Platform  
Overview

Content

[Introduction 3](#_Toc181596245)

[Platform Overview 4](#_Toc181596246)

[Headquarters Overview 5](#_Toc181596247)

[Robot Component Overview 7](#_Toc181596248)

[Platform Features 9](#_Toc181596249)

[Process Designer 9](#_Toc181596250)

[Steps Library 12](#_Toc181596251)

[Functions 14](#_Toc181596252)

[Process Lifecycle 16](#_Toc181596253)

[Scheduler 17](#_Toc181596254)

[Job API 19](#_Toc181596255)

[KPI Tracking & Reporting 22](#_Toc181596256)

[Realtime Monitoring 24](#_Toc181596257)

[Robot Remote Restart 25](#_Toc181596258)

[Global Variables 26](#_Toc181596259)

[Robot Remote Update 27](#_Toc181596260)

[Integrations with line of business (LOB) applications 28](#_Toc181596261)

[Security Overview 30](#_Toc181596262)

[HQ Cloud Infrastructure 32](#_Toc181596263)

[Role Based Access Control 34](#_Toc181596264)

[Encryption Of Data at Rest 36](#_Toc181596265)

[Encryption Of Data in Transit 38](#_Toc181596266)

[Site To Site VPN 40](#_Toc181596267)

[Single Sign-On (SSO) 40](#_Toc181596268)

[Robotiq.ai on-premise solution 41](#_Toc181596269)

# Introduction

Diagram

Description automatically generated with medium confidence  
Robotiq.ai is a **platform for business process automation using software robots** (RPA – Robotic Process Automation). Platform is a part of the 2nd generation RPA platforms who offer more **simplicity in implementation and consumption-based licensing models.** Software robots or digital workers replace humans in repetitive administrative tasks by mimicking their work on computers. Within any organization, there are a lot of manual processes in which employees lose a lot of time performing activities such as retyping data from one application to another, compiling reports and documents, copying files, processing customer requests, and others.

Spending years working with the customers we were able to understand all the complexities and technical drawbacks of the existing solutions. **We have built from scratch an enterprise-level RPA product** that combines the latest technical innovations and best practices from the field. Our vision is to enable our customers and partners to create true digital teammates in a simple and intuitive way.

The purpose of this document is to describe the components of the Robotiq.ai platform, emphasize key features and dive into security features and measures.

## A picture containing text, electronics Description automatically generatedPlatform Overview

Our platform consists of two main components: **Headquarters (HQ) and robots**. HQ is a management web application where processes are designed and where all aspects of robot orchestration and management are performed. Other RPA vendors call HQ Orchestrator, Control Room, or similar names.

HQ is hosted by Robotiq.ai (cloud component) or on premises while robots are either deployed on the customer infrastructure (**hybrid deployment**) or in the Robotiq.ai cloud environment (**cloud deployment**).

Robots perform in an **unattended mode** which means they are not installed on employees’ computers; they are deployed on virtual machines running in the data center.   
  
Cyber security is one of the most important topics in today’s IT and we take it seriously. Our platform is built with security in mind and is rigorously tested through penetration tests.

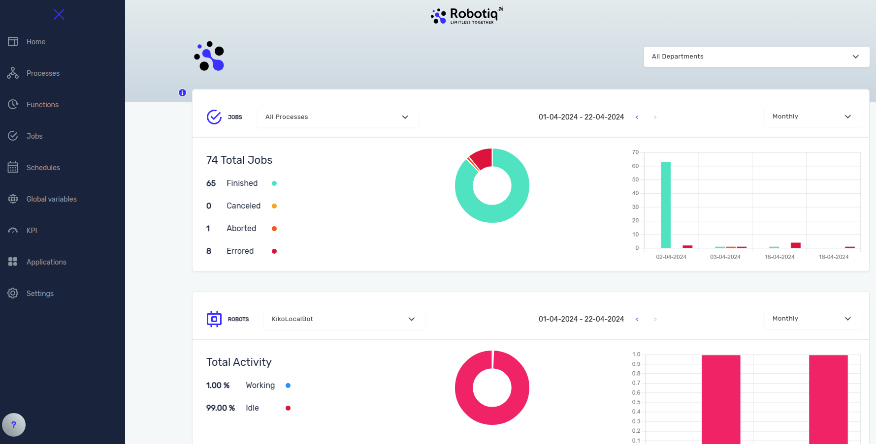
Confirmation of that is the fact that we have ISO certifications in areas of quality, security, cloud security and data privacy.

Availability and reliability of HQ components are also critical to us and our clients. HQ is running on a custom cloud service built from the ground up and is running in Microsoft Azure cloud across two regions: West Germany and Netherlands. Service is running on a scalable and highly available Kubernetes cluster with frequent backup procedures.

## Headquarters Overview

*Headquarters (HQ)* as the name suggests is the core component of the whole platform. It is a multi-tenant web application that provides all the necessary management and maintenance functionalities including:

* *Process Designer* – Main development environment in which automations are designed.
* *Lifecycle Management* – Versioning of the processes and publishing them (dev, test and prod).
* *Scheduler* – Component that starts processes on robots.
* *Execution Monitoring* – List of active and completed jobs with logs.
* *Robot management* – Registration, management, and monitoring of robots.
* *User management* – User, team, and department management.
* *Dashboards* – Reports about job execution and robot utilization.

HQ is based on Microsoft cloud infrastructure and if necessary, can be deployed on-premises (explained in details in the last chapter of this document). Additionally having a cloud-based HQ enables us to have a **rapid and flexible software release process**. We understand that having a cloud component raises a lot of security concerns and that’s why we went to great lengths to make sure that **we have the most secure RPA environment** possible. You can read more about our security related features and measures in a separate chapter of this document. With HQ in cloud clients get several crucial advantages:

* **Access From Anywhere** – You can access HQ from anywhere: home-office, work-office or on the go.
* **Faster Release Cycle** – New releases each month (improvements and new functionalities).
* **Faster Bug Fixing** – Response time for bug patching is around 1 day for smaller defects and bugs.
* **Lower Total Cost of Ownership (TCO)** – No need to worry about all the hardware and software needed for HQ (servers, storage, networking). There is no need for internal sysops personnel to maintain and monitor the deployment. This results in a much lower total cost of ownership.
* **High Availability, Redundancy** – HQ is backed in two different data centers on different locations and tectonic plates. It is deployed in a highly available model with no single point of failure.
* **Faster Security Patching** – Fixing security problems in a matter of hours.
* **Stronger Security** – Everything is encrypted.

HQ has been built using state-of-the-art technology following the best practices such as microservice architecture.

|  |  |
| --- | --- |
| Technology | Purpose |
| Kubernetes | Container orchestration system for automating software deployment, scaling, and management |
| Debian Linux | Operating system for the micro-service nodes deployed on Kubernetes |
| PostgreSQL | HQ database deployed in clustering mode |
| RabbitMQ | Used for message exchange (communication) with robots |
| KeyCloack | Authentication and authorization component |
| .NET Core | Framework in which the backend components were written |
| Angular | HQ User interface front-end framework |
| Open VPN | Used for securing access and for site-to-site VPN functionality |
| Azure DevOps | Automated build and deployment to HQ infrastructure |

Table 1. HQ - Used technologies and frameworks.

## Robot Component Overview

A robot is the component that does the actual work, orchestrated by Headquarters. A **robot is an application that is installed on a Windows based virtual machine**. Besides the robot application on this machine business applications used in the processes are also installed. This virtual machine is either hosted by the client on their infrastructure (hybrid deployment) or by us (cloud deployment). In most cases customers prefer the hybrid deployment model because of two reasons:

* Applications used in the processes only work from internal networks, they are not accessible from the Internet.
* For security reasons, customers want the processing to be done on their own infrastructure.

One thing which is important to note is that the **data that is sent to HQ from the robot (logs) can be configured in such a way that it does not contain any business data**. Only the metadata is transferred like which process was executed, at which time, what was the result status and so on. This is a very important fact related to GDPR and data privacy because **HQ will not contain any customer personally identifiable data**. In that sense we don’t have any issue with privacy regulations. Additionally, any sensitive data (global assets, robot credentials, processes) stored in HQ is encrypted (at rest and in transit). More on this topic you can read in the “Security Overview” chapter.

