# Installing Automic Automation v21 Containerized Agent

**How to build and run a container hosting an Automic Windows Agent**

**Version 1.1**

# Contents

# Chapter 1: Introduction

This document enables you to set up and configure an Automic Automation Windows agent running in a Docker or Kubernetes environment.

## 1.1 Prerequisites

- Automic Automation V21 (minimum 21.0.4), either standard deployment or K8s deployment
- Administrative access to Automic Automation and K8s cluster

.

# Chapter 2: Build Time

You must provide all the resources the agent needs during runtime at build time. This includes the following at a minimum:

- Docker base image
- Agent binaries (minimum 21.0.4)
- Service Manager binaries (minimum 21.0.4)
- Microsoft Visual C++ Redistributable (see https://docs.microsoft.com/en-us/cpp/windows/latest-supported-vc-redist)

An image should be generic so you can reuse it for several instances. Therefore, we recommend adding only resources that do not change for different instances. It should not include any instance-specific configuration.

Please see the provided Docker file (Dockerfile) for the agent.

```
[1]    FROM mcr.microsoft.com/windows/server:ltsc2022
[2]
[3]    WORKDIR "C:\\"
[4]    COPY vc_redist.x64.exe .
[5]    RUN vc_redist.x64.exe /quiet
[6]    RUN del vc_redist.x64.exe
[7]
[8]    RUN net user Administrator Passw0rd /active:yes
[9]
[10]   RUN mkdir “C:\\agent-windows”
[11]   WORKDIR "C:\\agent-windows"
[12]   COPY Automation.Engine_Agent_Windows.zip .
[13]   COPY Automation.Engine_ServiceManager.zip .
[14]   COPY run.ps1 .
[15]
[16]   ENTRYPOINT [ "powershell.exe" ]
[17]   CMD [ "C:\\agent-windows\\run.ps1" ]
```

In line 1, the Windows Server 2022 Docker base image is used for the Agent’s image. The Microsoft Visual C++ Redistributable is copied into the container and installed (lines 3 – 6). Then the Administrator account is enabled. This OS-user should be used to execute the Automation Engine’s JOBS or perform file transfers. A directory is created, and the Agent’s binaries and Service Manager’s binaries as zip files are copied into that directory (lines 10-13). In line 14, the prepared run script is copied into the container. The last lines specify the script executed when the Docker image is started using Powershell.

This script (_build.sh) creates the Docker image

```
[1]     docker build --no-cache . -t agent-windows:21.0
```

# Chapter 3: Runtime

At runtime custom configuration for a specific container (instance of an image) have to be set.

## 3.1 Agent Configuration File

The file named env.file holds the various settings for the agent.

```
[1]    automic_global_name=Windows-Docker-V210
[2]    automic_global_system=AUTOMIC
[3]    automic_tcpip_connection=myAEServer.local:8443
[4]    automic_variables_uc_ex_ip_addr=1.2.3.4
[5]    automic_variables_uc_ex_ip_port=21000
[6]    automic_authorization_keypassword=changeit
[7]    automic_variables_uc_ex_job_md_ip_addr=localhost
[8]    agent_liveness_check_intervall=3
```

The agent's name is specified in line 1 and must match Automation Engine's naming conventions. Line 3 contains the external jcp-ws endpoint for when the agent connects to an external Automic Automation instance. When deploying the agent to the same cluster where Automic Automation is running the internal jcp-ws endpoint can be configured. Line 4 and 5 specify the Agent's external IP-address and port which is used to contact the Agent when a file transfer is initiated. In Docker or Kubernetes environment the variable automic_variables_uc_ex_job_md_ip_addr must be set to localhost (line 7). The value specified for agent_liveness_check_intervall (line 8) determines in which interval it is checked that the agent is running inside the container.

## 3.2 Agent Container Start

This script (_start.bat) starts the Docker image by passing the proper parameters for mapping the folder into the container.

```
docker run -it ^
  --env-file env.file ^
  -p 21000:2300 ^
  -v "%cd%\trustedcert":"c:\agent-windows\trustedcert" ^
  -v "%cd%\security":"c:\agent-windows\security" ^
  agent-windows:21.0
```

## Agent Configuration

The Agent configuration consists of several areas.

- INI file configuration
  The most obvious configurations have to be done in the INI file. It is recommended to do this with ENV variables which are set in a run.ps1 (see below) to the corresponding settings in the INI file.
  An alternative is to provide the INI file with a mapped folder which is passed at the start of an agent.
- JCP certificate
  The agent has to trust the Automation Engine's JCP. If it's not a public trusted certificate the certificate has to be added to the trustedCertFolder.
- Agent certificate
  An agent authenticates against the Automation Engine with its security related files such as private keys, signed certificates and root certificates and stores it in the security folder. This folder has to be preserved in a mapped folder. Otherwise, the certificate (and private key) is lost and the Automation Engine would reject the agent because it thinks it's a different one.
  Not recommended workaround: Reset agent's key before start (see section below)

## The script run.ps1

```
function unpackZipFile($sourceArchive, $destinationFolder, $subFolder) {
    Expand-Archive -path $sourceArchive -destinationPath TMP_FOLD
    New-Item $destinationFolder -itemType "directory"
    Move-Item .\TMP_FOLD\$subFolder\* $destinationFolder
    Remove-Item .\TMP_FOLD\ -recurse
}

function replaceInFile($filePath, $replacements) {
    $result = Get-Content $filePath
    foreach ($replacement in $replacements.getEnumerator()) {
          $result = $result -replace $replacement.Name, $replacement.Value
    }
    Set-Content -Path $filePath -Value $result
}

function prepareAgent {
    $agentBasePath = ".\Automic\Agent\windows"
    Write-Output "Unpacking Agent"
    unpackZipFile -sourceArchive .\Automation.Engine_Agent_Windows.zip -destinationFolder "$agentBasePath\bin" -subFolder Agents\windows\x64

    New-Item "$agentBasePath\temp" -ItemType "directory"

    Write-Output "Adapting Agent's ini file"
    replaceInFile -filePath "$agentBasePath\bin\UCXJWX6.ini" -replacements @{
          "^NAME=.*"                    = "NAME=$env:automic_global_name";
          "^SYSTEM=.*"                  = "SYSTEM=$env:automic_global_system";
          "^connection=.*"              = "connection=$env:automic_tcpip_connection";
          "^trustedCertFolder=.*"       = "trustedCertFolder=..\..\..\..\trustedcert";
          "^agentSecurityFolder=.*"     = "agentSecurityFolder=..\..\..\..\security";
          "^keyPassword=.*"             = "keyPassword=$env:automic_authorization_keypassword";
          "^;UC_EX_IP_ADDR=.*"          = "UC_EX_IP_ADDR=$env:automic_variables_uc_ex_ip_addr";
          "^;UC_EX_IP_PORT=.*"          = "UC_EX_IP_PORT=$env:automic_variables_uc_ex_ip_port";
          "^;UC_EX_JOB_MD_IP_ADDR=.*"   = "UC_EX_JOB_MD_IP_ADDR=$env:automic_variables_uc_ex_job_md_ip_addr";
    }
}

function installAndStartServiceManager {
    $smgrBasePath = ".\Automic\ServiceManager"
    $smgrWorkdir = "$smgrBasePath\bin"

    Write-Output "Unpacking Service Manager"
    unpackZipFile -sourceArchive .\Automation.Engine_ServiceManager.zip -destinationFolder "$smgrWorkdir" -subFolder ServiceManager\windows\x64\
    New-Item "$smgrBasePath\temp" -ItemType "directory"

    Write-Output "Create smc and smd file"
    Set-Content -path "$smgrWorkdir\AUTOMIC.smd" -value "DEFINE Agent;*OWN\..\..\Agent\windows\bin\UCXJWX6.exe;*OWN\..\..\Agent\windows\bin"
    Set-Content -path "$smgrWorkdir\AUTOMIC.smc" -value "WAIT 0`nCREATE Agent"

```

```
[49]        Write-Output "Adapting Service Manager's ini file"
[50]        replaceInFile -filePath "$smgrWorkdir\ucybsmgr.ini" -replacements @{
[51]              "^\[Destination.*"      = "[Destination AUTOMIC]";
[52]              "^cmdfile.*"            = "cmdfile=*OWN\AUTOMIC.smc";
[53]              "^deffile.*"            = "deffile=*OWN\AUTOMIC.smd";
[54]        }
[55]
[56]        Write-Output "Installing Service Manager"
[57]        $smgrExe = "$smgrWorkdir\ucybsmgr.exe"
[58]        Start-Process -FilePath $smgrExe -ArgumentList "-install AUTOMIC" -
   WorkingDirectory $smgrWorkdir -Wait -Verb RunAs
[59]
[60]        $serviceName = "UC4.ServiceManager.AUTOMIC"
[61]
[62]        Write-Output "Starting Service Manager"
[63]        Restart-Service -Name $serviceName -Force
[64]        Start-Sleep -Milliseconds 15000
[65]        $agentProcess = (get-process | Where-Object {$_.ProcessName -eq 'ucxjwx6'} |
   Select-Object Id, ProcessName)
[66]
[67]        if ($null -ne $agentProcess) {
[68]              Write-Output "Agent started:  $agentProcess"
[69]              $agentPID = $agentProcess.id
[70]              do {
[71]                    $processStillActive = get-process | Where-Object {$_.ID -eq
   $agentPID}
[72]                    Start-Sleep -Seconds $env:agent_liveness_check_intervall
[73]              } while ($null -ne $processStillActive)
[74]              Write-Output "Agent process shut down"
[75]        } else {
[76]              Write-Output "Failed to start agent - check log files for more
   information"
[77]        }
[78]    }
[79]
[80]    #### Main ####
[81]    prepareAgent
[82]    installAndStartServiceManager
```

This script is executed when the container is started. In function prepareAgent the Agent’s files are unpacked and the parameters in the Agent’s configuration-file are set in lines 25 – 33. Similar steps in function installAndStartServiceManager are done to configure Service Manager (lines 41 – 54). Then the service is installed and started (lines 56 – 63). The loop in lines 70 – 73 checks if the agent is still running.

## Network Configuration

It has to be assured that needed hostnames are resolvable inside the cluster. Either by a configured DNS or services with ExternalName. The setting for UC_EX_JOB_MD_IP_ADDR must be localhost.

## Reset Agent's Key

When you don't want to store the public/private key in a volume for any reason, you have to reset the agent key in the Automation Engine before the agent can connect again. This can be done with the REST-API before starting the agent.

Example:

```
curl -f --request POST -u "$CLIENT0_USER/$CLIENT0_DEPARTMENT:$CLIENT0_PASSWORD" \
"http://$REST_CONNECTION/ae/api/v1/0/system/agents/$AGENT_NAME/resetpublickey" -v
```

Credentials for a user in client 0 must be provided in addition to the agent's name and Automation Engine's REST-endpoint.

# Chapter 4: General Advice

## 4.1 Service Manager

As a standard Service Manager is installed the Agent can be stopped using AWI but there is no way to start the Container of the Agent within AWI Administration screens. Nevertheless a proper JOBS that is executed on the Docker host can be used to start Container Agent.

Starting and stopping of the agent can also be managed by a native container orchestration tool.

## 4.2 CAU – Centralized Agent Upgrade

Even so Service Manager is installed it's not possible to use CAU properly because after a container restarts the agent would use the agent image again anyway.

Hint: It would be possible to download the latest agent package from the REST-API at every start of the container (see snipped below). This might be suitable for some installations, but could cause huge load depending on the amount of container starts.

```
echo "Create Agent package"
    URL="$REST_ENDPOINT/ae/api/v1/$CLIENT/system/agentpacks"

    BODY="{\"platform\" : \"UNIX\", \"operating_system\" : \"Linux\",
\"operating_system_architecture\" : \"x64\", \"name\" : \"$AGENT_NAME\" }"
    LOCATION=$(curl --header "Content-Type: application/json" \
    --request POST \
    --data "$BODY" \
    --dump-header - \
    --user "$AE_USER_PW" \
    -k \
    "$URL" | grep -i "location" | cut -d" " -f2)

    ZIP_URL="$(sed -e 's/[[:space:]]*$//' <<<${LOCATION})"
    echo "Download $ZIP_URL"
    curl --user "$AE_USER_PW" -k "$ZIP_URL" --output "./tmp.zip"
```

## 4.3 Starting and Stopping Agents

Any Agent that runs on top of Linux (or Windows) can be started (or stopped) by using the proper administrative view in the Automic Web Interface (AWI).

When using Docker to execute containers, the situation is different. So whenever the Agent is stopped from the Automation Engine (AWI), this also ends the execution of the Agent's container. To start the Agent again proper Docker commands must be used.

BROADCOM®