movizon CONTROL **Installation Guide**

System Overview

movizon CONTROL (mC) is a software that is specialized in controlling Automated Guided Vehicles (AGVs) in an industrial environment. mC is no executable application on its own, but a package of Java servlets ready to be hosted by a Java servlet container or application server. For more details please see the chart displayed on the last page of this document.

FOSS List

The following free and open-source software (FOSS) packages are used in mC v4.1.3:

Product name	Version	Homepage	License	mC usage
H2 Database	2.1.214	www.h2database.com	MPL 2.0	[optional] embedded production database
HikariCP	5.0.1	github.com/brettwooldridge/HikariCP	Apache License 2.0	JDBC connection pool management
Gson	2.10.1	github.com/google/gson	Apache License 2.0	JSON serialisation / deserialisation
Monaco Editor	0.32.1	microsoft.github.io/monaco-editor	MIT License	[client side only] script code editor

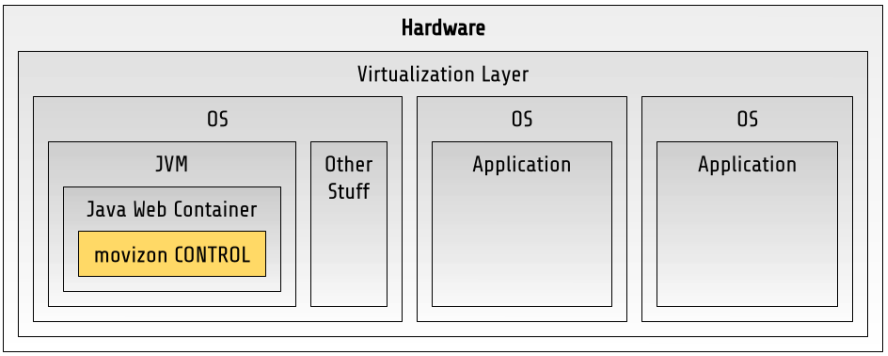
System Requirements

Server Hardware

A manufacturer of an application has to provide system requirements. The application in this case is your AGV system: mC as a framework plus its modules and project specific customization together with the interfaces and network architecture needed to communicate with your AGVs and other connected systems. Since your AGV system most likely is or will be first of its kind, it is impossible to define exact system requirements in advance. What we can do here is to give you insights into how mC is working to help you to build a suitable hosting environment.

Resource Sharing

All information in this document regarding hardware requirements assumes that any provided hardware is solely dedicated to operate mC. In reality, the available resources have to be shared. Every layer that is superordinate to mC only offers a share of the allocated resources based on its configuration – which is not the responsibility of movizon. When dimensioning the server hardware, please keep the surrounding architecture in mind which, in simplified form, could look like this:

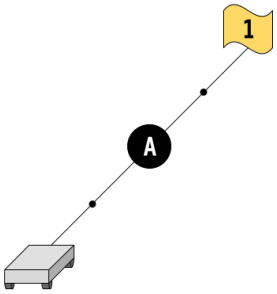
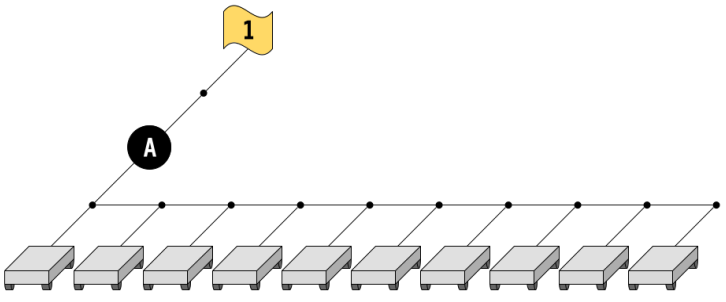