Robot is installed on Windows OS (versions 10,11 or Windows Server) and **cannot be installed on Linux or MacOS**. This was a conscious design decision as most of the business applications today run on Windows.

One question that we often get is: “Does one robot mean one process?”. A robot is in fact a virtual machine that **can execute as many processes as it can fit in with its time limitation**. There is a restriction though that it **can only execute only one process in given time** (no multitasking). But you **can have more robots performing the same process** which gives you scalability.

As mentioned before, a robot is merely an application installed on the computer. It’s not a single application that does everything, it also **has different components** to it that perform various tasks:

|  |  |
| --- | --- |
| Component name | Purpose |
| Script Executor | A command-line tool that performs the process as described in the process script. This would be the actual robot doing the work. |
| Recorder | Separate application that enables faster development time by recording clicks & inputs in fat client applications. |
| Gatekeeper | Windows Service that performs the login to the OS and invokes the handler component. |
| Handler | Component responsible for downloading the process script and other resources, starting the Script Executor, and uploading the log to HQ |
| Handler Monitoring Tool | Enables multiple instances of Handler component which allows multiple users on a single machine to develop and run robots. |
| Credential Provider | Unmanaged code (C++) component that does the actual login to Windows. |
| Robot Monitoring Tool | Responsible for updating the robot application, listening to Gatekeeper for updates, communicating installation logs to HQ |

Robot is developed in .NET technology, except for the Credential Provider component.Platform Features

## Process Designer

Process Designer is the heart of the Robotiq.ai and of the key components of the platform. It is a development environment where you program the digital workers and design your automation. Unlike many other vendors where this component is a desktop application, **our process designer is web-based** which means you can create and modify your processes from anywhere. As you can see from the figure, implementing a digital worker is **more like business process modelling or workflow rather than hard-core programming**, although we enable that as well. Because of this low-code nature of the solution it enables us to automate the process in days or weeks.

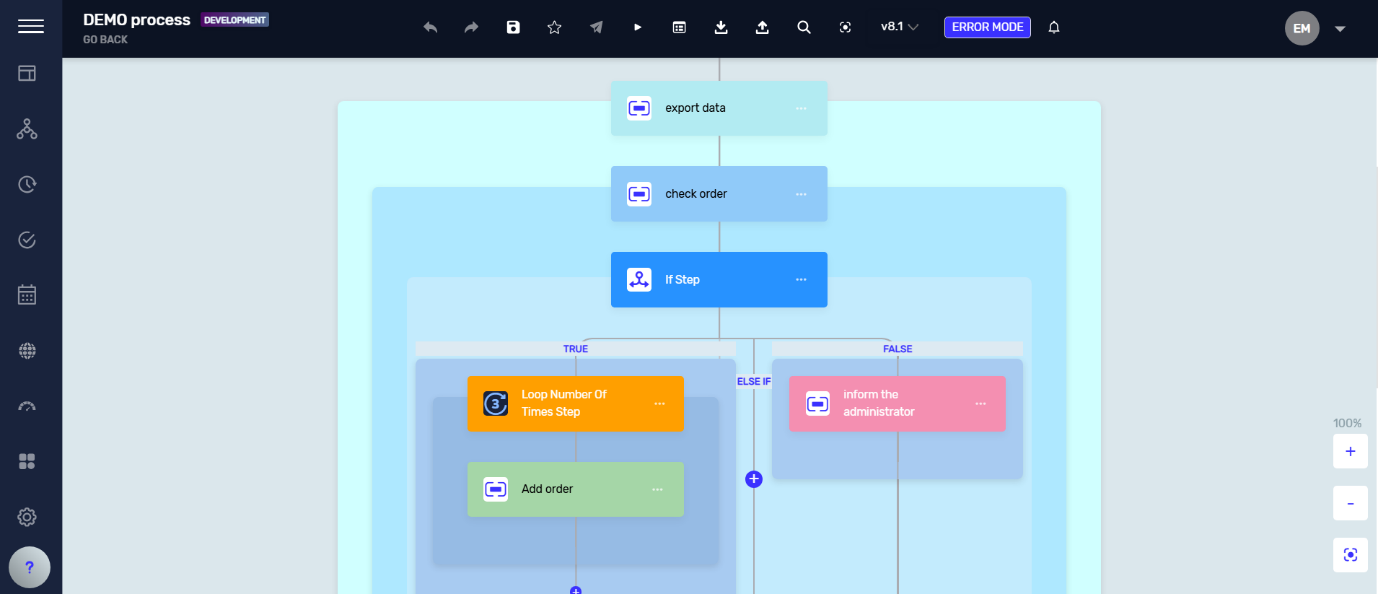


Figure 1. Process Designer

**Processes are modelled by adding series of steps (activities)** that perform various tasks like interacting with the application, Excel, file systems and others. Steps are described in more detail in a separate chapter. We support typical functionalities expected from such a designer as crosstab copy/paste, undo/redo, debug, process import/export and others.

Designer also has **capability to define try/catch blocks to capture errors and has a separate exception mode**, a place where you define what needs to be done when technical errors occur. For example, if an error happens you can send an email to the process owner, raise a support ticket in the helpdesk process and gracefully close all the applications and finish the process.

When working on larger processes there are a lot of variables being used in many of the places. So, if you decide to delete or modify some variable there is a question where this variable is referenced. There is a handy feature called variable viewer that can quickly answer this question and help you be more efficient during development.

## Recorder

What **helps immensely with speed of development is the recorder component**. Recorder is started from HQ, and it is a separate electron application that is monitoring and recording all the steps you are performing (click, key inputs and others) and once you are done those steps can be copied to the process designer. This way you don’t have to manually add every step in the process. You can record many micro sequences (couple of clicks in the application) and parametrize them.

A screenshot of a computer

AI-generated content may be incorrect.

Recorder has a mini process designer in which you can edit and delete steps before copy/pasting them to the regular process designer in HQ.

Click steps use advanced computer vision algorithms to locate the image it needs to click on. Each of the recorded steps can be edited to show where exactly did we click and does the algorithm detect the image. We can also provide anchor images (what is located left, right, top and bottom of the image) to increase robustness of the algorithm.

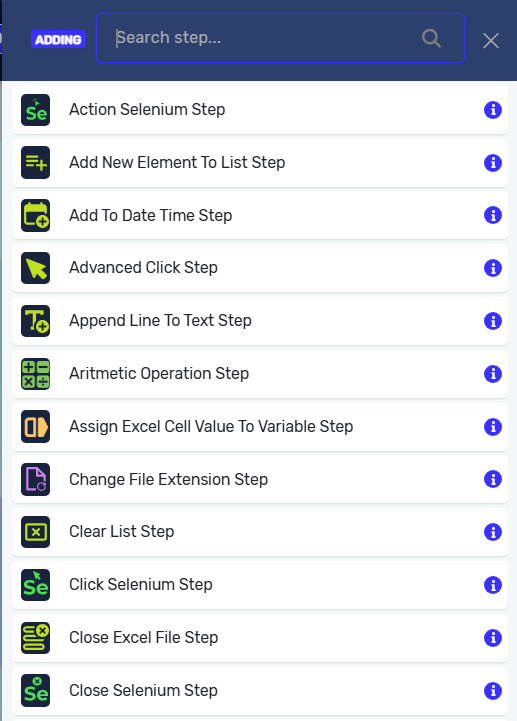
A screenshot of a computer

AI-generated content may be incorrect.

Recorder can be used for recording fat client applications such as Windows, Java, Oracle Forms, SAP and other types. In the future recorder will be expanded to support web automation and it will generate Selenium based steps when working with web apps.

