console output from "RPAComponent" to the event log.

wrapper.app.parameter.

The arguments for "RPAComponent". You can enter as few or as many as needed.

When the service is installed, the user is granted the "log on as a service" rights. If the service fails to start, check that the right is granted by opening gpedit.msc and (on Windows 10) navigate to **Administrative Tools** > **Local Security Policy** > **Local Policy** > **User Rights Assignment** > **Log on as a service** > **Properties** and add the user.

Run RoboServer and Management Console as a service

Both RoboServer and Management Console are started by the same RoboServer server program, depending on the arguments supplied to it when it starts.

See the RoboServer Parameters section in Start RoboServer for a detailed description of the command-line arguments for the RoboServer program.

The following are examples on how to start a RoboServer and Management Console automatically on Windows and Linux.

Start RoboServer on Windows

To make a RoboServer start automatically on Windows, add it as a Windows service. This topic explains how to add and remove Windows services using the `ServiceInstaller.exe` program that is included in the Kofax RPA installation.

Add Windows Services

The following are examples of installing RoboServers in different configurations. In the examples MC means Management Console and RS means RoboServer.

- The following script installs services that start RoboServers with default parameters. The name of the service can be changed as needed.

```
ServiceInstaller.exe -i RoboServer.conf
wrapper.ntservice.account="<DOMAIN>\<USERNAME>"
wrapper.ntservice.password.prompt=true
wrapper.ntservice.name="RoboServer11.5.0_MC"
wrapper.ntservice.starttype=AUTO_START wrapper.syslog.loglevel=INFO
wrapper.app.parameter.1="-p" wrapper.app.parameter.2="<PORT-NUMBER>"
wrapper.app.parameter.3="-mcUrl" wrapper.app.parameter.4="<MC-URL>"
wrapper.app.parameter.5="-cl" wrapper.app.parameter.6="<ROBOSERVER-
CLUSTER>" wrapper.app.parameter.7="-ss" wrapper.app.parameter.8="<MC-
SHARED-SECRET>"
```

- This script creates a Windows Service that only starts the Management Console. This is the recommended configuration as the Management Console should run under its own JVM if possible. The name of the Windows Service can be changed as needed.

```
ServiceInstaller.exe -i RoboServer.conf
wrapper.ntservice.account="<DOMAIN>\<USERNAME>"
wrapper.ntservice.password.prompt=true
wrapper.ntservice.name="RoboServer11.5.0_MC"
wrapper.ntservice.starttype=AUTO_START wrapper.syslog.loglevel=INFO
wrapper.app.parameter.1="-p" wrapper.app.parameter.2="<PORT-
NUMBER>" wrapper.app.parameter.3="-MC" wrapper.app.parameter.4="-
pauseAfterStartupError" wrapper.app.parameter.5="-mcUrl"
wrapper.app.parameter.6="<MC-URL>" wrapper.app.parameter.7="-ss"
wrapper.app.parameter.8="<MC-SHARED-SECRET>"
```

- The following scripts install services that start two RoboServers: one on port 50000 and the other on 50001. The service name can be different.

  ```
  ServiceInstaller.exe -i RoboServer.conf
  wrapper.ntservice.account="<DOMAIN>\<USERNAME>"
  wrapper.ntservice.password.prompt=true
  wrapper.ntservice.name="RoboServer11.5.0_50000"
  wrapper.ntservice.starttype=AUTO_START wrapper.syslog.loglevel=INFO
  wrapper.app.parameter.1="-p" wrapper.app.parameter.2="50000"
  wrapper.app.parameter.3="-mcUrl" wrapper.app.parameter.4="<MC-URL>"
  wrapper.app.parameter.5="-cl" wrapper.app.parameter.6="<ROBOSERVER-
  CLUSTER>" wrapper.app.parameter.7="-ss" wrapper.app.parameter.8="<MC-
  SHARED-SECRET>"
  ```

  ```
  ServiceInstaller.exe -i RoboServer.conf
  wrapper.ntservice.account="<DOMAIN>\<USERNAME>"
  wrapper.ntservice.password.prompt=true
  wrapper.ntservice.name="RoboServer11.5.0_50001"
  wrapper.ntservice.starttype=AUTO_START wrapper.syslog.loglevel=INFO
  wrapper.app.parameter.1="-p" wrapper.app.parameter.2="50001"
  wrapper.app.parameter.3="-mcUrl" wrapper.app.parameter.4="<MC-URL>"
  wrapper.app.parameter.5="-cl" wrapper.app.parameter.6="<ROBOSERVER-
  CLUSTER>" wrapper.app.parameter.7="-ss" wrapper.app.parameter.8="<MC-
  SHARED-SECRET>"
  ```

Remove Windows Services

To uninstall a service you can run the following command:

```
ServiceInstaller.exe -r RoboServer.conf wrapper.ntservice.name=Service-name
```

wrapper.ntservice.name

The name of the service to remove.