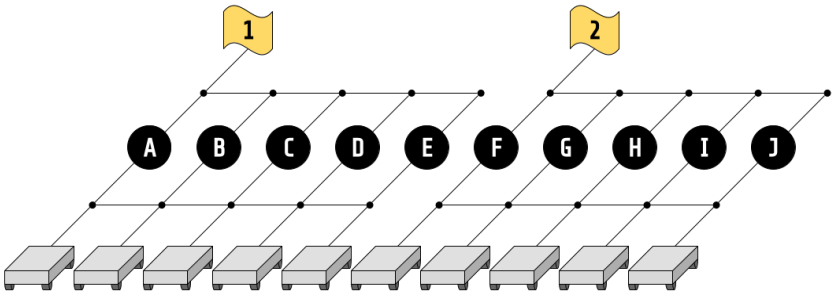
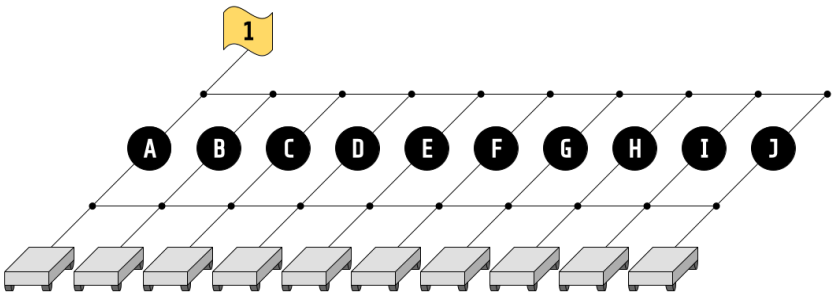
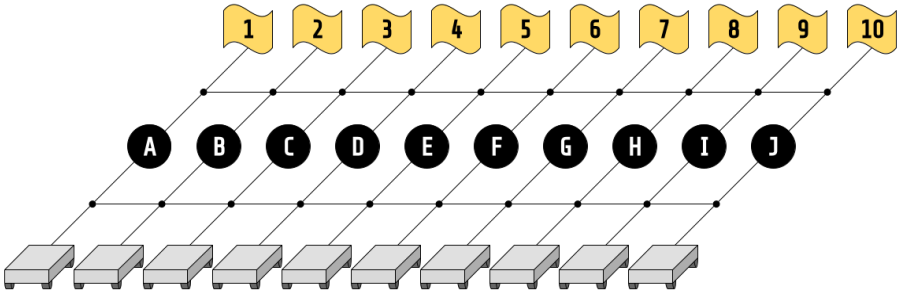
Performance Impact Factors

How does the size of the AGV system correlate with the needed hardware resources? First, it is all about the CPU. mC needs to constantly calculate and optimize a lot of options. Due to the architecture of the runtime these calculations primarily utilize the CPU of your system. In most use cases the biggest performance impact is derived from the following three elements:

- ➔ The number of AGVs that can be assigned to a job
- ➔ The number of route options for an AGV from current position to job destination
- ➔ The number of jobs that can be done by multiple AGVs