## Steps Library

As mentioned, steps are the building block of processes that are assembled like Legos to create a copy of the process as executed by human employees. Currently on our platform there are more than 100 different types of steps that you can use to create your own process. We can group them in several categories:

* *Application interaction steps* – open/close application, mouse clicks, keyboard inputs, hotkey inputs, find image step …
* *Branching and looping steps* – if/then/else branching, while and do/while loops, switch/case, for and for each loop
* *List and data table steps* – working with list and data table data types including creation, adding, reading and removing elements, sorting and other operations.
* *Excel steps* – open/close excel files, read/write cells, and other operations.
* *File system steps* – create folder or list files in directory, move, rename, copy, and delete files, zip/unzip, write and read from a file.
* *Text manipulation steps* – convert text to number (and vice versa), pad text, parse text, get text length, contains text, append line to text, get subtext, to lower/uppercase, trim text, …
* *Date manipulation steps* – convert text to date (and vice versa), get current date, add, and subtract date steps.
* *Selenium steps* – used for web automation scenarios with steps such as open/close browser, navigate, create, and close tab, go back/forward, click, input text, get value from element, …
* *Email steps* – used for sending and receiving emails from any type of email system (Exchange, O365, Gmail or any other POP, IMAP, and SMTP servers).
* *Database steps* – connect and query any type of database (MS SQL, MySQL, Oracle, IBM, PostgreSQL, Informix) from the robots.
* *Error handling and extensibility steps* – try/catch blocks, stop process step, script step (coded steps in .NET), display state step, command prompt step (execute cmd based applications)
* *Other auxiliary steps* – other steps that do not fall into any category such as take screenshot step, read text from pdf, pause step, read/write to clipboard, etc.

Since there are a lot of steps, sometimes it is hard to find the step you need. **Step search functionality** comes handy in such situations with search as you type functionality.

All steps have property editor that contains relevant properties and configuration. For maintainability and readability, a series **of steps can be grouped together** to form a logical group of steps.

If there isn’t a specific step that you need in the library, you can **extend the platform** either by writing .NET code or Python in their respective script steps or by invoking an outside code or script (Powershell, VBA, Node.js,…). Coding or scripting experience is required for such tasks.

*A screenshot of a computer

AI-generated content may be incorrect.*

## Functions

Many processes in your organization might have some parts that are the same. For example, they might be using the same application as ERP system. So, it makes sense to extract parts that are the same and reuse them across different processes. That might be as simple as functionality to login into the application. By extracting those functionalities, you increase the speed of development and make it easier to maintain the system.

You can do that with functions, a functionality of Robotiq.ai platform, that are like mini processes with one caveat: they have input and output parameters. Input and output parameters can be of many different data types such as integer, decimal, date, boolean, string, and others. These functions can then be called from processes using the Execute function step.

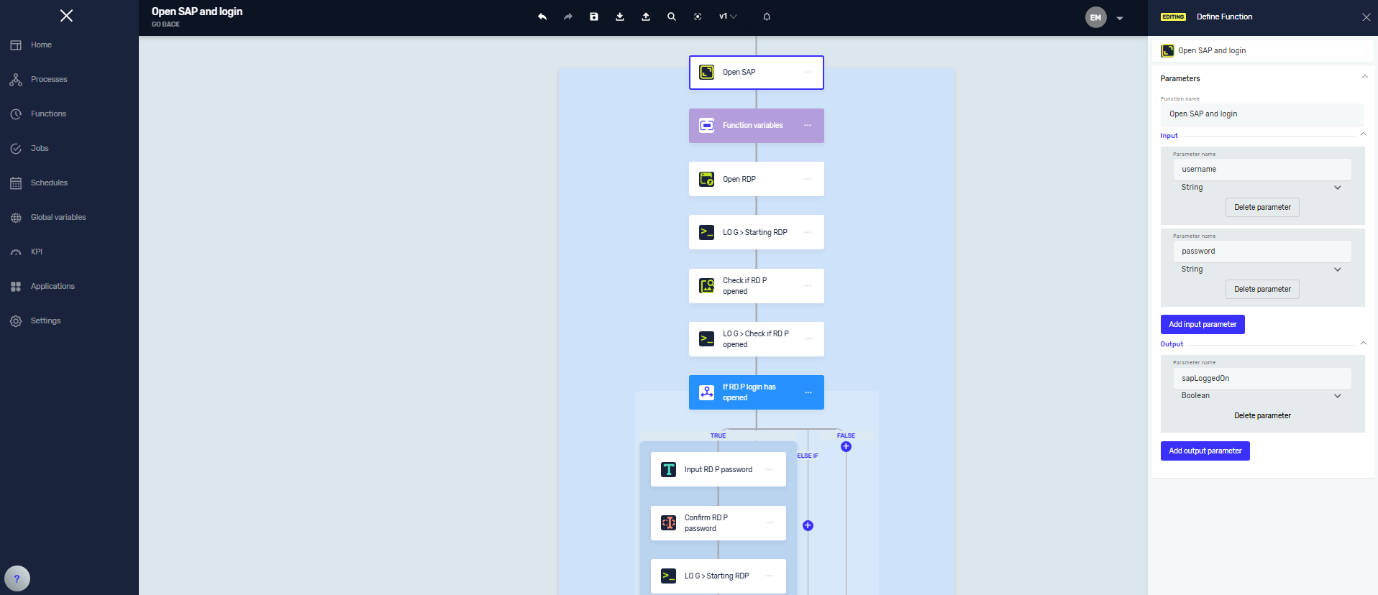


Figure 2. Example function for SAP login

In the example above we have a function for SAP system login that has input parameters username and password and one output parameter that returns to the main process information was the login successful. We can then use this function in many different processes that use SAP application and if anything changes in the login logic, we can just change it in one place, and it will be reflected in all the processes that reference it.

Using the functionality, we can build a library of reusable process components which is a useful feature especially if you have a large-scale deployment with many implemented processes.

If a function changes and there is a new version available, a notification will appear when working with the process that calls that function. User then has an option to update the calls to the new version that is available or continue using the existing function version.

A screenshot of a computer

AI-generated content may be incorrect.

## Process Lifecycle

All processes on the Robotiq.ai platform go through different stages: from development to testing and finally production. This enables scenarios where one version of the process is currently being executed on the production stage, while consultants are developing and testing the updated version of the process.

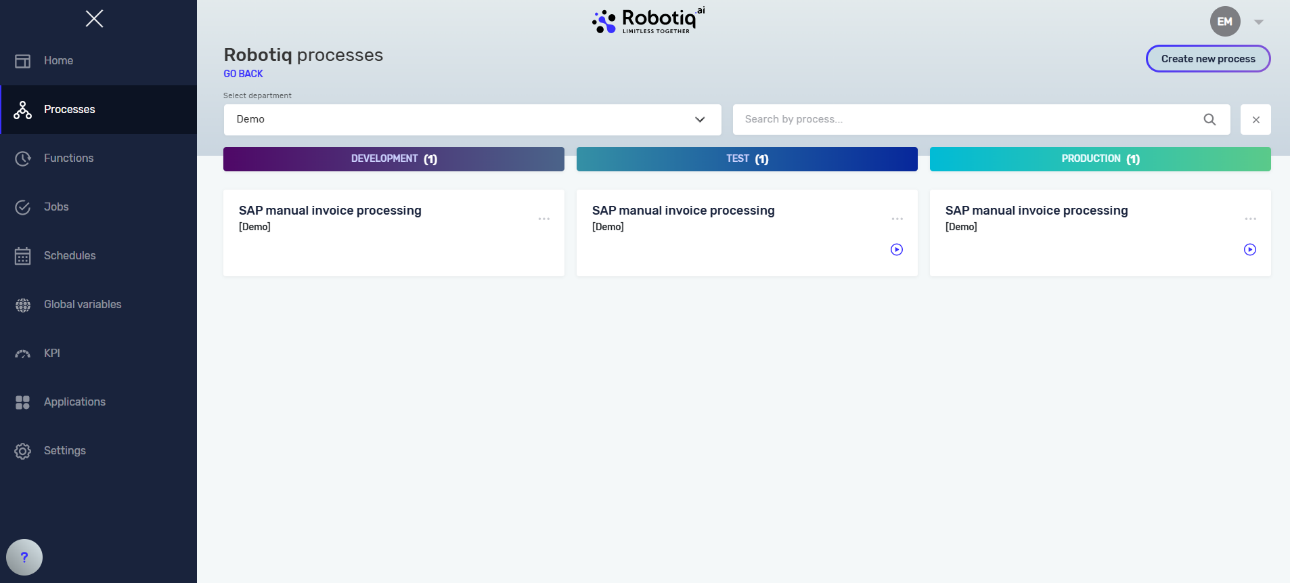


Figure 3. Process stages

There is a difference in how processes are executed depending on the stage they are at. In the development stage processes are **executed on the computer of the user** while processes in test and production stages are meant to be **executed unattended on virtual machines** (robots).