Start Roboserver on Linux

The simplest way to make a RoboServer start automatically on Linux is to use crontab. Use the following command to create or edit the list of scheduled jobs in Linux for the particular user:

```
crontab -u someUser -e
```

To the list of scheduled jobs add for example:

```
@reboot $HOME/Kofax RPA_11.5.0/bin/RoboServer -mcUrl http://localhost:8080/
ManagementConsole -p 50000 -ss <MC Shared Secret>
```

This way the RoboServer program starts with the indicated command-line arguments upon reboot. Note that you must identify the bin directory under the actual installation folder.

Run Synchronizer as a service

Kofax RPA Synchronizer compares and synchronizes the state of objects between the Management Console and your repository. For more information about Synchronizer, see "Robot Lifecycle Management" in *Kofax RPA Help*.

This topic provides example of starting Synchronizer as a Windows service.

Add Windows Services

The following is an example of installing Synchronizer as a service. See ServiceInstaller.exe explained for information about `ServiceInstaller.exe`.

1. In the Command Prompt window, specify the required parameters to run Synchronizer. See "Start synchronization" in the Kofax RPA Help for details.

   > ℹ The command line needs to be run as the user that will run the Synchronizer service, because the configuration settings are saved under the AppData of the user running the command.

2. Verify that Synchronizer works properly and save the configuration settings using the `-s` parameter in the command line.
3. Install the Synchronizer service using the following script.

   ```
   ServiceInstaller.exe -i Synchronizer.conf wrapper.ntservice.account=domain
   \account wrapper.ntservice.password.prompt=true
   wrapper.ntservice.name="Synchronizer11.5.0_MC"
   wrapper.ntservice.starttype=MANUAL wrapper.syslog.loglevel=INFO
   ```

If you need to change a parameter, stop the Synchronizer service, run Synchronizer from the command line with the new parameters, save the new configuration settings, and restart the Synchronizer service.

Remove Windows Services

To uninstall a service you can run the following command:

```
ServiceInstaller.exe -r Synchronizer.conf
 wrapper.ntservice.name="Synchronizer11.5.0_MC"
```

wrapper.ntservice.name

The name of the service to remove.

Chapter 4

Audit Log for Management Console

Audit log for Management Console logs all user operations in the Management Console including API calls. By configuring the `log4j2.properties` file, you can log the information to a file or a database. On a Windows system, the `log4j2.properties` file is located at: `User home\AppData\Local\Kofax RPA\version\Configuration`.

Logging to File

```
#Log4j2 log to file configuration example

name = PropertiesConfig
appenders = auditLogAppender

appender.auditLogAppender.name = auditLog
appender.auditLogAppender.type = File
appender.auditLogAppender.fileName=logFilePath/logFileName.log
appender.auditLogAppender.layout.type = PatternLayout
appender.auditLogAppender.layout.pattern=%d - %m%n

logger.auditLog.name = auditLog
logger.auditLog.level = INFO
logger.auditLog.appenderRef.auditLog.ref = auditLog
logger.auditLog.additivity = false
```

Logging to Database

ⓘ The instructions below use MySQL database as an example, but other supported databases can be used for logging by using their specific JDBC drivers and URL connections.

To enable audit logging to MySQL database of the Management Console running with the embedded RoboServer, perform the following steps:

1. Copy the MySQL connector JAR file to the `lib` subfolder of the Kofax RPA installation folder (where the `Kapowtech-common.jar` and `platform.jar` are located). For example, `mysql-connector-java-<version>.jar`. Use the latest available version of the driver for your Java. For more information, see https://repo1.maven.org/maven2/mysql/mysql-connector-java/.
2. Create a database table where you want to log the data to. The following is a MySQL script for creating tables:

```
CREATE TABLE LOGS
    (
     DATED   timestamp      NOT NULL,
     LEVEL   VARCHAR(10)    NOT NULL,
     MESSAGE VARCHAR(1000)  NOT NULL
    );
```

> To prevent loss of information, make sure the message column has a minimum varchar size of 600.