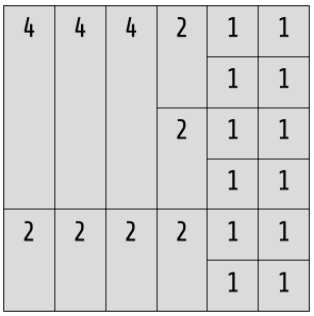
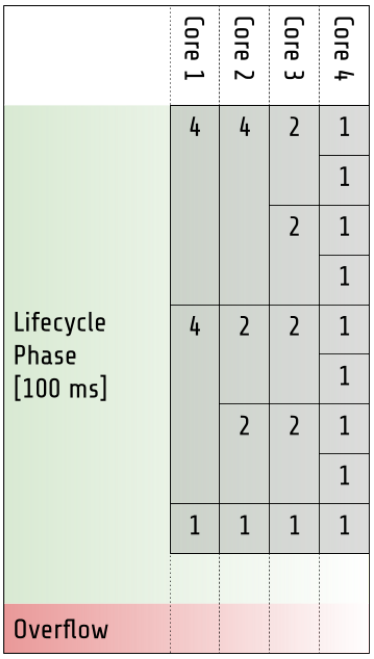
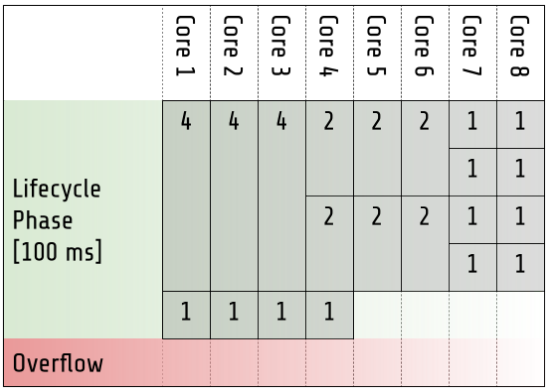
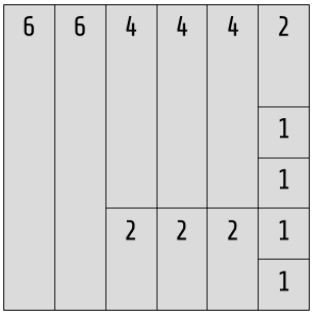
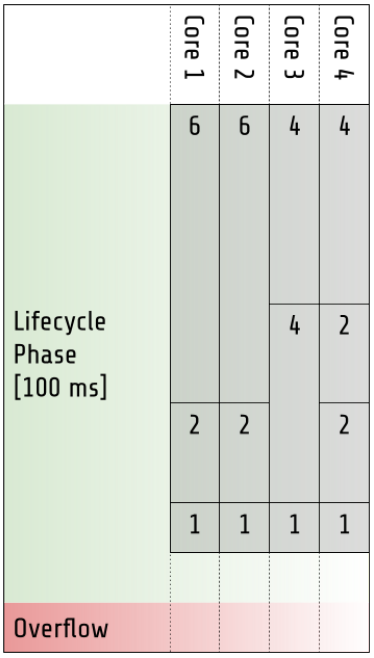
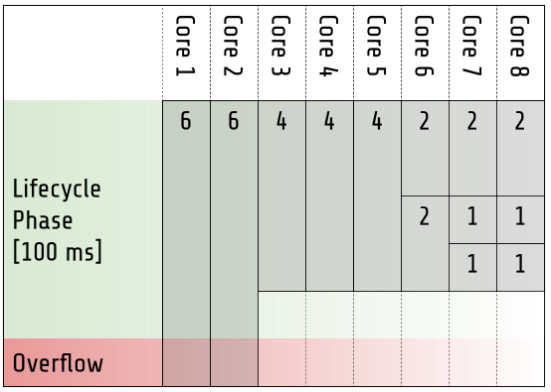
If you want to expand your current AGV system managed by mC, please refer to the following table to get an approximation of the expected increase in the CPU load:

AGV Options per Job	Route Options per AGV	Jobs	Performance Impact Factor	Graph
1	1	1	1	
10	1	1	10	

5	5	2	50	
10	10	1	100	
10	10	10	1000	

CPU Selection

Calculations in mC are executed in scripts or modules each of which can perform one or multiple tasks. Each task runs in a separate thread. mC can't manage the distribution of threads on CPU cores directly, but the JVM will try to utilize all cores if allowed by the operating system. Ideal for running mC would be a CPU with a lot of fast cores, but from a financial point of view you have to choose between a CPU with more but slower cores and a CPU with less but faster cores. Latter is better in most cases even if the CPU load could vary according to the structure of modules and scripts in your application:

Task structure 6 x 6	CPU with 4 fast cores	CPU with 8 slow cores
		
		

RAM

As a rule of thumb: For best performance let the size of the memory be at least the size of the production database used by mC in your AGV system.

Hard Disk Space

- ➔ Some space for the production database wherever it is located, exact values depend on the project
  - Size of database in small projects is under 1 MB
  - Size of database in large projects is about 10 MB
  - Size is not growing because database is for resource persistence, not for history data
- ➔ A maximum of 1 GB of local hard disk space for logfiles, more details here

Server Software

- ➔ Oracle GraalVM for JDK 20
- ➔ Java web container that supports Jakarta Servlet 6.0 specification or higher
  Recommendations: Apache Tomcat 10 or Jetty 12
- ➔ Any operating system that supports both packages above

Browser UI

- ➔ Up-to-date browser with Blink engine, for example
  - Google Chrome starting at version 28
  - Microsoft Edge starting at version 79

Network

- ➔ Full WLAN coverage in the operating area of the AGVs
- ➔ For most AGV types: fixed server address
- ➔ Infrastructure that supports WebSocket connections
- ➔ Low-delay data throughput, standard communication frequency of mC is 10 Hz
- ➔ Permanently open UDP and TCP ports, for more details please ask for a project specific communication structure chart

System Preparation

Please refer to the manual of the chosen servlet container or application server for general setup information. mC runs with the default configuration of common servlet containers after installation, so no universally applicable mandatory configuration steps can be listed here. Additional modules or packages of a Java EE application server in comparison with a plain servlet container should be deactivated because mC does not need any of them. Regarding security settings please follow your organization's policies.