Additionally, processes in testing and production stages are read-only and cannot be modified in order not to create discrepancies across the entire lifecycle.

One functionality that is related to the lifecycle is the **process versioning**. When working in the development stage of the process on each save of the process there is a new minor version created (0.1,0.2,0.3 and so on) that we call process draft. When the development is done, the process needs to be promoted to the official version which creates a new major version (1.0, 2.0, 3.0 and so on). The official process version can then be **published** to test stage and then from test to production.

## Scheduler

The primary way of executing processes in production is through a scheduled execution. Our scheduler has a couple of different ways of executing the process:

* *One time* – used when you just to need to execute process once. There are two options: run immediately or run at a specified date/time.
* *Recurring* – used when you need to execute the process multiple times. There are several options for a recurring schedule:
  + One time per day – you can specify the time and on which days of the week you need to run the process.
  + Multiple times per day – you can specify the time interval (like every 2 hours), limit the interval by specifying from which hour to which hour and select the days of the week when the process needs to run.
  + One time per month – you can specify on which exact day in the month or week of the month you need to run the process and during which months. For example, you can say run this process on Monday of the second week of January, February, and March.

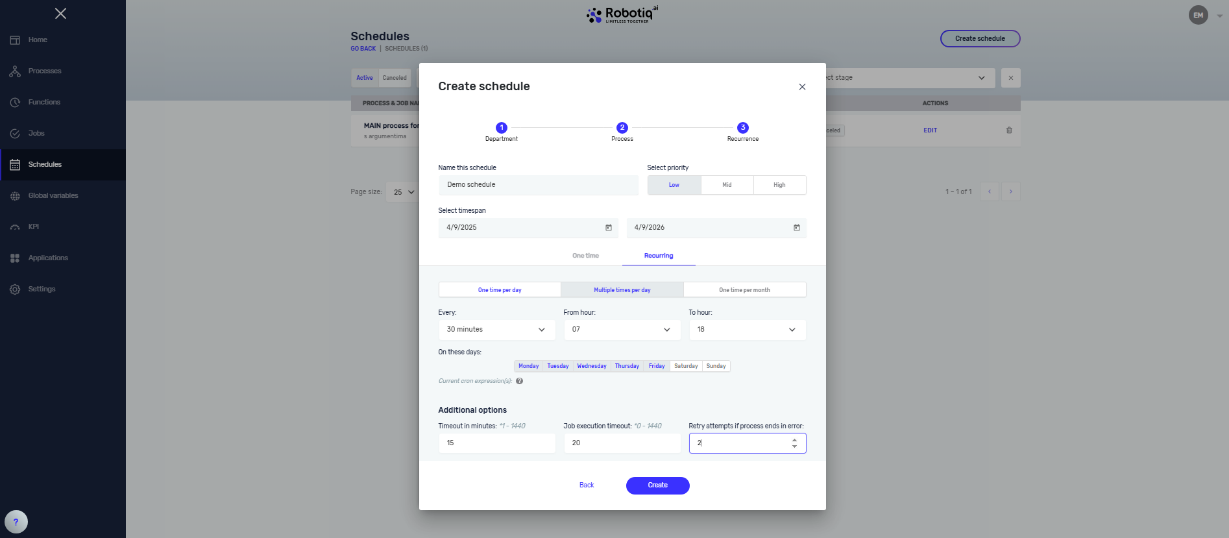


Figure 4. Scheduler

When the schedule is created Headquarters will create the jobs for the robots and make sure that the robots execute them. Jobs created from schedules can be seen in Active Jobs and Completed Jobs lists available from the main HQ navigation.

Scheduling has additional options and functionalities which are important:

* *Priority* – determines the priority of the jobs that will be created from this schedule. Higher priority jobs will get executed faster than lower priority jobs.
* *Schedule validity* – limits the time span during which the schedule is used.
* *Timeout* – determines the number of minutes that HQ will wait for the process to get executed by the bot. If that timeout is exceeded the job will go to cancelled status.
* *Job execution timeout* – defines the maximum execution time for the process. For example, when set that the process can last for max one hour. This prevents the process that is somehow stuck to keep the bots occupied even though it’s idling.
* *Retry attempts* – if the process fails, we can set how many times the robot will try again.

## Job API’s

Another way of executing the processes in production is by creating a job using the HQ API call. This enables integration scenarios where you can integrate your application with the bots. For example, you can integrate in your application when a user clicks “Submit” button that the job is created which is then executed by the robot.

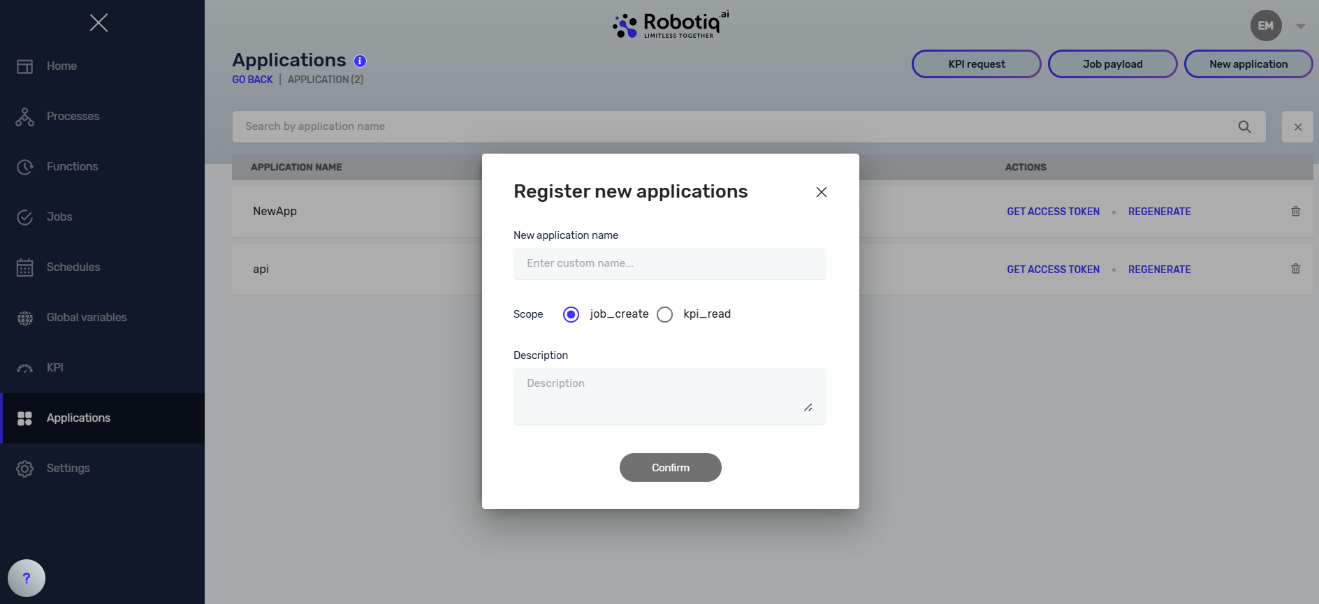


Figure 5. Application registration

To be able to call the HQ job API you first need to register your application in the Settings -> Applications. For the ClientID you specified you will get the Client Secret you will specify in your API requests. Creating a job is a two-step process:

1. Retrieve the access token.
2. Call and API endpoint.

In most cases you also want to pass some input parameters needed to execute the process. Those parameters would come from your application and would be passed to the robot. Also, you might have an output argument for the process that will contain results for process execution. These start and output arguments are defined in the process designer. When you create a new process in the designer on the top of the process is a step called “Process Start and Output Arguments”. In there you can specify the input parameters for the process that will then be passed on as a JSON payload of the Start Job API. Also, you can specify the output parameters that will be stored after the job execution and will be available when calling the Get Job Info API endpoint.