3. Add the following lines to the `log4j2.properties` file:

```
#Log4j2 log to MySQL database configuration example

name = PropertiesConfig
appenders = auditLogAppender

appender.auditLogAppender.name = auditLogAppender
appender.auditLogAppender.type = JDBC
appender.auditLogAppender.connectionSource.type = DriverManager
appender.auditLogAppender.connectionSource.connectionString = jdbc:mysql://
localhost/YourDatabaseSchemaName
appender.auditLogAppender.connectionSource.username = user
appender.auditLogAppender.connectionSource.password = password
appender.auditLogAppender.connectionSource.driverClassName = com.mysql.jdbc.Driver

appender.auditLogAppender.tableName = LOGS

appender.auditLogAppender.columnConfigs[0].type = Column
appender.auditLogAppender.columnConfigs[0].name = DATED
appender.auditLogAppender.columnConfigs[0].pattern = %d{yyyy-MM-dd HH:mm:ss}

appender.auditLogAppender.columnConfigs[1].type = Column
appender.auditLogAppender.columnConfigs[1].name = LEVEL
appender.auditLogAppender.columnConfigs[1].pattern = %p

appender.auditLogAppender.columnConfigs[2].type = Column
appender.auditLogAppender.columnConfigs[2].name = MESSAGE
appender.auditLogAppender.columnConfigs[2].pattern = %msg

logger.auditLog.name = auditLog
logger.auditLog.level = INFO
logger.auditLog.appenderRef.auditLogAppender.ref = auditLogAppender
logger.auditLog.additivity = false
```

> Define the database schema name, user name, password and the table name that you created in the database.

> The timestamp format used in the example above is not universal. The correct processing of the query depends on the database type and the time format used in the database.
>
> Make sure timestamp format meets the database requirements.

To enable audit logging to MySQL database of the Management Console running with the Tomcat, perform the following steps:

1. Copy the MySQL connector JAR file to the `lib` directory under Apache Tomcat. For example, `mysql-connector-java-8.0.16.jar`. Use the latest available version of the driver for your Java. For more information, see https://repo1.maven.org/maven2/mysql/mysql-connector-java/.

2. Steps 2 and 3 are the same as for the Management Console running with the embedded RoboServer. The `log4j2.properties` file is located under `tomcat directory\webapps \Management Console\WEB-INF\classes`.

Audit Log Reference

This section provides a list of operations that are logged when they are executed successfully or fail due to access restrictions. Log file examples are also provided for your reference.

Login Events
RoboServers are using credentials to register to the Management Console, therefore all user logins are logged. When a RoboServer starts, there is a login event in the audit log from the user that was given access to the RoboServer.

Logged Operations

The logged operations are grouped by sections in the Management Console.

Example: Logs

Here are examples of what the robot execution logs look like in different scenarios. MySQL is used as the logging database and the log message consists of timestamp, logging level, and detail message.

Robot run from REST call

A robot that is started by REST call, first logs the robot name with ID, execution ID and task ID; and then the user who started the robot with the project name and task ID.

```
2019-11-27 16:42:26        INFO      Robot Wait60 with id = 17 execution id =
 -1-9-67f20877bebb task id = 9 has requested to start
2019-11-27 16:42:26        INFO      admin run Robot f1/f2/f3/Wait60.robot from Project
 Default project with task id = 9 from REST
```

Robot run from SOAP call

```
2019-11-27 16:34:24        INFO      Robot Wait60 with id = 17 execution id =
 -1-1-67f20877bebb task id = 1 has requested to start
2019-11-27 16:34:24        INFO      admin run Robot f1/f2/f3/Wait60 from Project
 Default project with task id = 1 from SOAP
```

Robot run started from UI

This robot was started from the Robots section in Management Console.

```
2019-11-27 16:35:56        INFO      Robot Wait60 with id = 17 execution id =
 -1-2-67f20877bebb task id = 2 has requested to start
2019-11-27 16:35:56        INFO      admin run Robot Wait60 with id = 17 task id = 2
```

Schedule triggered by time

No log.