Firewall Settings

mC includes a UDP server to communicate with the AGVs. Each AGV will send telegrams from its own address to the server, while mC uses a single UDP port (configurable, default 2222) for outgoing telegrams. Please ask for a project specific communication structure chart and set your firewall rules carefully.

ATTENTION: If no target shows up inside mC (see browser UI or server logs), one can safely assume that there is a network or firewall problem. Tools like Wireshark listen to telegrams right after the network adapter, so communication might still be cut off by the firewall of the server's operating system.

Database Connection

mC reads all resources from the production database on startup and keeps them in memory. During runtime there is no read access any more. Changes to the resources are collected and written to the database once per core loop run (default setting 10 Hz). JDBC connections are managed by an integrated connection pool manager. For each resource type in mC one database table is used to store all resources of this type. There is no need to prepare database tables because mC creates them automatically if needed. Each table contains only two columns, the first one is for the ID of the resource, the second one holds a JSON string representation of the whole resource.

Using the Embedded H2 Database (Recommended)

mC ships with an embedded H2 database that is very much suited for the use case described above. If not configured otherwise via JNDI, this database will be connected automatically on startup. It uses (or creates if not present) a single-file-datasource with the path [MC_BASE]/WEB-INF/config.mv.db.

Connecting an External Database

It is possible to connect an external production database to mC. Do this at your own risk and only if you have experience in configuring and operating the database type you want to use. It may or may not run reliably with good performance for your project in your system environment. Theoretically, most SQL databases should work with mC. As far as we know, customers have successfully connected MySQL and Oracle databases. To connect an external database, connection details have to be deposited as a JNDI entry in the appropriate configuration file of the container. In case of using Apache Tomcat this configuration file would be conf/context.xml and the entry could look like this:

Example of JNDI entry to connect a MySQL database to mC

```
<Resource name="jdbc/tcOperation" auth="Container" type="javax.sql.DataSource"
    maxTotal="20" maxIdle="10" maxWaitMillis="-1"
    username="ADMIN" password="r7Gh4k-6#z" driverClassName="com.mysql.jdbc.Driver"
    url="jdbc:mysql://localhost:1433/tc"/>
```

LDAP Connection

Users in mC can be externally authenticated (not authorized!) via LDAP. Since user requests are strictly stateless each request must be authenticated for itself. Therefore user credentials will be sent unencrypted to an authentication service that can be connected to mC with the help of a JNDI entry like this:

Example of JNDI entry to connect an LDAP authentication service to mC

```
<Resource name="ldap/tcAuthentication" auth="Container" type="javax.naming.ldap.LdapContext"
    factory="de.movizon.control.kernel.LDAPFactory" singleton="false"
    java.naming.factory.initial="com.sun.jndi.ldap.LdapCtxFactory"
    java.naming.provider.url="ldap://192.168.178.88:389"
    java.naming.security.authentication="simple"
    java.naming.security.principal="UID=admin,OU=people,DC=gsf,DC=de"
    java.naming.security.credentials="e4RE$dw2+a" />
```

Deployment

mC will be provided as a zipped folder whose size is less than 20 MB. If the hosting environment demands a war archive simply rename *.zip to *.war and deploy. If not, unpack the archive and copy it into the deployment directory off the hosting container. mC uses annotations to describe its servlet so don't look out for a web.xml file.

Startup

After deployment start or restart your servlet container or application server if needed. mC logs events separately from the logging system of the container. Those logfiles are located in [MC_BASE]/logfiles. Up to 1000 files will be created – each with a maximum of 1 MB – before the oldest will be overwritten. Look into the latest logfile (named server_0.log) to verify a clean startup. If mC's logfiles folder is empty, refer to the logfiles of the container because it might have caught a more serious problem.