A screenshot of a computer

AI-generated content may be incorrect.

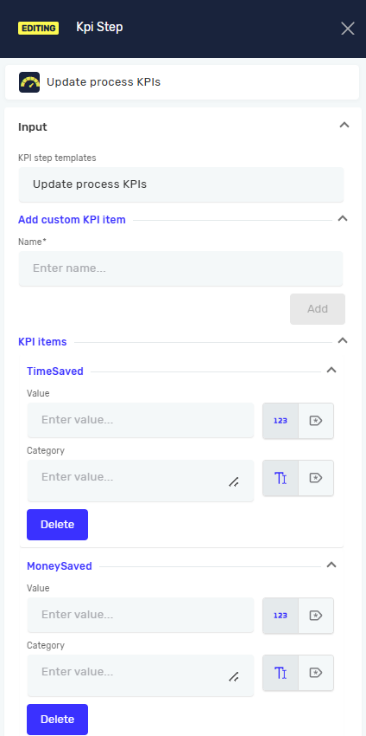
Figure 6. Process start and output arguments

Besides starting a job there are various API endpoints that you can call to get a list of departments, processes, process information, job info and job logs.

The table below shows all the endpoints, their inputs and outputs and URL. List of endpoints will be expanded to include getting info about organization, robots, restarting robot machines and getting list of jobs using various filters.

|  |  |  |  |
| --- | --- | --- | --- |
| Name | Input | Output | URL |
| Get departments (GET) | none | JSON list of department names and department ID’s | *https://api.robotiq.ai/{{tenantName}}/departments/info?apiVersion=1.0* |
| Get processes in department  (GET) | departmentId stage | JSON list of processes with:  processInfoId departmentId name description | *https://api.robotiq.ai/{{tenantName}}/processes/info?apiVersion=1.0* |
| Get process info  (GET) | processInfoId | JSON with attributes above plus list of start and output arguments and version of the process | *https:// api.robotiq.ai//{{tenantName}}/process/{{processInfoId}}?apiVersion=1.0* |
| Start Job  (POST) | JSON payload with job properties | jobId | [*https://api.robotiq.ai/{{tenantName}}/job*](https://api.robotiq.ai/%7b%7btenantName%7d%7d/job) |
| Get Job Info  (GET) | jobId | JSON with all properties of the job (status, output args and other metadata) | *https://api.robotiq.ai/{{tenantName}}/job/{{jobId}}* |
| Get Job Log  (GET) | jobId | JSON technical log of execution | *https://api.robotiq.ai/{{tenantName}}/log/{{jobId}}* |

## KPI Tracking & Reporting

Since digital workers are performing a lot of business processes that were previously done by human employees, they can report on the work they did and provide valuable insights into savings they are making and other business-related data. They can track how much time they saved for your employees, financial savings, contract values, number of support tickets they processed and so on.

Robotiq.ai platform enables you to define KPI’s and update them (increment or decrement) from the processes using the KPI step. You can increase/decrease the KPI value by integer and decimal value from one or more processes. Besides the value there is a possibility of also defining a category (label) of the value. This way you can track for example how many different types of loan requests (real estate/ vehicle/ cash) you processed. Each KPI update will create a new record in the database together with a timestamp when it was created so you can track KPI’s through time.

Usually customers have their own reporting & dashboarding tools of choice, and we didn’t want to create our own solution for the problem that has been already been solved. So, we allow you to retrieve the KPI data through API which means you can create visually stunning reports and dashboards using a tool of your choice. That can be plain old Microsoft Excel or some of the more advanced tools such as Tableau, QlikView or Power BI. From there you can expose the reports & dashboards to whomever the information is needed.

Graphical user interface, application

Description automatically generated

Figure 7. Dashboard example in PowerBI tool

## Realtime Monitoring

Troubleshooting unattended robots is a bit difficult because the execution is happening on the remote machine, and you can’t see what is happening on the screen as you would see on attended robot. Of course, you can take screenshots in case of error, and you have execution logs, but sometimes you also need to see what is happening in real time. That means that you need to connect to the robot machine via RDP. That could have an impact on process execution of the robot and might cause it to fail.

Realtime Monitoring functionality of Robotiq.ai platform allows you to peek inside the virtual machine without disturbing the robot in execution. In HQ you can activate the real time monitoring functionality, and it will start streaming two sets of data:

* *Video stream* – you can see the video of the robot machine desktop.
* *Log stream* – you will receive output of technical step executions in real time so you can see what steps the robot is performing.

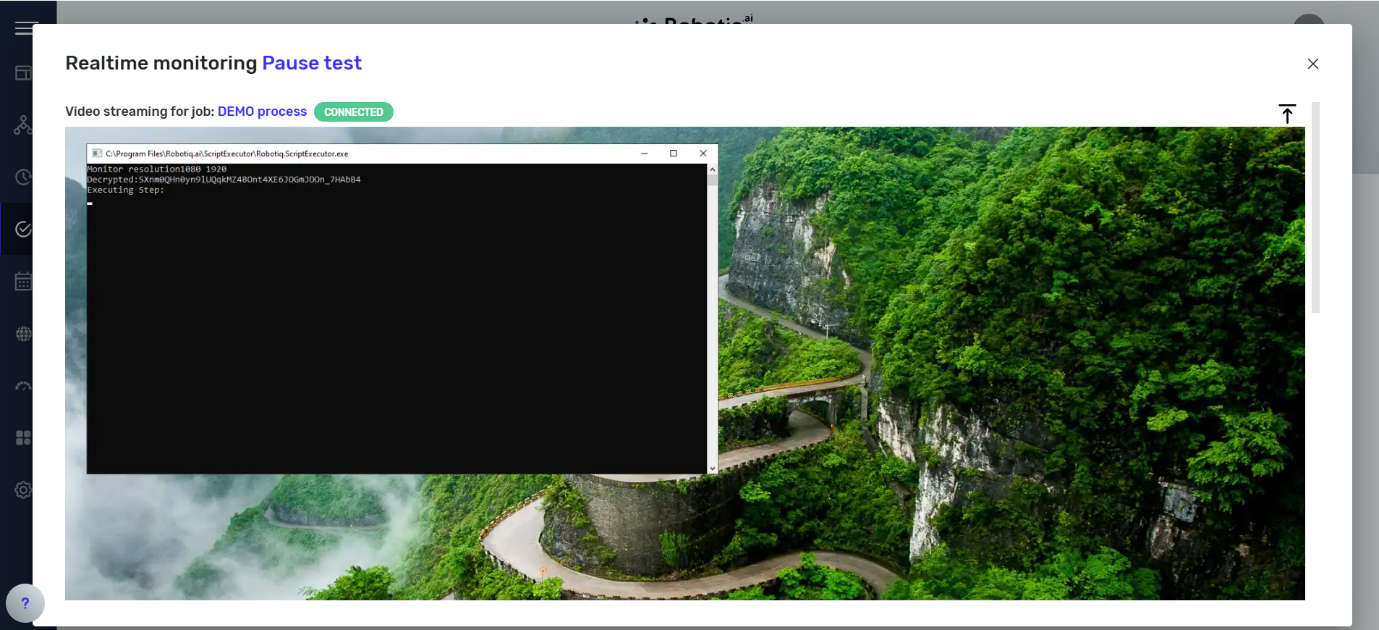


Figure 8. Realtime monitoring

## Robot Remote Restart

Sometimes there is a need to restart the robot virtual machine for various reasons: application updates, robot process is stuck or other reasons. To achieve this, you would need to remotely connect to the machine via RDP and that might take some time. With Robot Remote Restart, small but useful feature, you can initiate restart of the machine with one click from the HQ. If you are tenant admin you can navigate to the list of the robots and just click “Restart”.