Schedule triggered by user

Log messages can be referenced by schedule ID and task ID. The execution log messages with postfix "- from schedule, started by user" also indicates this robot was triggered by a manual schedule run. For example, "MultipleTaskSchedule" schedule contains 4 robot jobs: ExampleRobot1, ExampleRobot2, ExampleRobot2, ExampleRobot3. When a user manually runs the schedule, the log messages should look as follows:

```
2019-12-02 10:50:22        INFO      admin start Schedule MultipleTaskSchedule with id
 = 373284858997185
2019-12-02 10:50:22        INFO      Robot ExampleRobot1 with task id = 53 has been
 queued for schedule MultipleTaskSchedule with id = 373284858997185
2019-12-02 10:50:22        INFO      Robot ExampleRobot2 with task id = 54 has been
 queued for schedule MultipleTaskSchedule with id = 373284858997185
2019-12-02 10:50:22        INFO      Robot ExampleRobot2 with task id = 55 has been
 queued for schedule MultipleTaskSchedule with id = 373284858997185
2019-12-02 10:50:22        INFO      Robot ExampleRobot3 with task id = 56 has been
 queued for schedule MultipleTaskSchedule with id = 373284858997185
2019-12-02 10:50:22        INFO      Robot ExampleRobot1 with id = 1 execution id =
 3816-53-1dc33f3a2a44b task id = 53 has requested to start - from schedule, started by
 user
2019-12-02 10:50:22        INFO      Robot ExampleRobot2 with id = 39 execution id =
 3816-54-1dc33f3a2a44b task id = 54 has requested to start - from schedule, started by
 user
2019-12-02 10:50:23        INFO      Robot ExampleRobot2 with id = 39 execution id =
 3816-55-1dc33f3a2a44b task id = 55 has requested to start - from schedule, started by
 user
2019-12-02 10:50:23        INFO      Robot ExampleRobot3 with id = 20 execution id =
 3816-56-1dc33f3a2a44b task id = 56 has requested to start - from schedule, started by
 user
```

Chapter 5

SQL Scripts for Kofax RPA Tables

The SQL scripts for creating and dropping tables in your database are located in the `documentation\sql` directory in your Kofax RPA installation directory. For example, `C:\Program Files\Kofax RPA 11.5.0\documentation\sql` on the Windows system. The name of the script file includes the name of the database the script is intended for.

> ⓘ SQL scripts are installed together with Kofax RPA documentation and when you install Design Studio.

SQL Scripts for Database Tables

The `sql` directory contains four subdirectories with different scripts as follows:

- `kapplets`: Scripts for creating and dropping Kapplets tables
- `logdb`: Scripts for creating and dropping logdb tables
- `mc`: Scripts for creating and dropping Management Console tables
- `statistics`: Scripts for creating and dropping Statistics (Kofax Analytics for RPA) tables

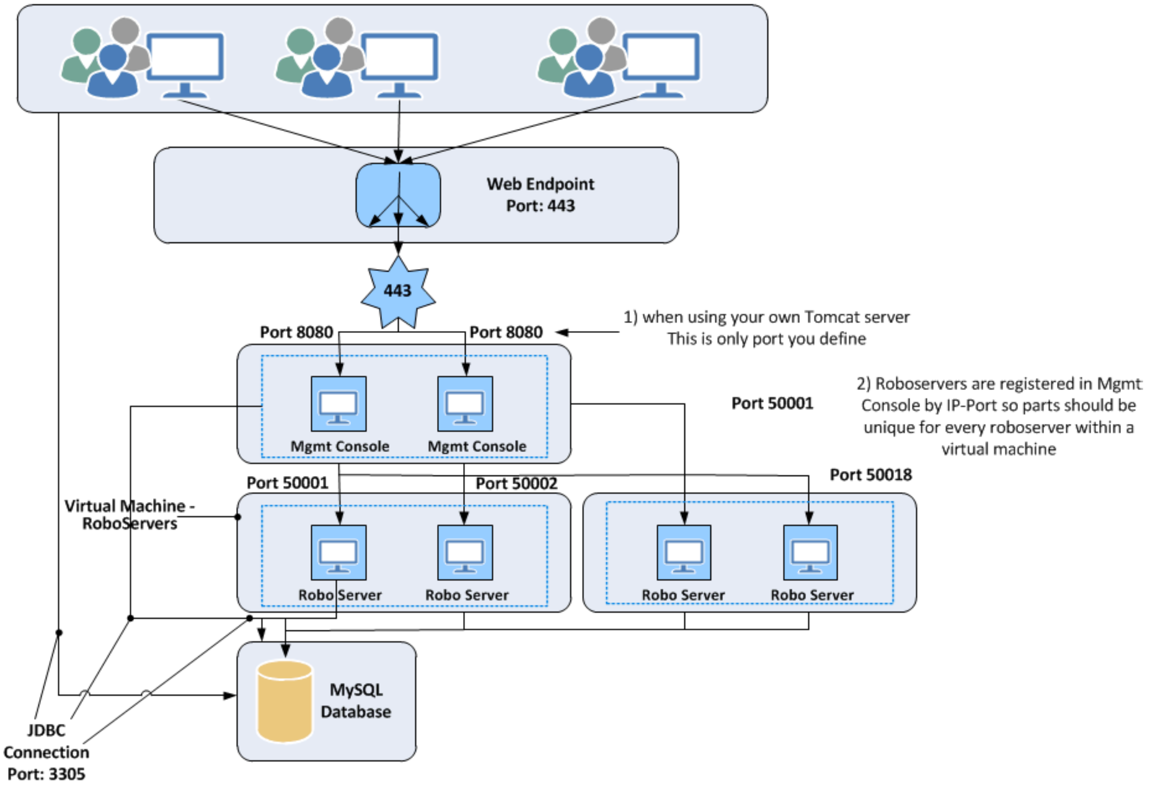
Management Console uses a 3rd party scheduling component called Quartz. Quartz also requires a number of tables which must reside among the other platform tables. These tables are also created automatically when the Management Console starts, or may be created manually using the scripts.

