Critical manufacturing
an ASM PT company

Landscape Design

12.0.1

November 2026

DOCUMENT ACCESS

Public

DISCLAIMER

The contents of this document are under copyright of Critical Manufacturing S.A. it is released on condition that it shall not be copied in whole, in part or otherwise reproduced (whether by photographic, or any other method) and the contents therefore shall not be divulged to any person other than that of the addressee (save to other authorized offices of his organization having need to know such contents, for the purpose for which disclosure is made) without prior written consent of submitting company.

Integration Landscape Design

Copy Page

Before you build the first workflow of a plant, decide how the equipment maps to **Automation Controllers**, instances, and **Automation Managers**. These decisions are expensive to change later, because every instance, driver definition, and Setup workflow is built on them. This page compares the three controller topologies, what each one costs when the integration runs and changes, and where to place each **Automation Manager**. It is written for the solution architect or project lead who designs the landscape. For the daily operation of what you decide here, see Connect IoT Runbook.

What Runs for Each Instance

The following run-time facts decide what each topology costs:

- An **Automation Controller** is a definition: its workflows, the task libraries they use, and its drivers. It runs only as an **Automation Controller Instance**, connected to one entity, such as a **Resource**. The **Automation Manager** that you select in the Connect wizard runs the instance and all its driver instances. See Connect Automation Manager.
- For each instance, the **Automation Manager** runs one controller process and one driver process per driver of the controller, next to its own manager and monitor processes. See Deployment.
- All workflows of a controller run in its controller process. A restart of the instance restarts every workflow and every Setup chain, and loses the values stored as `Temporary`.
- Each instance keeps its own data store. Two instances of one controller do not share stored values.
- An entity that is connected to an active instance, or to a driver of another active instance, cannot be connected again.
- Each driver instance is connected to an entity of the entity type of its **Automation Driver Definition**, which the Connect wizard asks for separately from the entity of the instance.
- A running instance has one driver instance for each driver that its controller had when the instance was created. A driver added later reaches the instance through a new controller version and **Update** of the stopped instance, which

adds its driver instance, or through a replacement of the instance, which gives it a new name and ID. The **Update Controller Version** action of an **Automation Scheduled Action** fails when the target version has a different number of drivers than the instance.

- Work on two drivers meets inside one controller: in one workflow, or through a function, a subworkflow, the data store, or a queue. Two controllers exchange data through MES or the message bus.

> **Info**
>
> For what each change in MES reaches and when a restart is needed, see Connect IoT Runbook.

Three Controller Topologies

One Controller per Machine

Each machine has its own **Automation Controller**, with the drivers of that machine, and one instance on the entity of the machine. Use it when each machine needs logic of its own. Every controller is a separate definition, so a fix in logic that several machines share must be applied, versioned, and rolled out in each controller. To share that logic, see the reuse options on the Developer Portal and Extend a Controller ↗.

One Controller per Line with Several Drivers

One **Automation Controller** holds one driver for each piece of equipment of a line, or for each connection. One instance runs on the entity of the line, and each driver instance is connected to the entity of its equipment. Use it when the work of one piece of equipment depends on the data of another, for example a barcode that a scanner reads and a tracking system must receive, because the two drivers then meet inside one workflow.

The line runs and changes as one unit:

- A restart, an update, or a move to another manager stops every piece of equipment of the line.
- Adding equipment adds a driver, so the instance must be replaced.
- The entities of the equipment are taken by the driver instances, so they cannot carry instances of their own.

One Controller on Many Machines

One **Automation Controller** runs as one instance per machine, and each instance is connected to the entity of its machine. The workflows name no machine. What differs between machines, such as a folder, an address, or a port, is kept in MES per entity, in a **Smart Table** row or an entity attribute, and the Setup workflow reads it at `OnInitialize` before **Equipment Configuration**. See Entity Instance and Resolve Table.

Use it for machines of one kind that differ only in their settings. Adding a machine is a new row or attribute and a new instance, with no change to the workflows. A fix is made once and rolled out to each instance, one machine at a time.

What Each Topology Costs at Run Time

Concern	One controller per machine	One controller per line	One controller on many machines
What a restart stops	One machine	Every machine of the line	One machine
Adding a machine	A new controller and an instance	A new driver, then the instance is updated or replaced	A row or attribute in MES and a new instance
A fix in shared logic	Applied in each controller	Applied once	Applied once
Rolling out a new controller version	Each controller and its instance	One instance, all equipment at once	Each instance, one machine at a time
Data between pieces of equipment	Through MES or the message bus	Inside the controller	Through MES or the message bus
Per-machine values	In MES, or in the controller	In the Setup chain of each driver	In MES, per entity

Table: What each controller topology costs at run time

An instance runs one version of its controller. A change to the workflows of that version reaches the instance at once. A new version reaches it only when the instance is updated with **Update** on the **Connect IoT** page, or by an **Automation Scheduled Action**. See Connect IoT and What Is Automation Scheduled Action?.