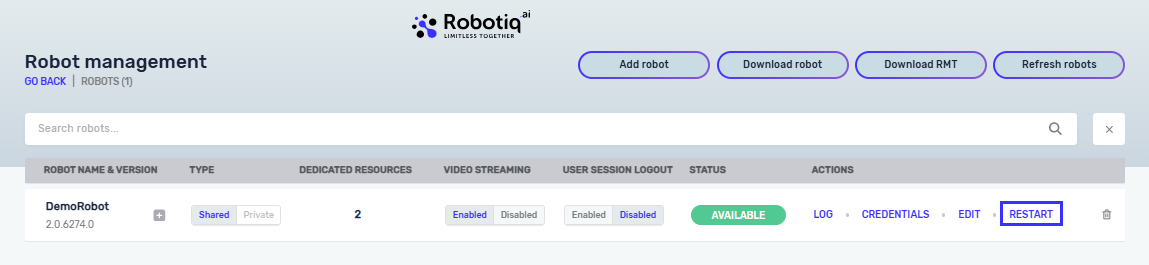


Figure 9. Robot Remote Restart

## Global Variables

Instead of hardcoding variables in your process such as URL’s, username and passwords, file paths and other data you can create them as global assets in HQ. This means you can extract the configuration data and business rules, share them across processes and manage them centrally from HQ.

Global assets can be applied to process, department or tenant level depending on the level of access you need. Also, global assets can be assigned to a certain stage (dev, test, or production) which means you can specify different values for different environments. For example, the web application used in the process will most probably have a different URL for test and production environments. With stage assignment you can have a single global asset with different values and the robot will fetch the right value depending on which stage the process is.

A screenshot of a computer

Description automatically generated

Figure 10. Global Variables

There is also an option to protect the global variable value by encrypting it and hiding the value that was entered when displaying it in HQ. This is useful for storing sensitive data such as application credentials and other. If Bring Your Own Key (described in the security chapter) feature is used, then the protected data is encrypted using the uploaded certificate.

## Robot Remote Update

As mentioned, before we are releasing new versions of our platform monthly. Often that includes update to the robot client component. That means that you need to download the robot installation, connect to each of the robot machines and manually perform the installation. Now imagine if you have a large number of machines that can be a lengthy and cumbersome task.

That’s why we have created a feature called Robot Remote Update. With this feature you can update the robot installation directly from HQ with only one click. The platform will do the rest. It will deploy the installation to the machine, do a silent update and then report back if it was successful or not.

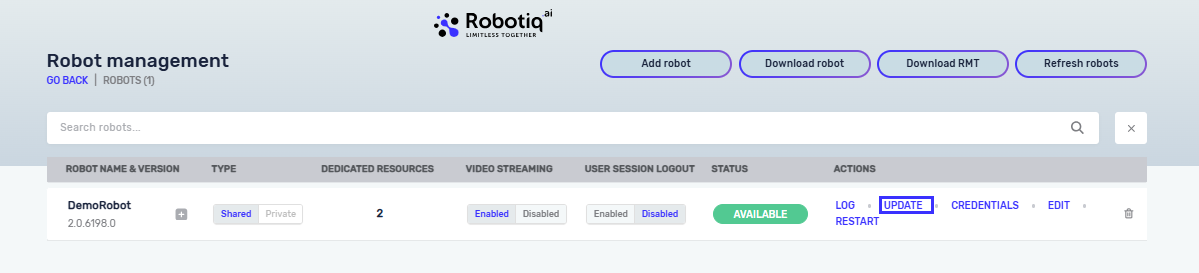


Figure 11. Robot Remote Update

# Integrations with line of business (LOB) applications and other platforms

## Integrations with line of business applications

Integration with applications and systems deployed in your IT environment can be achieved on two levels:

* *User interface (UI) integration* – This is the primary way RPA is used. In this case the robot is interacting with applications in the same manner as a normal human user would do it.
* *System integration* – In this approach robots can use different programmatic interfaces to connect to your systems. It can be API calls (SOAP or REST calls), database access or other means of code-based integration.

When it comes to UI integration you would use the steps from the Step Library such as click step, key input step and others to interact with the application. Since Robotiq.ai uses computer vision to interact with the application we can interact with different types of apps built on different technology stacks:

* Windows Desktop applications
* Web application (web browser automation)
* WPF applications
* Java applications
* Oracle Forms
* DOS-based applications
* IBM Host (mainframe) applications
* Applications exposed through Citrix or Remote Desktop Protocol (RDP)

If there is an option to integrate with the LOB systems on the system level that can bring significant benefits such as much faster execution time, less maintenance due to application updates and increased robustness of the process.

If the application is exposed through API’s we can use .NET or Python Script Step. Or maybe there is an option to read/update data through database layer and then you can use the built-in database steps to connect to it and perform the necessary operations.

You can also use other means of integration such as invoking various scripting engines (Powershell, VBS, Node.js,…) or something simple as file drops.

We have integrated Robotiq.ai with various types of systems used by our clients such ERP, DMS, BPM, CRM, BI/DW, OCR and others. Just to name a few of the most popular ones:

* SAP
* Camunda
* SharePoint
* Aris
* Oracle APEX
* Salesforce
* Abby

In most scenarios Robotiq.ai is calling and interacting with various systems. But also, it is possible to integrate it the other way around so that other systems are invoking robot execution. This is achieved through Job API which is described in a separate chapter of this document.

## Platform Connectors

Robotiq.ai RPA platform has been integrated with a couple of other Integration Platform as a Service (iPaaS) products such as HCL Universal Orchestrator, Workato and Zapier. These platforms have built-in native connectors for Robotiq.ai which means you can start jobs and get execution information without having to call the APIs manually.

More details on HCL UNO connector: <https://www.yourautomationhub.io/detail/robotiq>

More details on Workato connector:  
<https://docs.workato.com/connectors/rpa-by-workato.html>

# Security Overview

Robotiq.ai platform is an enterprise grade RPA platform and **security is one the biggest focus areas and differentiators**. Product is used by many organizations such as banks, insurances, telecoms, and others. We handle sensitive processes and data and that’s why we made a huge investment in making our platform reliable and secure.

The cloud infrastructure for HQ as well as the services we provide are ISO certified in different areas:

* *ISO 9001* – Quality
* *ISO 27001* – Security
* *ISO 27017* – Cloud Security
* *ISO 27701* – Data Privacy and GDPR

*Text

Description automatically generated with medium confidence*

Figure 12. ISO certificates

To test our HQ infrastructure, we submitted it to vulnerability scans and penetration test. This service was handled by an external company which specializes in this area. Cyber security mark after penetration test is LOW which is a confirmation that we took all necessary measures to protect our IT environment.



In the following chapters we will describe different security features of the platform.

## HQ Cloud Infrastructure

Cloud infrastructure that hosts the HQ components is based on Microsoft Azure with two data centers in West Germany and Netherlands. Azure provides high availability and disaster recovery and also enables auto-scaling according to application load.

HQ is deployed on Kubernetes Linux based cluster running on three separate physical machines which host Kubernetes nodes. High availability is achieved through clustering and there are also backup procedures in place for database servers and application servers.

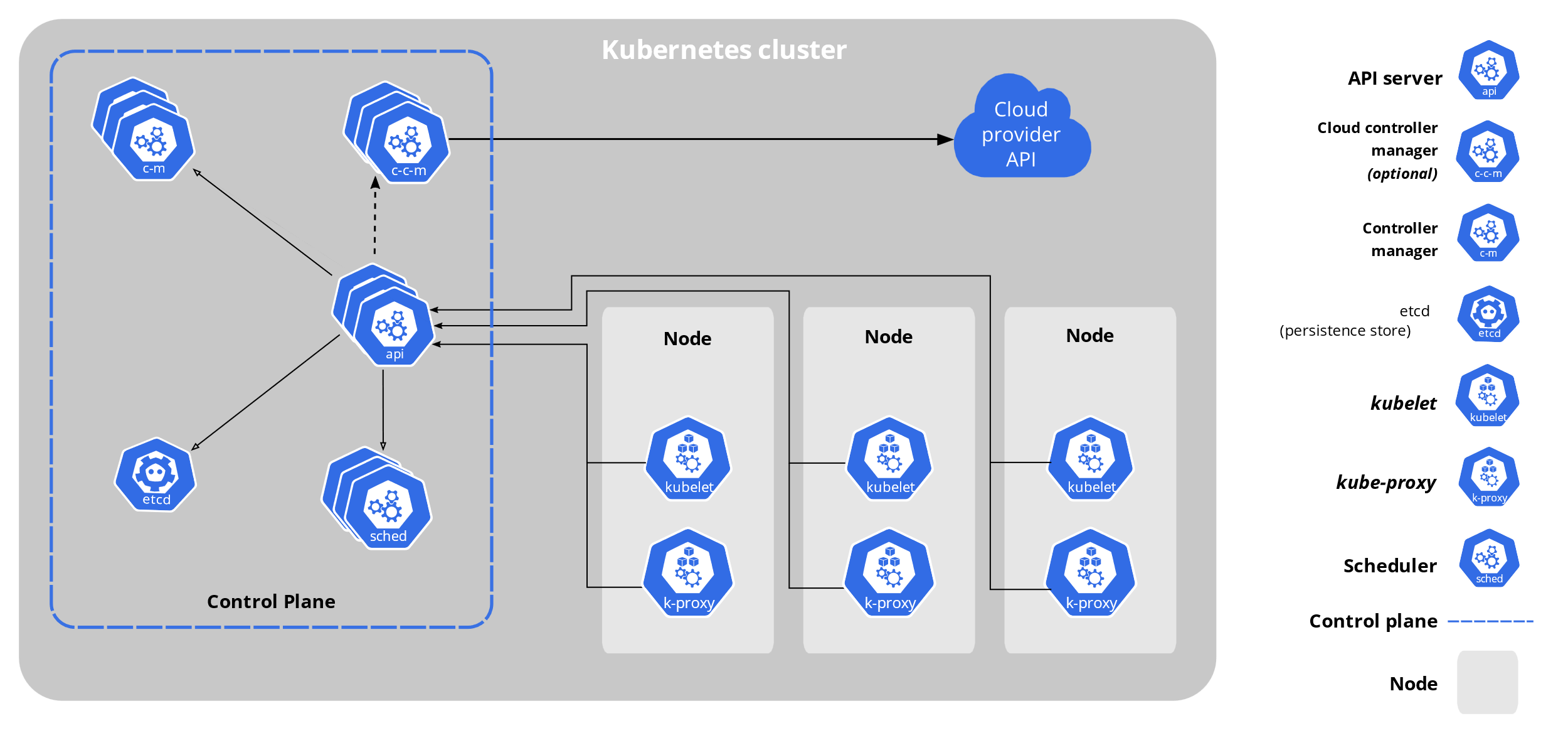


Figure 13. Kubernetes stack

Access to servers is protected by VPN which is filtered by IP range so the connection can only be made from Robotiq.ai offices. Only authorized personnel can access the servers. Deployment of new versions is automated through Azure DevOps pipelines, so no physical access to servers is needed except for maintenance purposes.

Diagram, schematic

Description automatically generated

Figure 14. HQ topology

## Role Based Access Control

HQ has several security related concepts that enable us to separate data and provide role-based access to it. The first important concept is called **tenant** and it represents whole organization or company. Tenant can have one or more **departments** and they contain **processes**. For example, we could have a Finance department, HR department, Customer Support and so on.

Figure 15. Security model in Robotiq.ai

The second concept is the **users** which are assigned to **teams**. Teams can have **tenant admin** rights or can have a specific **security role** assigned per department. Members of the team can have **department contributor** or **department admin** role. Department admin has more rights than the contributor role in respect that this role can publish processes into test and production stages and run processes in those stages.

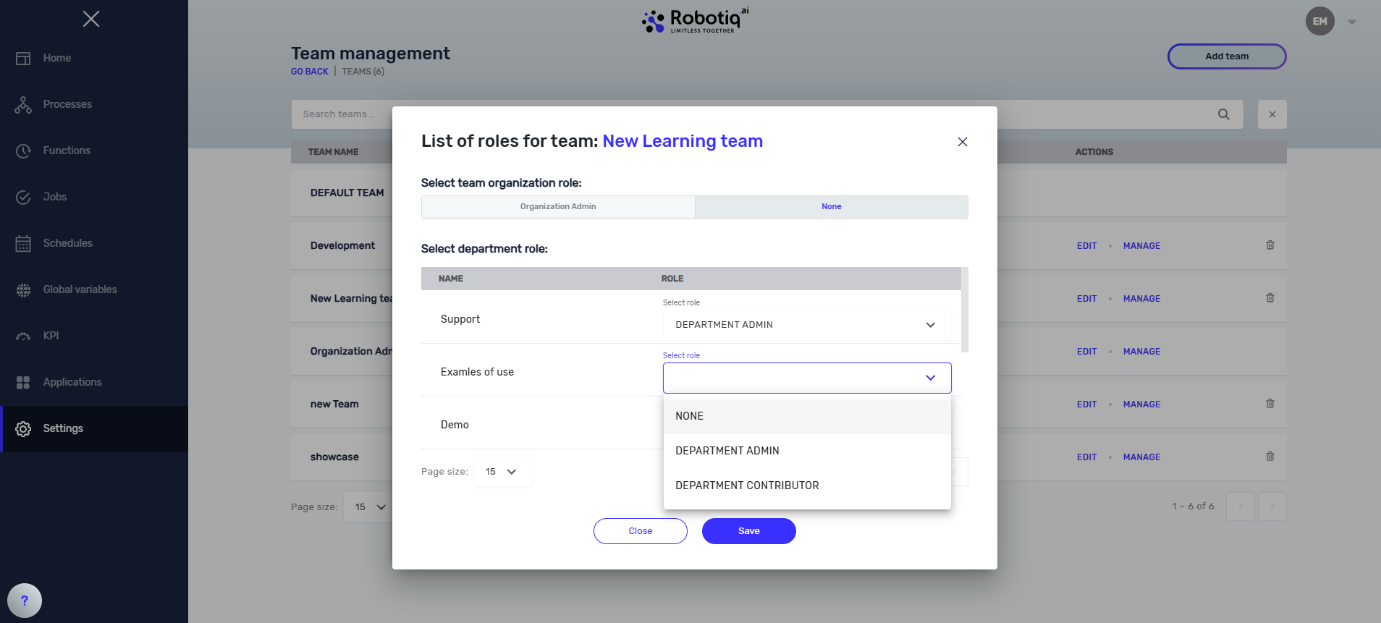


Figure 16. Team permissions

Robots are also assigned to departments, and we can declare that the robot is either **private** or **shared**. If the robot is private, it is exclusively assigned to a specific department and a specific stage. For example, we can have a robot execute processes on test stage of HR department. On the contrary, if the robot is shared then we can specify to which departments and stages it is assigned to.

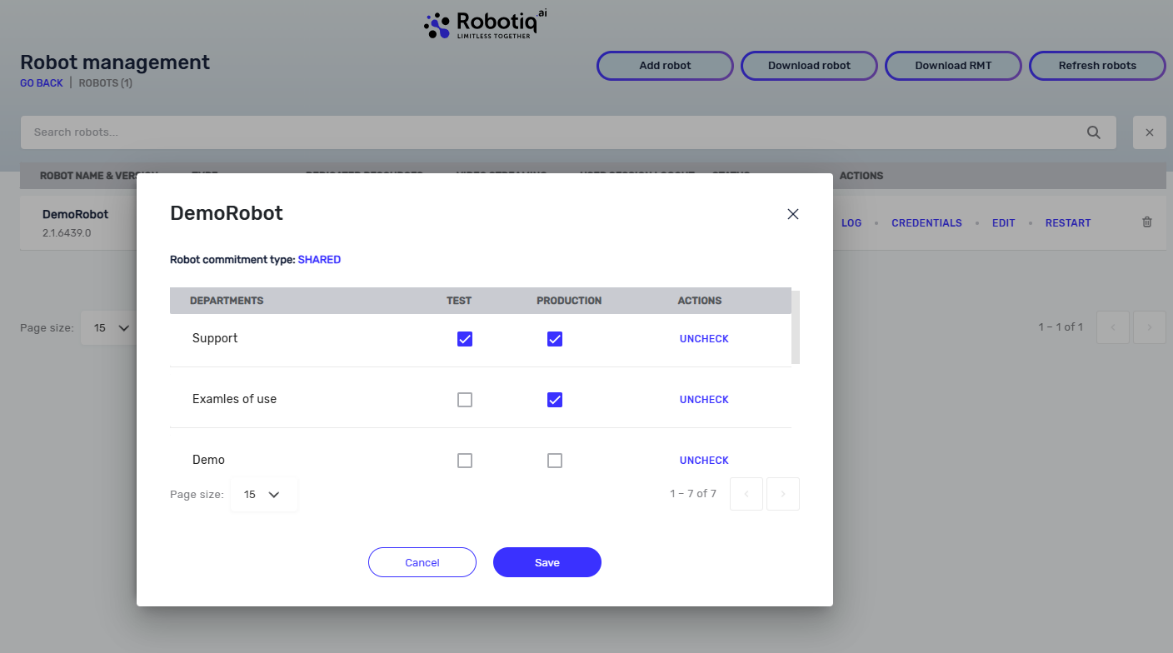


Figure 17. Robot assignment to departments

This has an implication on what jobs the robot will execute. If the robot is not assigned to a specific department, it will not receive any jobs for that department.

## Encryption Of Data at Rest

A major concern in cloud solutions is protecting the data that is stored in the cloud service provider (CSP). At Robotiq.ai we offer BYOK encryption for the end customers. Bring Your Own Key (BYOK) is an encryption key management system that allows enterprises to encrypt their data and retain control and management of their encryption keys. Its primary focus is to give the client ownership of data that is critical for the client (business critical).

A screenshot of a video game

Description automatically generated with medium confidence

Figure 18. Bring Your Own Key Concept

All sensitive data in the HQ database is encrypted. This includes various types of data:

* Process scripts
* Functions
* Job logs
* Schedules
* Global variables (login credentials for applications, sensitive data in variables…)
* Robot login credentials

Data protection at rest means that client data is protected with encryption at the storage layer.

For data protection at rest, we use symmetric cryptography. The algorithm currently used for generating keys is AES256 with GCM or CCM cipher modes. These keys are data encryption keys or short DEKs.

DEKs are stored encrypted along with the encrypted data. Encryption of DEK is done with key called Key Encryption Key or KEK. KEK is client-provided key, and it is imported by client.

For DEKs we have a key rotation policy to ensure that keys don’t encrypt too much data (key exhaustion) and to reduce the attack surface which lowers security risk of user data being compromised.

We offer clients the possibility to rotate DEKs from one day to one month. Rotating DEKs doesn’t have tradeoffs in performance.

KEK used for encrypting and decrypting DEKs is stored in a separate location, in Azure based key vault backed by HSM (Hardware security module).

## Encryption Of Data in Transit

Securing data is important both at rest and in transit. When users access HQ via web browser, they do so via HTTPS protocol which means traffic is encrypted between the user and the HQ. Robots also communicate with HQ, and they do so via mTLS encrypted protocol. When a bot is registered, HQ generates a new security certificate for that specific robot. This certificate needs to be installed on the robot machine for the installation to be completed. So, each bot has a different certificate and if one bot is compromised the certificate cannot be used for man-in-the-middle attack.

A picture containing diagram

Description automatically generated

Figure 19. Encryption Of Data in Transit

To add an extra layer of protection and to give the client control of his data in motion, we introduced similar mechanisms for data at rest.

Upon Robot installation, the client has the option to import public/private keypair onto robot VM. If the client does this, Robot won’t generate its own keypair and will use client provided keys to encrypt and decrypt data at rest and in transit.

Robot has its own local storage for process scripts and logs which makes the need for protecting data at rest. Since it is on local machine, it will be protected by default with robot public/private keypair.

There is possibility for robot to store private key on robot virtual machine in secret storage (Windows CSP) or to send the request to decrypt on local KMS (this is available only if client exposes KMS/HSM API to robot VM, and client configures robot to work on this option). In the second case, there will be performance downgrade since robot needs to decrypt data and for each decryption it needs to request it from client KMS/HSM.

By importing private/public keypair on robot side, client can periodically change keypair, reducing attack surface, or even in the situation if keypair gets compromised, simply change it. It will wipe out the data on robot side, since without private key from keypair, data couldn’t be decrypted. It is not a concern, robot will start pulling data from HQ again and store it locally for further use (process scripts, logs, etc.).

Business critical data that goes in from HQ to Robot is:

* Process script
* Functions
* Global variables (login credentials for some applications, etc)

Business critical data that goes from Robot to HQ is:

* Job logs

All this data is encrypted with data in motion.

## Site To Site VPN

To further protect user and robot communication with the HQ it is possible to establish site-to-site VPN connection. This means that the communication to HQ will be limited to from customer site to HQ site and the user from the customer organization will be accessing HQ through VPN tunnel.

## Single Sign-On (SSO)

Authentication and user management in Robotiq.ai is handled by KeyCloak service. Each user sets their own password for accessing HQ. In enterprise environments there is often a need to provide a more seamless experience so that the users can login with their organization credentials. That can be achieved through Single Sign-On (SSO) scenario where the authentication is delegated to the customer through ADFS (Active Directory Federation Services) or similar other services. The experience for the user is when they open HQ and click the login button, they are forwarded to the customer identity provider to authenticate. Once they are authenticated HQ gets the token and the user is authenticated to use HQ.

A screenshot of a computer

Description automatically generated with medium confidence

Figure 20. SSO experience with Azure AD

# Robotiq.ai on-premise solution

Robotiq.ai on-premise solution is exactly the same as Cloud hosted solution (regarding functionality) but has different deployment. Features and functionality of on-premise solution is the same as cloud based which is described in the document Robotiq.ai – Platform overview.

Talking about cloud vs on-premise solution, we are actually talking about on prem or cloud hosting Headquarters application (HQ app). Hosting robots has already been explained in the document Robotiq.ai – Platform overview.

Main differences between cloud and on-premise solution:

* **No BYOK functionality** – since deployment is on the client infrastructure, BYOK functionality is needed to secure data at rest in cloud, but on on-premise solution Robotiq.ai relies on client provided PostgreSQL database which can be encrypted based on client setup
* **HQ has several services shipped as Docker images** **– instead of Kubernetes cluster** which is on cloud infrastructure, **here is Docker compose** needed to run the HQ - simplified installation
* **Cloud solution is managed by Robotiq.ai team** along with patches and new releases, **on-premise solution is managed by client infrastructure team** (installation of new releases and patches) along with the help of one of our colleagues (if needed).
* **Robot – HQ communication –** **is AMQPS, without certificate sign-in and mTLS for robots** – since it is on client's infrastructure (local network), there is no need for generating robot certificates. But if client insists on it, there is a way to import certificate for each robot installed and to enable mTLS and AMQPS communication.
* **In on-premise solution client provides service certificates and manages those certificates** for HTTPS and AMQPS communication between services – we provide environment variables so that client's infrastructure team can place the needed certificates somewhere where services can reach them for communication
* **In on-premise solution client infrastructure team needs to give hostnames (DNS records) for Frontend, gateway, and authentication service** so that client end users can reach HQ application
* **Provisioning new releases for on-premise solution,** client infrastructure team has granted access to Robotiq.ai repository where signed Docker images for HQ are stored. Then they can download new images and install them following instructions given by the release management team from Robotiq.ai.

These are the main differences between on-premise solution and cloud hosted solution of Robotiq.ai.

Below is the detailed picture with HQ and Robots topology, along with protocols, ports used in communication and brief description of each service that HQ consists of.