The following is a Quartz verification script.

```
select count(*) from QRTZ_SIMPLE_TRIGGERS;
select count(*) from QRTZ_BLOB_TRIGGERS;
select count(*) from QRTZ_CRON_TRIGGERS;
select count(*) from QRTZ_CALENDARS;
select count(*) from QRTZ_FIRED_TRIGGERS;
select count(*) from QRTZ_LOCKS;
select count(*) from QRTZ_PAUSED_TRIGGER_GRPS;
select count(*) from QRTZ_SCHEDULER_STATE;
select count(*) from QRTZ_TRIGGERS;
select count(*) from QRTZ_JOB_DETAILS;
```

Appendix A

Kofax RPA Security Model


Web Endpoint
Port: 443
443
Port 8080
Port 8080
1) when using your own Tomcat server
This is only port you define
Mgmt Console
Mgmt Console
Port 50001
2) Roboservers are registered in Mgmt Console by IP-Port so parts should be unique for every roboserver within a virtual machine
Port 50001
Port 50002
Port 50018
Virtual Machine - RoboServers
Robo Server
Robo Server
Robo Server
Robo Server
MySQL Database
JDBC Connection Port: 3305

A. User login and authentication

Category	Authentication and Authorization
Description	User provides login credentials for Kofax application.
Security Details	Kofax RPA supports synchronizing users/groups with Active Directory/LDAP. This allows Kofax RPA to take advantage of the corporate infrastructure for authentication and credential management. Kofax RPA also has an application-specific authentication and authorization mechanism for convenience. This includes credential management and storage. Stored passwords are encrypted.

B. Client transmits to Kofax RPA server(s)

Category	Data in transit
Port	80 or 443
Protocol	HTTP or HTTPS
Description	Clients transmit to the Kofax RPA servers.
Security Details	All connectivity from Kofax RPA clients (Management Console and Design Studio) to the Kofax RPA servers is via HTTP/HTTPS. HTTPS should be configured for maximum security.

C. Kofax RPA server(s) transmits to another Kofax RPA server(s)

Category	Data in transit
Port	Configurable. Defaults 80, 443, 50000, 50443, 49999, 49998
Protocol	HTTP/HTTPS, socket TCP/IP
Description	Kofax RPA servers transmit to/from another Kofax application or server.
Security Details	All Kofax RPA components can be configured to use secure encrypted communication (TLS 1.2) with custom certificates.

D. Kofax RPA servers transmit to Database server

Category	Data in transit
Port	Varies depending on protocol
Protocol	TCP/IP
Description	Kofax RPA servers transmit to/from database.
Security Details	The Kofax RPA servers connect to the SQL database. Typically, the database server system is co-located or otherwise physically protected such that transmission need not be otherwise encrypted. However, if such encryption is needed, you can encrypt the database connection via SSL.

E. Robot and Data storage

Category	Data at rest
Description	Robots, configurations and related metadata are stored via the Management Console. Robots can store customer data in databases.

Security Details	Robots, configurations and related metadata are stored in the Kofax database, which is accessed through a configured system account. Database level encryption is also available using the encryption feature within the database itself. Whether or not file system and/or database encryption is enabled, passwords (for external systems or application-specific users), are further protected. Passwords stored in the Password Store or as input to a schedule are encrypted using a customer generated certificate. We will use the cipher selected for the certificate to encrypt any stored passwords. By default, the installation comes with an RSA 1024 bit encrypted certificate, but we strongly recommend that the customer generates their own certificate. See Password Encryption for details.