Values stored as `Persistent` belong to the instance ID. They stay with an instance that you update, and an instance that **Change Automation Manager** replaces does not take them along.

Update, **Restart**, **Update Controller Version**, and **Change Automation Manager** stop the instance for about 1 to 20 seconds.

Placing Automation Managers

An **Automation Manager** runs as a Windows Service, as a console application, or as a container. See Deployment. Where you place each manager depends on the drivers of the instances it runs, the network, and the files and devices the drivers need:

- **Drivers that cannot be containerized** - OIB and OPC DA run only on a Windows host. Instances that use them need a manager installed as a Windows Service.
- **Drivers that read a local device** - Bluetooth, Keyboard Wedge, and Serial need the device on the computer that runs the manager, or forwarded to its container. Keyboard Wedge runs only on Linux.
- **File drivers** - the folders must be accessible, mapped, and authenticated in the operating system of the manager. In a container, each folder needs a volume.
- **Drivers that the equipment connects to** - REST Server, Fuji Nexim, PanaCIM, and TCP/IP or SECS/GEM in Passive mode listen on a port that must be open on the manager. On Kubernetes, add a route or expose the port, see Configure Automation Manager Deployment. The DevOps Center does not set inbound ports through its GUI.
- **The package repository** - each site needs one local package repository that both MES and the managers can reach. A manager can also get its packages through MES, without direct access to the repository.
- **Persistency** - each manager stores the persisted data of the instances it runs under its own persistency path. When a manager can be replaced by another, keep that path on shared storage. For high availability as a Windows Service, use Windows Server Failover Clustering. On Kubernetes, a manager runs as one replica. See Deployment.

For the requirements of each driver, see Connect IoT Requirements. To move an instance to another manager when the equipment is idle, use the **Change Automation Manager** action of an **Automation Scheduled Action**. See Actions.

No fixed figures size a manager. The memory and CPU of each controller and driver process, and the number of instances that one manager can run, depend on how the integration is used: the drivers, the number of equipment, the rate of events and data, and the work of the workflows. Test the instances in conditions as similar as possible to production, and confirm that the performance meets the non-functional requirements before you decide how many instances each manager runs and when to add a manager.

Decision Table

If	Choose
Machines of one kind differ only in their settings	One controller on many machines, with the settings in MES per entity
Machines differ in their logic	A controller for each kind of machine, sharing common logic through a parent controller or a shared workflow
The work of one piece of equipment needs the data of another, at the same moment	One controller with several drivers
Each piece of equipment must restart, update, or move to another manager on its own	One instance per piece of equipment
Equipment is added or removed often	One controller on many machines, so that a new machine needs no driver and no replaced instance
An instance uses OIB or OPC DA	A manager installed as a Windows Service on a Windows host
A driver reads a local device or a local folder	A manager on that computer, or a container with the device or folder forwarded
The equipment connects to the driver	A manager whose port the equipment can reach

If	Choose
A manager must survive the loss of its host	Shared storage for its persistency path, and a failover option from Deployment

Table: Choosing a topology and a manager placement

Legal Information

Disclaimer

The information contained in this document represents the current view of Critical Manufacturing on the issues discussed as of the date of publication. Because Critical Manufacturing must respond to changing market conditions, it should not be interpreted to be a commitment on the part of Critical Manufacturing, and Critical Manufacturing cannot guarantee the accuracy of any information presented after the date of publication. This document is for informational purposes only.

Critical Manufacturing makes no warranties, express, implied or statutory, as to the information herein contained.

Confidentiality Notice

All materials and information included herein are being provided by Critical Manufacturing to its Customer solely for Customer internal use for its business purposes. Critical Manufacturing retains all rights, titles, interests in and copyrights to the materials and information herein. The materials and information contained herein constitute confidential information of Critical Manufacturing and the Customer must not disclose or transfer by any means any of these materials or information, whether total or partial, to any third party without the prior explicit consent by Critical Manufacturing.

Copyright Information

All title and copyrights in and to the Software (including but not limited to any source code, binaries, designs, specifications, models, documents, layouts, images, photographs, animations, video, audio, music, text incorporated into the Software), the accompanying printed materials, and any copies of the Software, and any trademarks or service marks of Critical Manufacturing are owned by Critical Manufacturing unless explicitly stated otherwise. All title and intellectual property rights in and to the content that may be accessed through use of the Software is the property of the respective content owner and is protected by applicable copyright or other intellectual property laws and treaties.

Trademark Information

Critical Manufacturing is a registered trademark of Critical Manufacturing.

All other trademarks are property of their respective owners.

Copyright © 2023 Critical Manufacturing. All rights reserved.