A successful startup of mC is also indicated by displaying the login or licensing screen after calling the browser UI with one of the following URLs:

Main UI	Scheme	[PROTOCOL]://[SERVER_ADDRESS]/[MC_BASE]/
	Example	https://127.0.0.1:8080/movizoncontrol
Main UI with auto login	Scheme	[PROTOCOL]://[SERVER_ADDRESS]/[MC_BASE]?username=[USERNAME]&password=[PASSWORD]
	Example	https://127.0.0.1:8080/movizoncontrol?username=customer&password=project2020

ATTENTION: While mC allows Unicode characters in passwords, only ASCII characters may be used inside URLs. Please choose passwords accordingly if you want to use the auto login feature.

Licensing

It is not possible to log in and work with mC without a valid license. Licenses are deposited on a remote license server. To retrieve a license it is necessary that mC was started on the customer system beforehand. So if you are greeted with the licensing screen after calling mC's browser UI, please follow the displayed instructions. You should have an email stating the ID of the registered license owner to use in order to request your license.

ATTENTION: mC ensures that a user for the registered owner of the license is always present and has full access to all resource types. This user can't be deleted or limited in any way.

Maintenance

mC is designed as a maintenance free application. No mandatory actions need to be planned. Depending on the use case and project, a regular backup of logfiles and the database would be optional actions. mC's hosting environment is likely to be serviced regularly. Although mC can be shut down and restarted via container for this purpose without loss of data, make sure to consult the AGV service team in advance so that they have the chance to ensure safe conditions on the AGV side too.

ATTENTION: In case of using the embedded H2 database please note that the H2 driver locks the database during runtime of mC. On Linux systems it might even look like you've successfully copied the database, but your copy will be hollow. So to create backups of the database, mC must be stopped by shutting down the hosting container.

Update

Since most of our customers use offline production servers, there is no remote update procedure for mC. New versions of mC will be provided just like the package for the first installation, so an update is basically a redeployment. If you use the embedded H2 database please take it over to the new version manually. You might also want to save mC's logfiles before deleting the previous version.

ATTENTION: Please beware if you want to rename and keep the last version of mC inside the deployment directory of your servlet container. If multiple instances of mC tries to startup, only the random first instance will be able to bind the UDP ports if their numbers collide. You can change UDP ports anytime after startup via browser UI.

Example: Installation on Windows Server with Apache Tomcat

Follow these steps to create a basic hosting environment for mC:

1. Download and install GraalVM for Java 20 (20.0.2)
    a. Currently there is no installer for GraalVM, so just unzip the downloaded archive and copy its content to an appropriate location, referred to as [GRAALVM_BASE] in this example
2. Install GraalVM's JavaScript Engine
    a. Run Windows Command Prompt (cmd.exe) as administrator
    b. Change directory to [GRAALVM_BASE]\bin
    c. Execute the following command: gu.cmd install js
3. Download and install Apache Tomcat 10
    a. The 32-bit/64-bit Windows Service Installer offers hassle-free installation
4. Deploy mC
    a. Unzip the provided mC archive
    b. Clear the content of the following directory: [TOMCAT_BASE]/webapps/ROOT
    c. Copy the content of the unzipped mC folder into the directory above without creating any subfolders and check the resulting path: [TOMCAT_BASE]/webapps/ROOT/WEB_INF/sql.properties
5. Apply mC license
    a. mC licenses must be activated within 24 hours so if you already got a license file some days ago you have to ask for a new one
    b. Copy the provided license.txt into the WEB_INF folder of mC and check the resulting path: [TOMCAT_BASE]/webapps/ROOT/WEB_INF/license.txt
6. Start mC by starting Tomcat
    a. Open the service manager at [TOMCAT_BASE]/bin/tomcat10w.exe
    b. Under tab "Java" and input "Java Virtual Machine" enter GraalVM path if not already selected: [GRAALVM_BASE]\bin\server\jvm.dll
    c. Unter tab "General" click the "Start" button
7. Open a browser and call http://localhost:8080
    a. If the login page of mC does not load check the most recent logfiles of Apache Tomcat under [TOMCAT_BASE]/logs
8. Login with the username of your mC license user and "admin" as initial password
    a. If login is not possible check the most recent logfile of mC under [TOMCAT_BASE]/webapps/ROOT/logfiles/server_0.log
    b. If the mC logfiles folder is empty (mC could not be started at all) check the Apache Tomcat logfiles (see above)

Data Flow

The following chart is not project specific but contains typical elements of a system running mC: