Content

Version 210825

Manual

Document: Manual - FORCAM FORCE EDGE.docx

Release date: 2023-09-08

Document version: 2

Author: AEgilmez, STernes

Content

1 Concept

FORCAM FORCE EDGE offers manufacturing companies a solution for digitally connecting their heterogeneous machinery. Almost all machines can be digitized with FORCAM FORCE EDGE, regardless of age or technical status. Thus, FORCAM supports the digital transformation of a manufacturing plant in the Brownfield environment.

FORCAM therefore delivers a product that solves the main requirement of Industry 4.0 by extracting digital information from the production machinery. This makes a significant contribution to the digital transformation by closing the gap between IT (information technology) and OT (operational technology).

FORCAM FORCE EDGE interconnects the various machine connections and signals and delivers them as standardized events to superordinate systems. These can be ME or MOM systems such as SAP DMC/ME or MII, among others. FORCAM can thus reduce the time and effort required for digitization and create a standardized interface to the machine park. The machines are connected via an innovative plugin concept for easier future expansion. All common machine manufacturer-specific (proprietary) protocols are presently supported (such as HEIDENHAIN, Siemens S7 or FANUC & Co.) as well as all common communication standards (such as MTConnect, OPC-UA or MQTT). The FORCAM I/O Controller is available as separate hardware for digitizing the machine if the machine is not network-capable. FORCAM FORCE EDGE is continually expanded with plugins in order to meet the challenge of digitally mapping every type of machine via the EDGE solution.

The machine connections are used to obtain a wide variety of information. This includes information about the current status of the connected machines or their sensor readings, such as temperatures, pressures or energy consumption. Particularly in the Brownfield environment, it is important not only to pick up the signals and transmit them, but also to interpret them for use. This task is performed by the EDGE Composition Layer. For instance, the interpretation of when a machine is actually in production or at stoppage is fundamentally important. FORCAM FORCE EDGE is not only able to read or write machine signals. Another essential part of the solution is the handling of NC programs and the ability to transfer them to and from the machine.

FORCAM FORCE EDGE's modern, cleanly structured menu navigation makes it quick and efficient to digitally connect machines using the available control and signal information.

The Machine Repository component makes it easy to create and use machine templates. This means that templates can be defined for machine connections or derived from existing connections and used for the connection of the same machine types. This further reduces the individual effort required to connect a machine, which saves time and resources for the digitization project. The template structure ensures a standardized connection of identical machines, thus enabling the comparison of machines of the same type.
FORCAM provides standard templates for common machines.

FORCAM FORCE EDGE is flexible and can be applied to any manufacturing company. The individual components of the solution can be located in different areas and levels and provide benefits at each level.

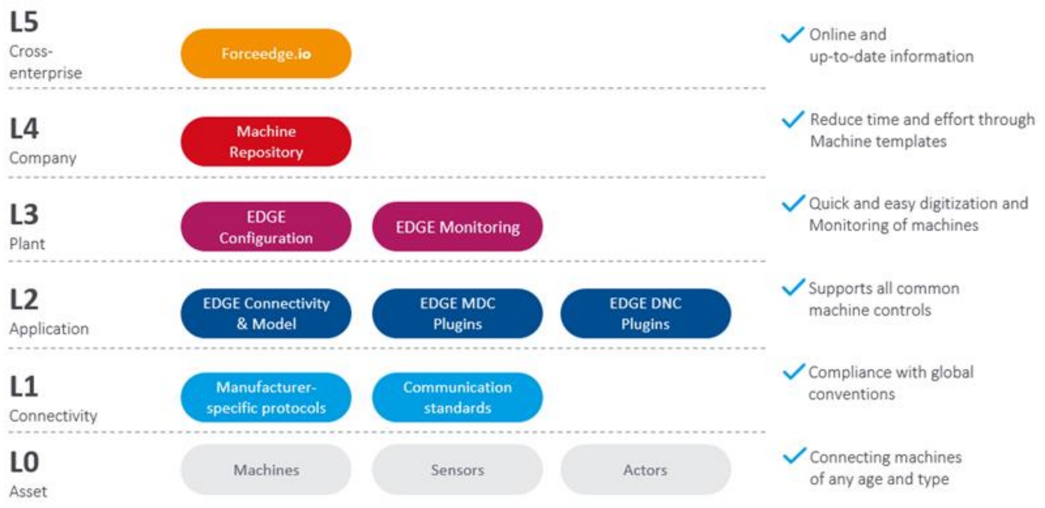

L5
Cross-enterprise
Forceedge.io
Online and up-to-date information
L4
Company
Machine Repository
Reduce time and effort through Machine templates
L3
Plant
EDGE Configuration
EDGE Monitoring
Quick and easy digitization and Monitoring of machines
L2
Application
EDGE Connectivity & Model
EDGE MDC Plugins
EDGE DNC Plugins
Supports all common machine controls
L1
Connectivity
Manufacturer-specific protocolls
Communication standards
Compliance with global conventions
L0
Asset
Machines
Sensors
Actors
Connecting machines of any age and type

Fig. 1: Location of the FORCAM FORCE EDGE solution components

The following figure shows the reference architecture of the Open Industry 4.0 Alliance, which is also the basis of the FORCAM FORCE EDGE architecture. FORCAM contributes significantly to digitalization in industry and focuses on customer benefits. The connectivity of hardware through intuitive and user-friendly software is what makes FORCAM FORCE EDGE stand out.

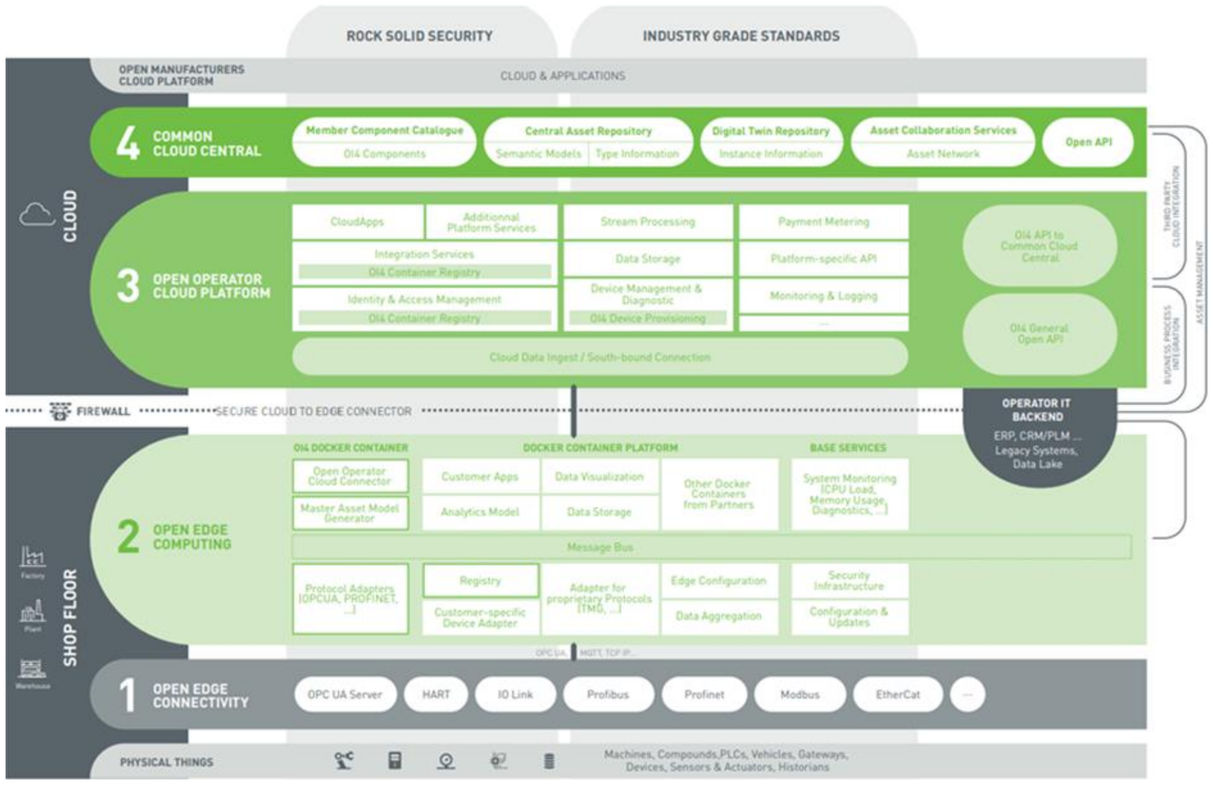


Fig. 2: Reference architecture of the Open Industry 4.0 Alliance

2 System Components

This chapter explains the individual components of FORCAM FORCE EDGE and their functions.

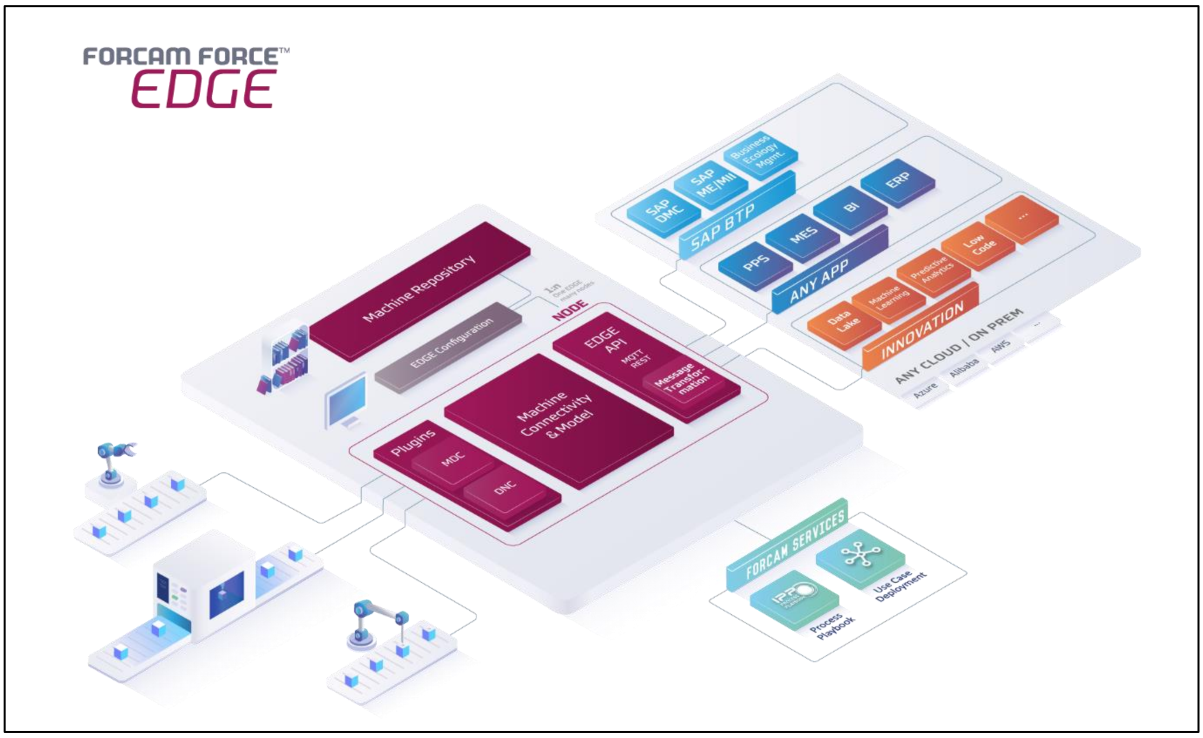


Fig. 3: Schematic structure of FORCAM FORCE EDGE

2.1 Machine Connectivity & Model

FORCAM FORCE EDGE's central system component **Machine Connectivity & Model** is the core element of machine connectivity and contains the following relevant subcomponents:

EDGE MDC Layer

The EDGE MDC layer manages the actual connection of the machines. The essential elements are the selection of the suitable plugin for the communication with the machine control, the configuration of the machine master data, the setting of the network connection and the definition of the machine signals. In addition, the EDGE MDC Layer forwards machine signals to the EDGE Composition Layer.

EDGE DNC Layer

The EDGE DNC Layer manages the actual connection of machines with an NC supply. The essential elements here are the selection of the suitable plugin for communication with the machine control, the configuration of the machine master data, the setting of the network connection and the configuration of the DNC supply.

EDGE Composition Layer

The EDGE Composition Layer enables deriving logical machine states. Using a simple script language, status events can be derived from signal combinations. This is an important factor for success in the Brownfield environment, since older machines can be operated, which would otherwise not provide any usable information. For this purpose, the standardization of the report capability is ensured by the composition layer. This is done via a scripting language. In addition, options for individual events are also provided. The scripts also make it possible to react to events and write values to the machine's control unit.

2.2 Plugins

Plugins in the FORCAM FORCE EDGE environment are used to establish communication links with specific machine controllers. They allow direct communication with various machine controllers, but also cover modern communication protocols such as MQTT, UPC UA and many more. The plugin concept of FORCAM FORCE EDGE is extendible, FORCAM is continuously expanding the number of supported plugins.

The plugins are divided into those for Machine Data Collection (MDC) and for Distributed Numerical Control (DNC).
MDC plugins include those designed for unidirectional readout of machine signals as well as for bidirectional signal transmission, i.e. for reading out and writing back signals.
DNC plugins are used for transferring and reading NC files. They are used for transferring NC programs to the machine's file system or to query the program active on the machine.

For the most common control types, a set of plugins is included in FORCAM FORCE EDGE by default. An overview of the current FORCAM plugins is listed in section 8.4.

⚠ Providing, editing or managing NC programs is not a function of FORCAM FORCE EDGE

2.3 EDGE API

The EDGE API as RESTful API is used to retrieve machine master data and for configuring the machine connections. The EDGE API event service is used to forward machine data in the form of standardized events to superordinate systems (3rd party). Superordinate systems can be connected either via HTTP/REST or MQTT. The events are transmitted via HTTP in JSON format. Optionally, an MQTT broker can be used as middleware.
The EDGE API is delivered with preconfigured standard events for communication with the MES or ERP level. If necessary, these can be further individualized.

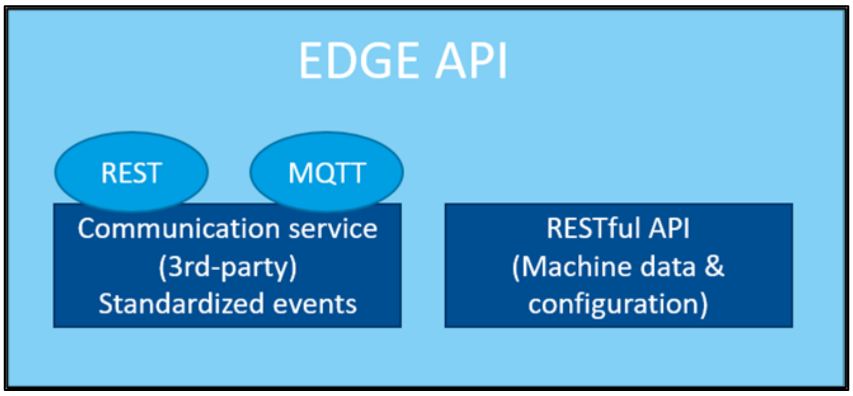

EDGE API
REST
MQTT
Communication service (3rd-party) Standardized events
RESTful API (Machine data & configuration)

Fig. 4: The BRIDGE API structure

2.4 EDGE components

2.4.1 EDGE Configuration

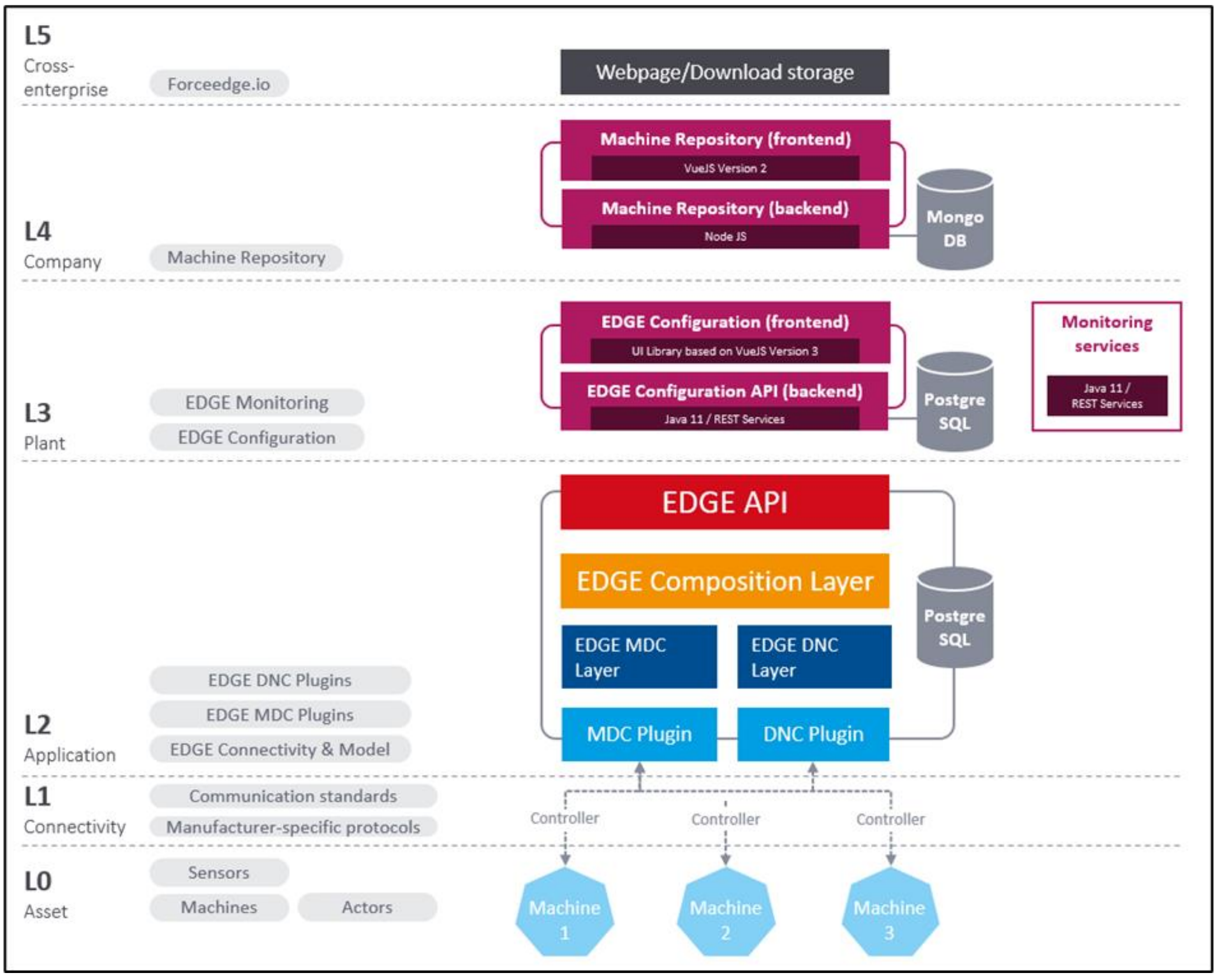
EDGE Configuration is the management interface for FORCAM FORCE EDGE. It can be used to manage multiple EDGE nodes. An EDGE node is the bundling of signal collection from several machines. Depending on the amount of data, one or more EDGE nodes are used per plant. The management of the nodes is done centrally.

2.4.2 Machine Repository

The Machine Repository lets you generate templates from existing machine connections or for new ones. These templates can be used to connect machines of the same type and the same usage type in a standardized manner. The template contains all configuration elements that are not machine-specific. Machine and machine connection-specific configuration elements are, for example, IP address, machine name, equipment number, etc.
By using an existing template, the time required to connect a machine is significantly reduced.

3 System Architecture

FORCAM FORCE EDGE is architecturally divided into levels (layers). These are based on the business use case, which enables a high scalability of the individual components. For example, multiple EDGE nodes can be hosted to divide the machines logically, but also based on performance.



Level 0 - Asset

The lowest layer is where the machines, sensors and actuators are connected. In this respect, the value added pledge is to support every machine in terms of type and age.

Level 1 - Connectivity

The growing selection of plugins facilitates the connection of a wide variety of controllers with their different communication standards such as OPC UA or MT Connect, as well as manufacturer-specific protocols.

Level 2 - Application

The number of possible edge nodes is not limited. A node encompasses several layers or tasks:

- **EDGE MDC Layer** controls the connection of the machine via **Plugins** and forwards the signals to **EDGE Composition Layer**. Similarly, DNC capable machines are connected via **EDGE DNC Layer**.
- **EDGE Composition Layer** is responsible for signal interpretation and for creating the standardized events.
- **EDGE API** is the programming interface through which machines, event management and notification of third party systems can be configured and done "northbound".

Level 3 - Plant

The configuration component can serve 1 to n EDGE nodes. This can be provided per plant, as well as per production line.
Each component is standalone and can run without a connection (for example, if temporarily down). This enables a wide variety of deployment. For example, EDGE Configuration does not necessarily have to be hosted in FORCAM FORCE EDGE itself, but simply a connection to the respective API has to be established.
All components communicate via standardized interfaces (HTTP/REST).

Level 4 - Company

The Machine Repository is a FORCAM FORCE EDGE component that lets you create and manage machine connection templates.

Level 5 - Cross-enterprise

FORCAM FORCE EDGE provides a collection of machine configuration templates freely available via forceedge.io. These can be downloaded via forceedge.io and then imported into your own machine repository.

4 Deployment

FORCAM FORCE EDGE can be purchased and installed via a classic license or directly as Software as a Service (SaaS).

Fig. 5: Options for installing FORCAM FORCE EDGE

License model

Two installers are provided for an installation using the classic license model: **ForceEdgeUI** and **ForceEdgeNode**. These are installed by the customer themselves, or by a FORCAM service provider. ForceEdgeUI contains the entire user interface including all functions and is installed first. ForceEdgeNode contains the Edge node and can be installed as often as needed, since the number of nodes in FORCAM FORCE EDGE is not limited, it only depends on the license purchased. This defines how many nodes can be created and how many controllers can be connected per node.

SAP DMC

FORCAM FORCE EDGE can be purchased as a solution extension of **SAP Digital Manufacturing Cloud** (SAP DMC) on the SAP Business Technology Platform (SAP BTP).
If purchased as SaaS, FORCAM provides the hardware on which FORCAM FORCE EDGE is run. A Microsoft Azure Stack Edge (ASE) is used for this purpose, which is preconfigured by FORCAM. To do so, FORCAM requires specific IP addresses, which must be entered manually by the customer or communicated to FORCAM's service providers. The customer is responsible for integrating the ASE into the existing hardware environment. FORCAM guarantees the function and availability of the ASE and the software components.

5 Basic settings

General settings for FORCAM FORCE EDGE are possible in the **Home** section.
In addition to the user administration and permissions, current documents can also be downloaded here. This chapter goes into **user administration, licensing** and the **download scope**.

ⓘ The settings made for language and dark mode in the profile are saved in the user profile and apply only to this user.

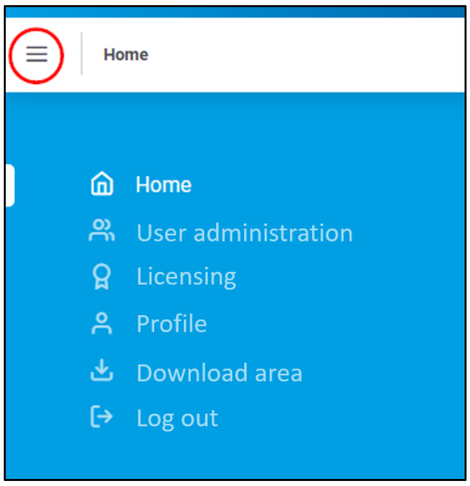
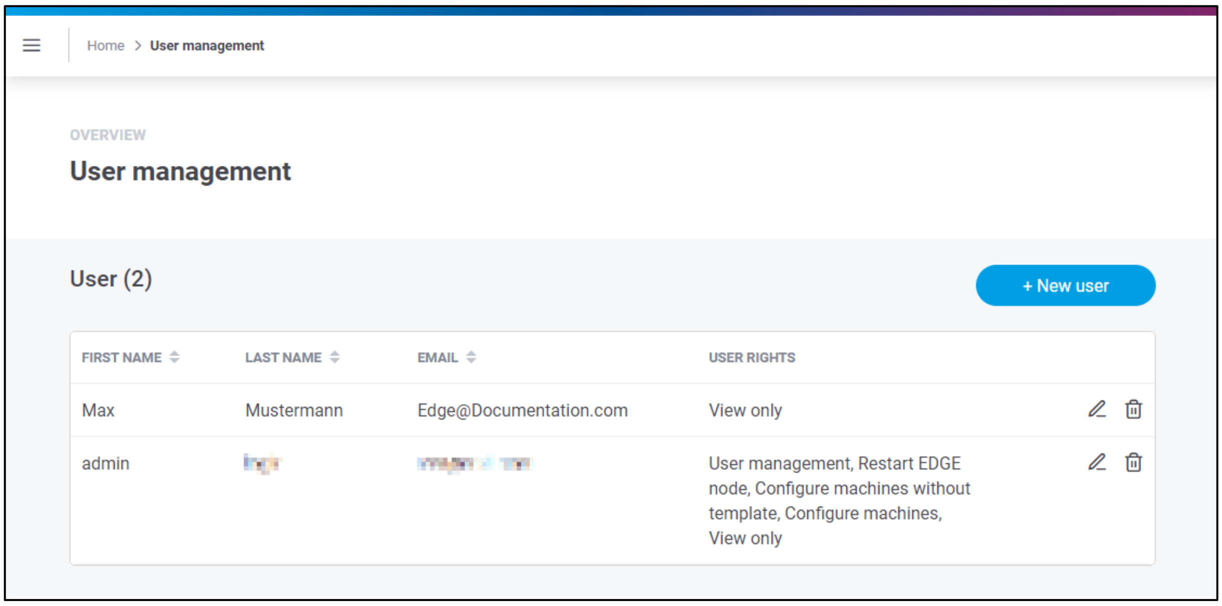


Fig. 6: Calling up the Home section

5.1 User administration

Users are created for FORCAM FORCE EDGE in the user administration. Each user can be assigned permissions containing only the functions appropriate or intended for that user (e.g. configure machine, restart node, etc.). Existing user accounts can also be edited subsequently.

ⓘ Once the permissions of a logged-in user have been changed, they take effect immediately after a new login. However, it may take up to 30 minutes for the change to take effect if the user does not log in again.



User management

User (2)

FIRST NAME	LAST NAME	EMAIL	USER RIGHTS
Max	Mustermann	Edge@Documentation.com	View only
admin	[redacted]	[redacted]	User management, Restart EDGE node, Configure machines without template, Configure machines, View only

Fig. 7: FORCAM FORCE EDGE user administration with 2 users

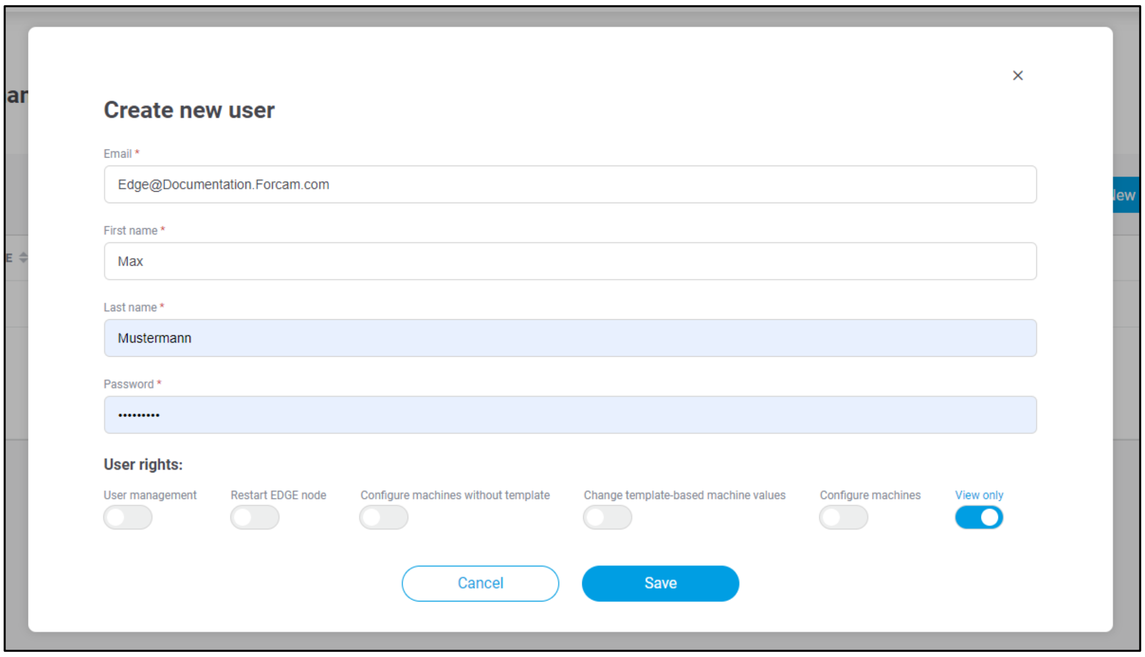
To create a new user:

1. Click on **+ New user**.
2. Enter an email address, first and last name in the subsequent dialog.
3. Set the desired password.
   This must be at least 8 characters long, consist of upper and lower case letters and contain at least one number and one special character.
   The following special characters are permitted: @$!%*?&
4. Assign user permissions (see below).
5. Save.

ⓘ A user with the same data cannot be created a second time.

Table 1: User permissions in FORCAM FORCE EDGE

User permission	Description
User administration	The user can call-up the user administration, create new users and assign/remove permissions.
Restart Edge node	The user can restart an Edge node.
Configure machines without template	The user can create machines without the use of MR templates.
Change template-based machine values	The user can modify MR templates (signals and script).
Configure machines	The user can create machines only by means of MR templates. Changes cannot be made.
View only	The user is only able to view the Edge node. Changes cannot be made.


Create new user

Email *: Edge@Documentation.Forcam.com

First name *: Max

Last name *: Mustermann

Password *: •••••••••

User rights:

User management: off; Restart EDGE node: off; Configure machines without template: off; Change template-based machine values: off; Configure machines: off; View only: on

Fig. 8: Dialog for creating a new user

5.2 Licensing

Licenses can be imported and viewed under **Licensing**.

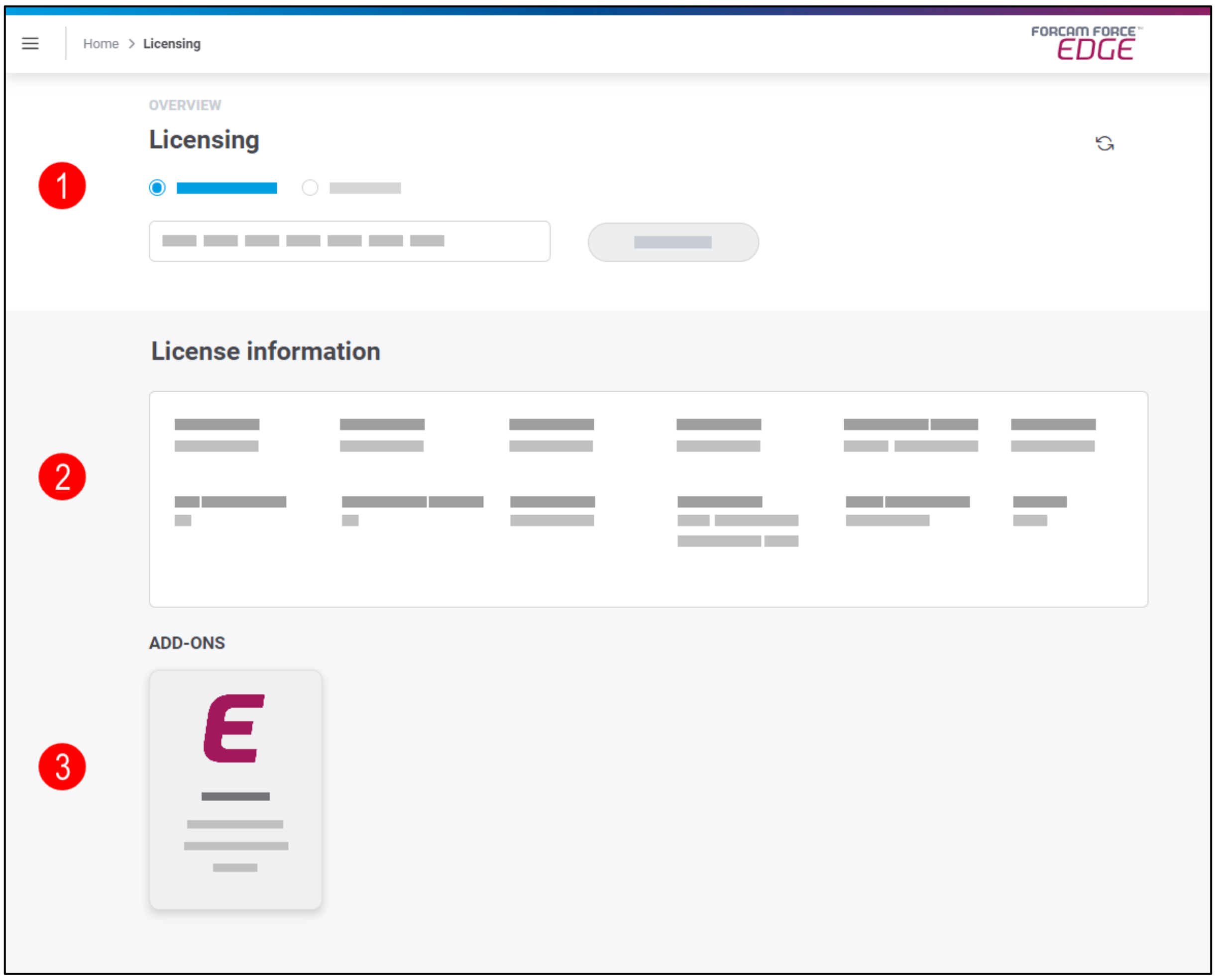


Fig. 9: Licensing and overview

(1) A new license can be uploaded as a file or entered directly as a key.
(2) License information consists of type and status of the license, number of licensed nodes and machines, maintenance, validity and other data.
(3) All booked add-ons are listed here.

5.3 Download area

The current FORCAM FORCE EDGE documentation can be downloaded in several languages via the **General** tab. The manual and a product description are available. The manual is this document with detailed configuration instructions. The product description is a shorter document describing only the function and benefits of the application and a listing of the scope of performance functions.

FORCAM provides a custom application in the tabs **MDC Plugins** and **DNC Plugins**. This application is needed to communicate with a machine via the corresponding plugin. It is located between the EDGE MDC layer and the machine and enables the bi-directional communication.

6 EDGE Configurations

The configuration of an edge node as well as a machine is done completely in the EDGE Configuration of FORCAM FORCE EDGE. The user-friendly interface will guide you through all relevant settings and shows all nodes and the statuses in the overview.

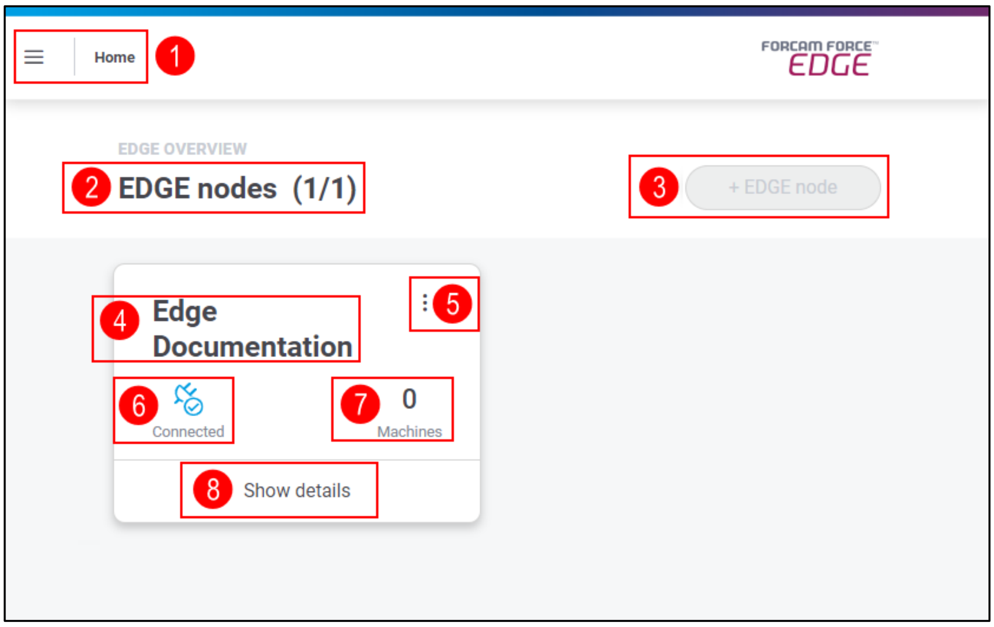


Fig. 10: FORCAM FORCE EDGE entry and overview page

(1) FORCAM FORCE EDGE settings menu
- Licensing
- User administration
- Language

(2) Shows how many edge nodes are configured (first number) and how many nodes can be configured in accordance with the license (second number)

(3) Adds a new EDGE node

(4) Node name

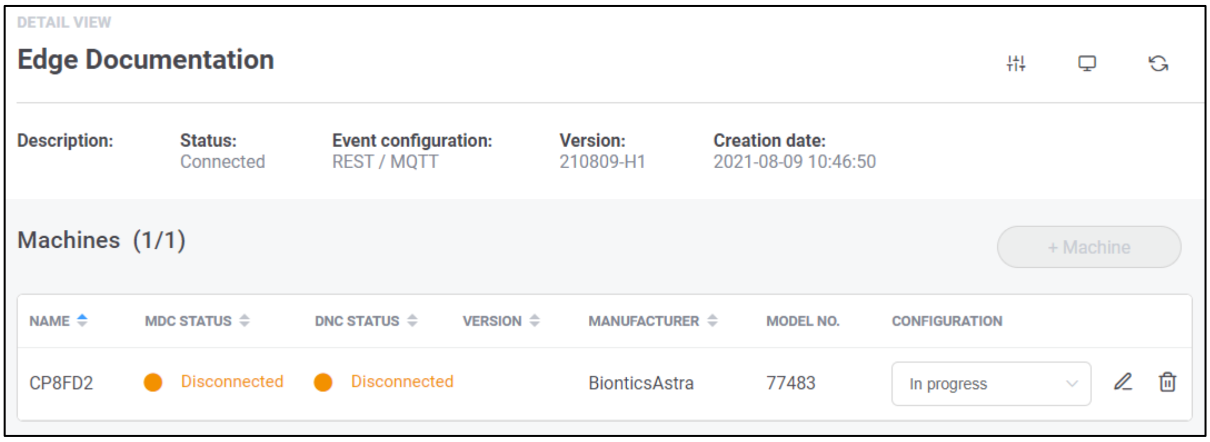
(5) Node settings menu:
- Edit
- Event configuration
- Delete

(6) Status of the connected machines

(7) Number of connected machines

(8) More detailed node information:
- List of all connected machines and their status
- Option to add a new machine
- Monitoring connected machines

ⓘ Changes to user administration regarding user rights can take up to 30 minutes to take effect throughout the system.


DETAIL VIEW

Edge Documentation

Description:	Status:	Event configuration:	Version:	Creation date:
	Connected	REST / MQTT	210809-H1	2021-08-09 10:46:50

Machines (1/1)

+ Machine

NAME	MDC STATUS	DNC STATUS	VERSION	MANUFACTURER	MODEL NO.	CONFIGURATION
CP8FD2	● Disconnected	● Disconnected		BionticsAstra	77483	In progress

Fig. 11: Machine overview as next page after clicking on "Show details"

TEMPLATE VERSION shows the current implementation (see section 4.2.1).

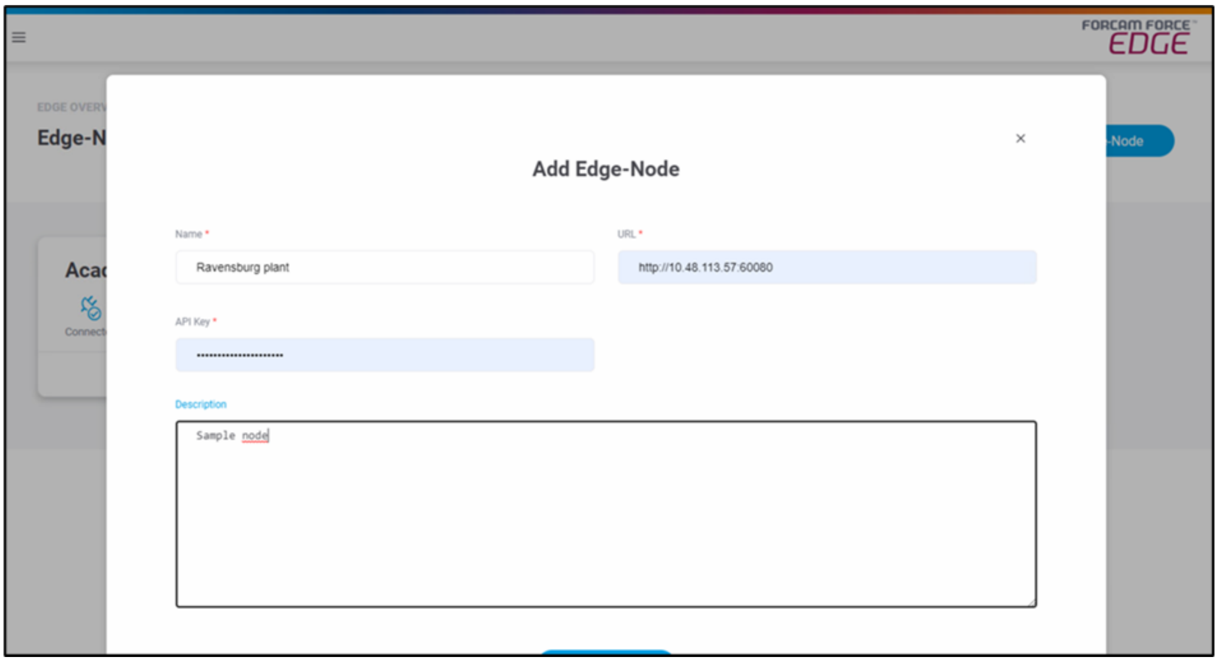
CONFIGURATION lets you manually determine which status the configuration should have, for example to give employees an overview or to call them in:

- In progress:
  The configuration is not yet complete and is to be continued at another time.
- In validation:
  The configuration of the machine is to be checked for errors and consistency.
- Completed:
  The configuration is completely finished. This is the only status in which the MR learning cycle can take place to generate a template from the configuration.

6.1 Add EDGE node

FORCAM FORCE EDGE lets you add nodes in just a few steps. An edge node corresponds to an instance of a connection variant. There can be several nodes per plant. They are logically bundled so that the machine workload is distributed practically.

ⓘ If a configured edge node is removed from the interface, its configuration is preserved. If the node is recreated under the same data, it automatically adopts the previously configured data.



Add Edge-Node

Name * Ravensburg plant

URL * http://10.48.113.57:60080

API Key * ••••••••••••••••••••••

Description

```
Sample node
```

Fig. 12: Dialog for adding a new node

To add a new EDGE node:

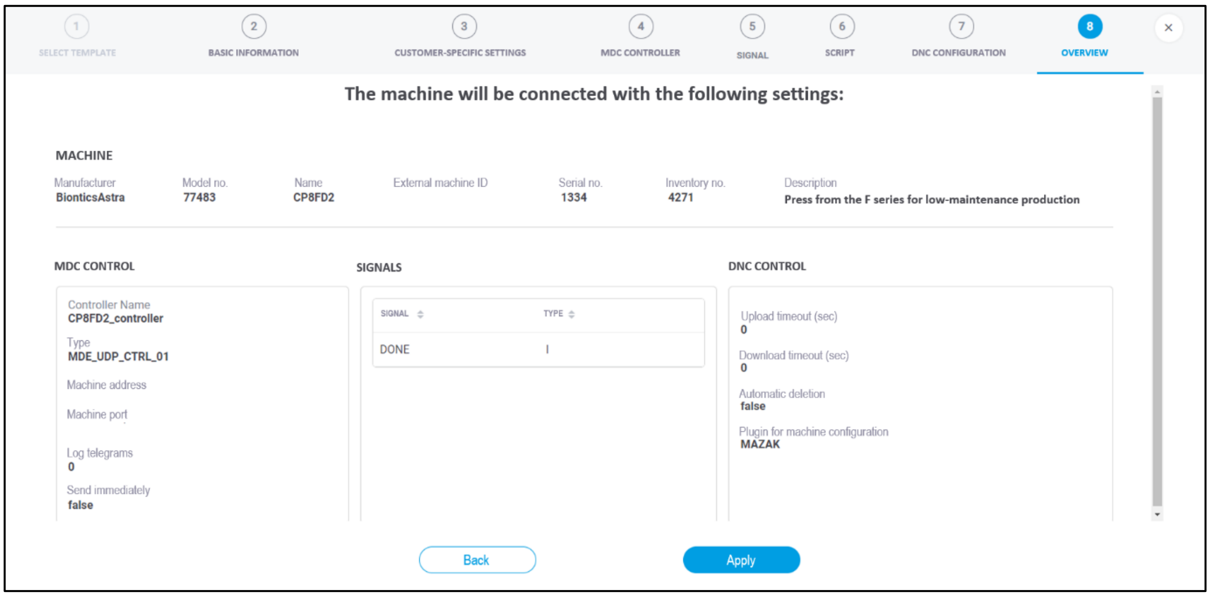
1. Click on **+ EDGE node**.
2. Fill in all mandatory fields in the next dialog:
   - Name:
     Appears in the node overview as the node title
   - URL:
     Consists of http + IP address + host. Only one edge node can be created per URL.
   - API key:
     Password that was assigned during the initial node installation
3. Optional: add description.
4. Save.

6.2 Add machine

The dialog for adding a machine guides you through eight steps necessary for a connection. This is where MDC/DNC controls are configured and machine signals are defined, among other things.

Negative values are not permitted in the machine configuration.

When a step is completed, it is highlighted in blue in the top bar.
To return to a completed step, click on it.
Each configuration page can be selected and called-up directly when editing a machine that has already been configured.



The machine will be connected with the following settings:

MACHINE

Manufacturer	Model no.	Name	External machine ID	Serial no.	Inventory no.	Description
BionticsAstra	77483	CP8FD2		1334	4271	Press from the F series for low-maintenance production

MDC CONTROL

Controller Name
CP8FD2_controller
Type
MDE_UDP_CTRL_01
Machine address

Machine port

Log telegrams
0
Send immediately
false

SIGNALS

SIGNAL	TYPE
DONE	I

DNC CONTROL

Upload timeout (sec)
0
Download timeout (sec)
0
Automatic deletion
false
Plugin for machine configuration
MAZAK

Fig. 13: Dialog for configuring a machine in FORCAM FORCE EDGE

To add a machine:

1. Click **Show details** at the desired node in the EDGE overview.
2. Click on **+ Machine** on the next page.

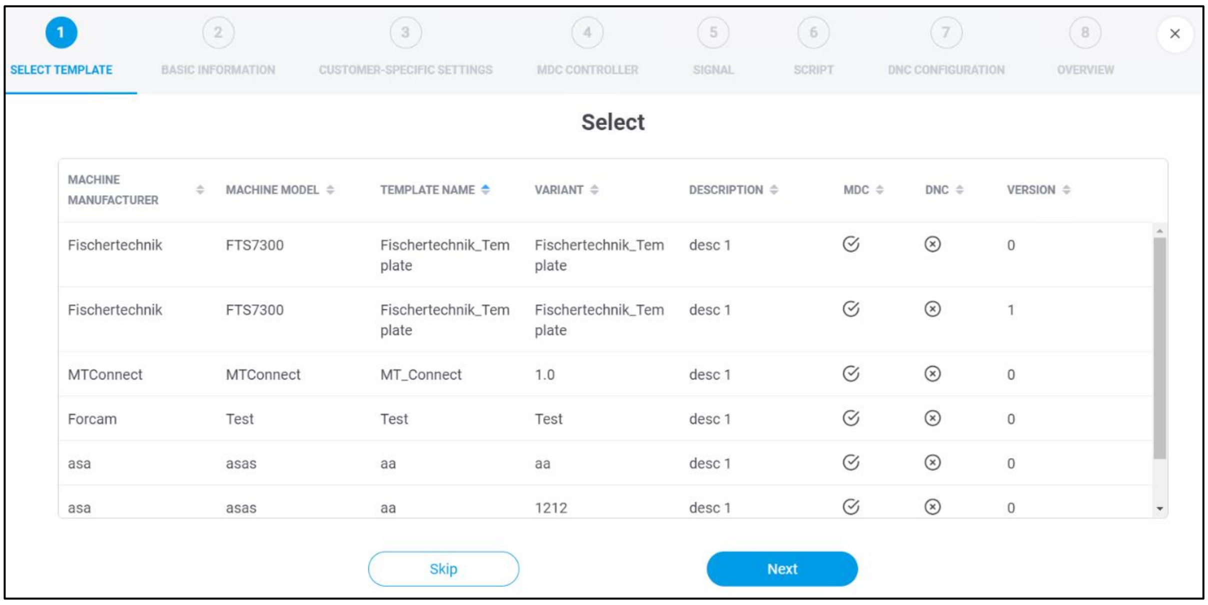
➔ The next dialog guides you through the following eight steps to configure a machine.

6.2.1 ① Select template

In FORCAM FORCE EDGE, multiple machines of the same type do not have to be completely reconfigured every time: Once a machine has been configured, it can be entered as a template in the Machine Repository and will then be offered for the next machine connection in this mask. If the template is selected at this point, all settings are automatically used for this machine and all configuration fields that are not machine-specific are pre-filled. Only machine-specific (e.g. serial number) and information specific to the connection (e.g. IP address or port of the machine or controller) must still be edited.

The template **VERSION** shows which version it is in. If a template is revised, the version number is automatically incremented, and the earlier version is overwritten. Version 0 means that no script is configured in the corresponding template.

ⓘ This step is only available if the MR component is used. If no template is configured or MR is not in use, the machine connection will start with step ②.



MACHINE MANUFACTURER	MACHINE MODEL	TEMPLATE NAME	VARIANT	DESCRIPTION	MDC	DNC	VERSION
Fischertechnik	FTS7300	Fischertechnik_Template	Fischertechnik_Template	desc 1	✓	⊗	0
Fischertechnik	FTS7300	Fischertechnik_Template	Fischertechnik_Template	desc 1	✓	⊗	1
MTConnect	MTConnect	MT_Connect	1.0	desc 1	✓	⊗	0
Forcam	Test	Test	Test	desc 1	✓	⊗	0
asa	asas	aa	aa	desc 1	✓	⊗	0
asa	asas	aa	1212	desc 1	✓	⊗	0

Fig. 14: Add machine - select template

1. Select the desired machine connection from the list.
2. Click on **Next**.

ⓘ If a template is selected during subsequent editing of a previously configured machine, it will overwrite the existing configuration after clicking **Next**.
This is why the template must be confirmed by the checkbox at the bottom.
The checkbox will only be visible if a template has been selected.

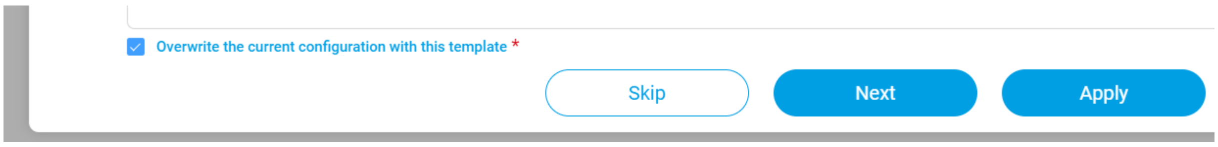
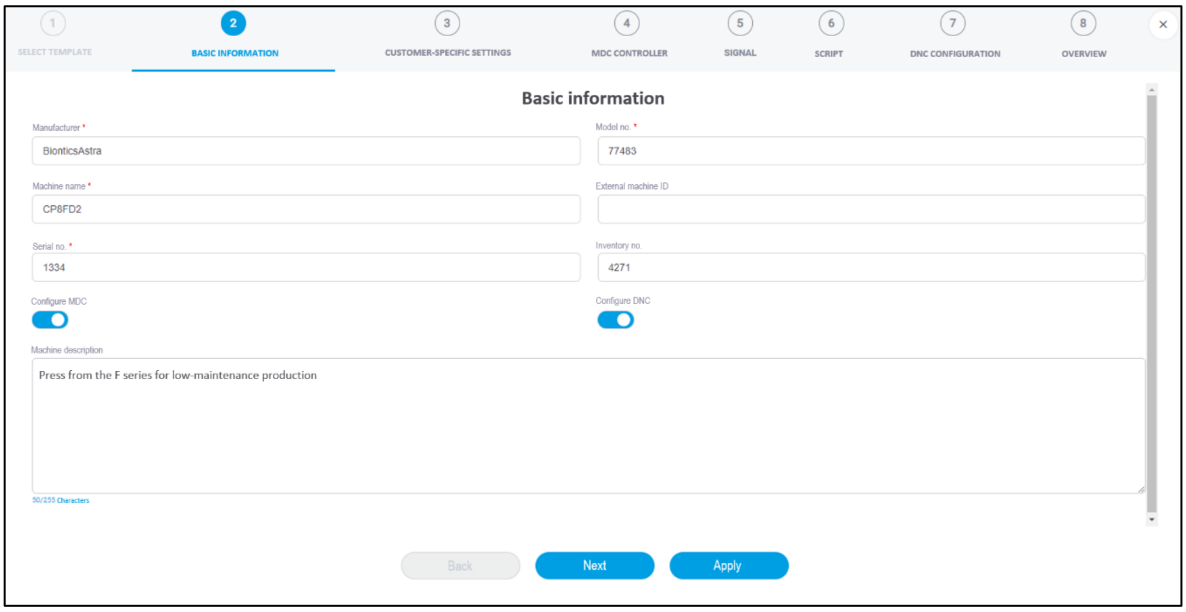


Fig. 15: Confirming the overwriting of the configuration with a template

6.2.2 ② Basic information

This is where basic information of the machine to be configured, such as name or serial number is entered. This is also where you determine whether an MDC and/or a DNC control is to be configured. With the MDC controller, signals are collected from the machine and transmitted or written to it. NC files are transferred to the machine via the DNC controller.



Basic information

Manufacturer *: BionticsAstra
Model no. *: 77483
Machine name *: CP8FD2
External machine ID:
Serial no. *: 1334
Inventory no.: 4271
Configure MDC: ●
Configure DNC: ●
Machine description: Press from the F series for low-maintenance production
50/255 Characters

Fig. 16: Add machine - basic information

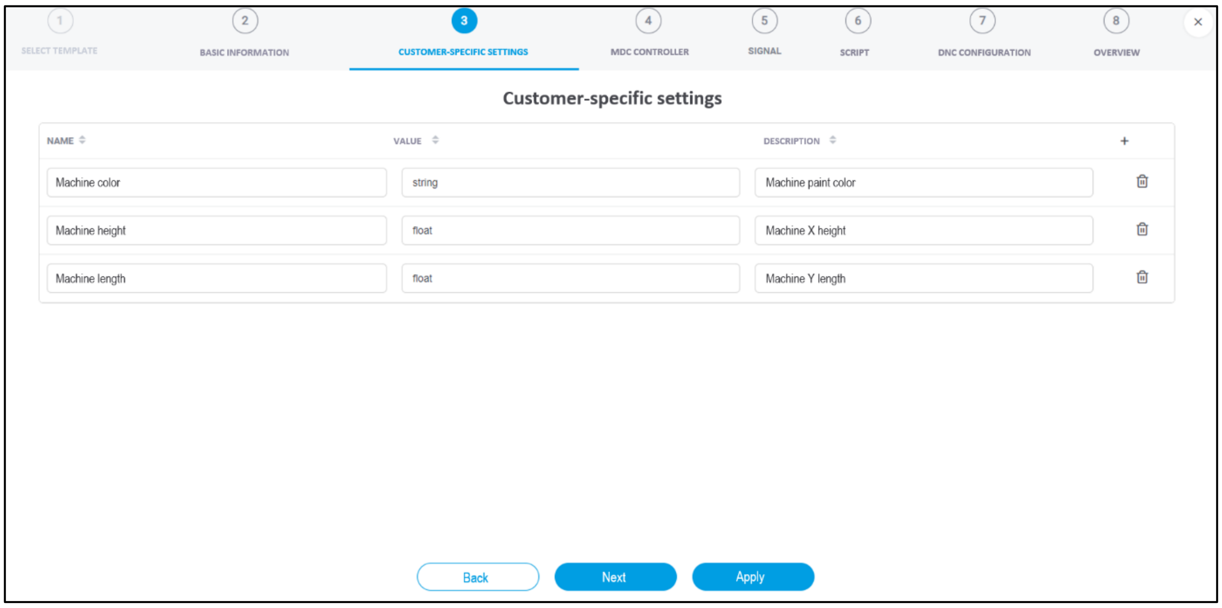
1. Enter **machine name**, **manufacturer**, **model no**. and **serial no**.
2. Optional: Add further information if necessary.
3. Activate **Configure MDC** and/or **Configure DNC**.
4. Click on **Next**.

6.2.3 ③ Customer-specific settings

This enables individual, plant-specific information to be added to a machine to further supplement the machine's data. This data can later be retrieved from the API to provide more information to a third-party system.

Example: Name = location, value = hall 2.
This is where an additional locational aspect is added to the machine data to help accurately locate the machine in the event of a malfunction.

ⓘ This step is optional.



Customer-specific settings

NAME	VALUE	DESCRIPTION
Machine color	string	Machine paint color
Machine height	float	Machine X height
Machine length	float	Machine Y length

Fig. 17: Add machine – Customer-specific settings

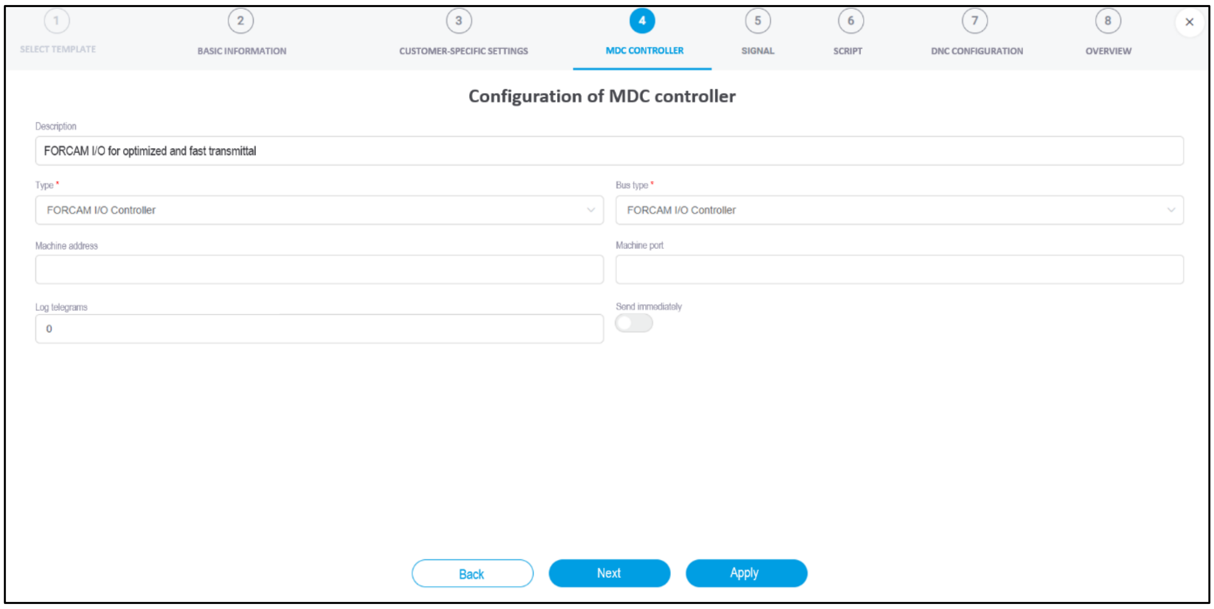
1. Click on the **+** icon.
2. Enter the desired parameters.
3. Click on **Next**.

6.2.4 ④ MDC controller

Option to configure an MDC controller. Specifies the type and method of connection to the machine. FORCAM supports all common controllers on the market and continuously strives to expand their availability. An overview of the current FORCAM plugins is listed in section 8.4.

The bus type is a specific communication protocol of the controller type. Many controllers have only one protocol and therefore only one bus type to choose from (e.g. bus type **FORCAM I/O Controller connection** for controller **FORCAM I/O Controller**). For the Siemens S7 controller, for example, several protocols are possible, which is why there are several bus types available.

ⓘ This step is only available if ② **Configure MDC** was selected in step .



Configuration of MDC controller

Description: FORCAM I/O for optimized and fast transmittal

Type *: FORCAM I/O Controller

Bus type *: FORCAM I/O Controller

Machine address:

Machine port:

Log telegrams: 0

Send immediately

Fig. 18: Add machine - MDC controller

1. Optional: Enter the description of the control.
2. Select the control type.

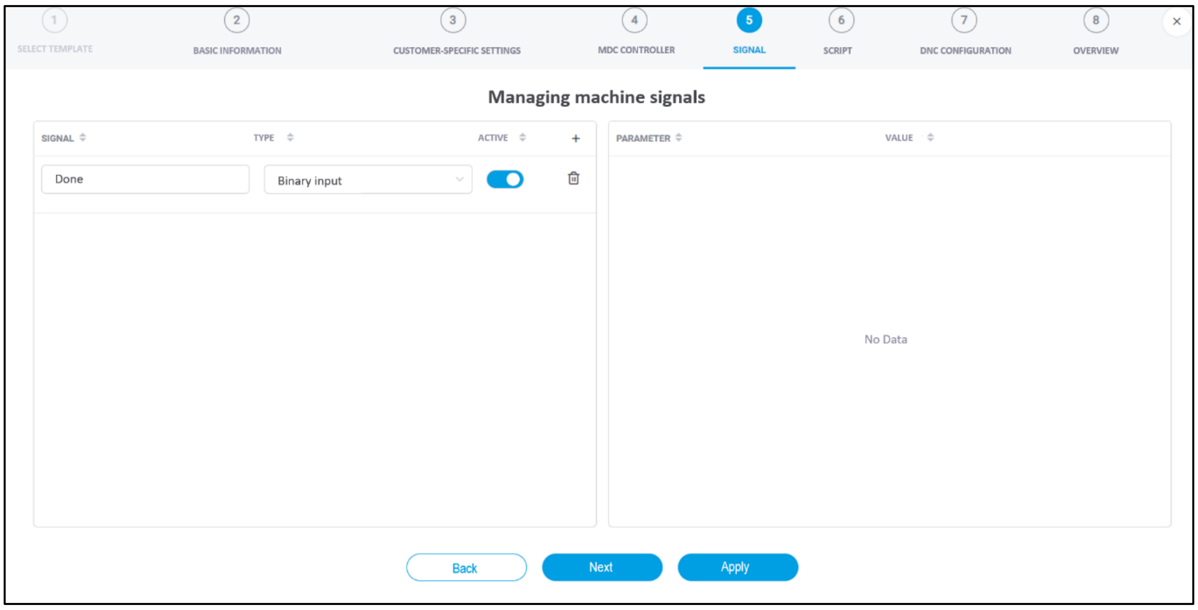
➔ Additional configuration parameters appear depending on the selected type.

3. Select bus type.
   The possible selection depends on what type of control was previously selected.
4. Adjust the configuration according to the controller.
5. Click on **Next**.

ⓘ The **Log telegrams** function is used to log out of the UDP telegrams in the DCU log. The number of characters for logging out of each telegram is specified in the input field.

6.2.5 ⑤ Signal

Defines which signals are read from the controller. For example, a binary signal can be configured to allow the interpretation "Machine running" or "Machine not running". If 1 is reported, the machine is in production, and if it is 0, it is at a stoppage. In this example, the signal is called "DONE". Additionally, different data types can be used to map various signals such as temperature. Signals can also be defined that can be written to the controller, e.g. an order number or default times.



SIGNAL	TYPE	ACTIVE
Done	Binary input	✓

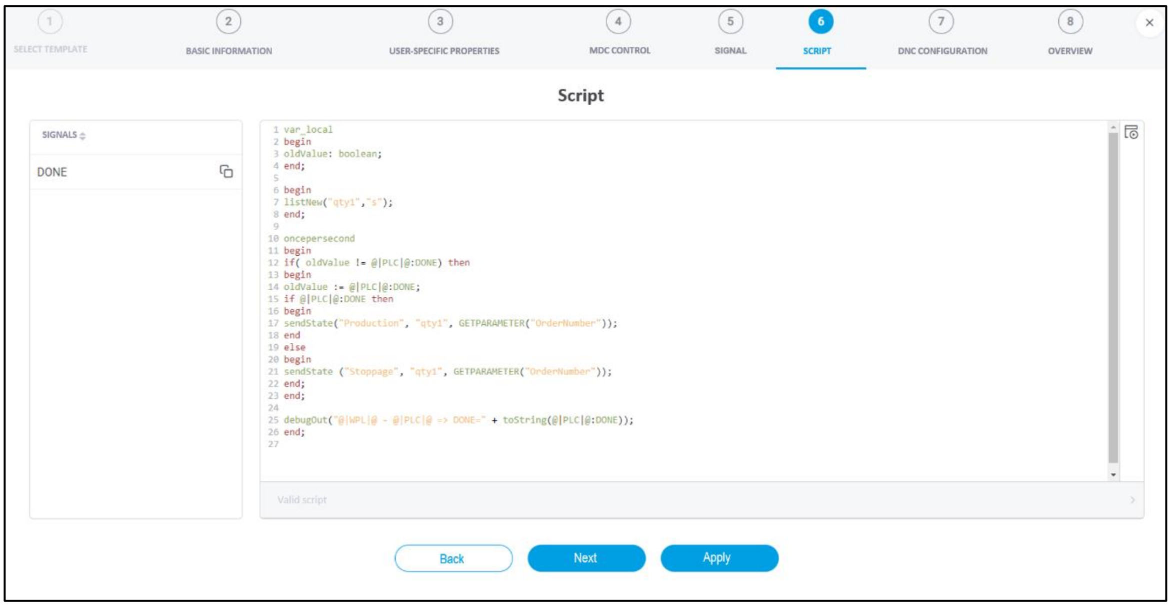
Fig. 19: Add machine - Signal

1. Click on the **+** icon.
2. **Enter Name**, **type** and **description**.
3. Expand the drop-down menu on the left.
   The name is prefilled according to the selected type.
4. Enter the desired value.
5. Click on **Next**.

6.2.6 ⑥ Script

Defines how the machine signals are to be interpreted.
Different signals can be read out via a script depending on the control, e.g. "Production in partial automatic state" or " Stoppage due to mechanical error".
All signals configured in step ⑤ are listed in the left part of the mask.
The area in the center is the input field for a script, where the actual machine logic is defined.

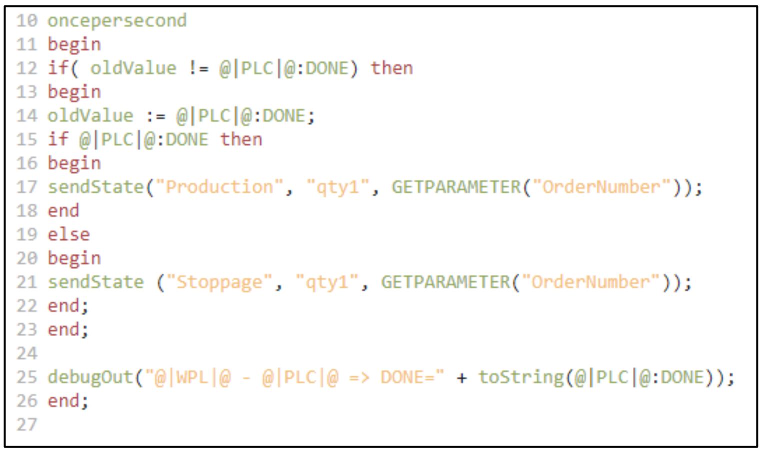
ⓘ See the **Scripting Language** manual for a listing of formula elements and operators in scripts.



```
var_local
begin
oldValue: boolean;
end;

begin
listNew("qty1","s");
end;

oncepersecond
begin
if( oldValue != @|PLC|@:DONE) then
begin
oldValue := @|PLC|@:DONE;
if @|PLC|@:DONE then
begin
sendState("Production", "qty1", GETPARAMETER("OrderNumber"));
end
else
begin
sendState ("Stoppage", "qty1", GETPARAMETER("OrderNumber"));
end;
end;

debugOut("@|WPL|@ - @|PLC|@ => DONE=" + toString(@|PLC|@:DONE));
end;
```

Fig. 20: Add machine - Script

In order to follow the example from the "DONE" signal in step ⑤, the script in Fig. 20 is as follows: If bit 1 for "DONE" comes from the machine, an event **sendStateProduction** is sent, thus eventually corresponding to the status **Production**. If no bit flag is set to 1, **sendStateStoppage** is sent, thus **stoppage**. More script examples are available in section 8.6.



```
oncepersecond
begin
if( oldValue != @|PLC|@:DONE) then
begin
oldValue := @|PLC|@:DONE;
if @|PLC|@:DONE then
begin
sendState("Production", "qty1", GETPARAMETER("OrderNumber"));
end
else
begin
sendState ("Stoppage", "qty1", GETPARAMETER("OrderNumber"));
end;
end;

debugOut("@|WPL|@ - @|PLC|@ => DONE=" + toString(@|PLC|@:DONE));
end;

```

Fig. 21: Excerpt from the script sample for the "DONE" signal

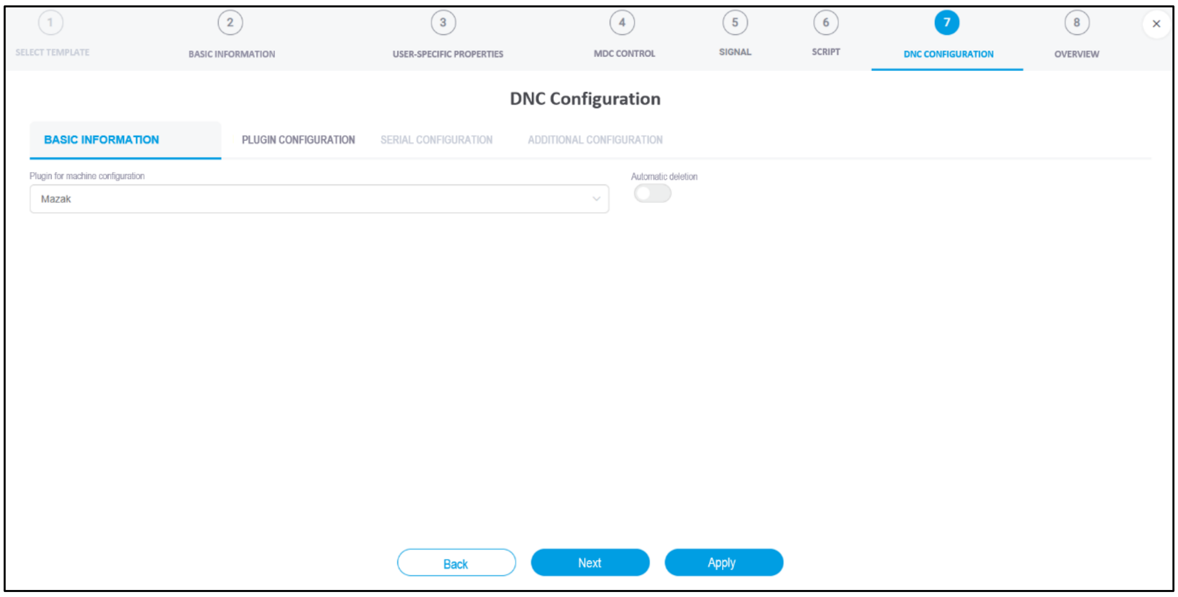
To configure a script:

1. In the left-hand area, select the signal to which you want to add a script.
2. Enter the desired script in the central input field.
3. Optional: Use the play icon in the upper right corner to run the script and check the validity.
4. Click on **Next**.

6.2.7 ⑦ DNC configuration

Option to configure a DNC control. Determines the way in which an NC file is to be transferred to the machine.
FORCAM supports all common controllers on the market and continuously strives to expand their availability. An overview of the current FORCAM plugins is listed in section 8.4.

ⓘ This step is only available if **Configure DNC** was selected in step ②.



Plugin for machine configuration: Mazak
Automatic deletion

Fig. 22: Add machine - DNC configuration

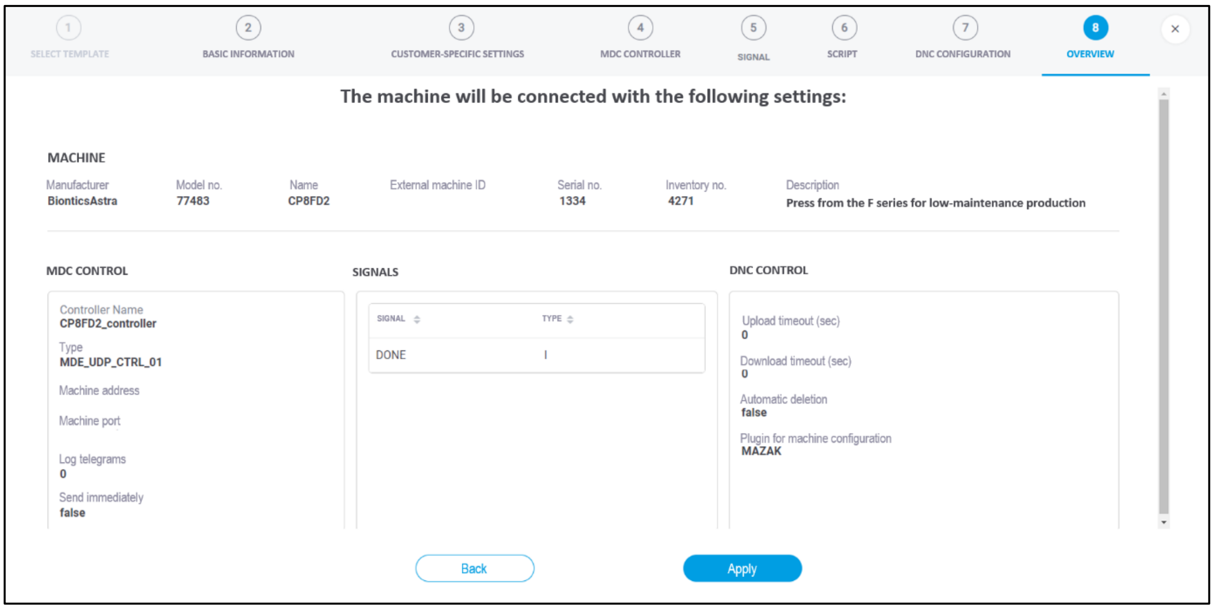
1. Optional: Enter the **Upload** and **Download timeout**.
2. Select **plugin for machine configuration**.

➔ Additional configuration parameters may appear in further tabs depending on the selected plugin.

3. Optional: Set automatic deletion.
   If enabled, the NC file is automatically deleted from the machine after it has been read.
4. Enter additional configuration parameters depending on the selected plugin in the remaining tabs.
5. Click on **Next**.

6.2.8 ⑧ Overview

A summary of the previous configuration from all steps and a list of all defined signals. After confirming, the machine is mapped with the specified configuration and is therefore digitized. The configured machine appears under the specified name in the overview (see Fig. 11).



The machine will be connected with the following settings:

MACHINE

Manufacturer	Model no.	Name	External machine ID	Serial no.	Inventory no.	Description
BionticsAstra	77483	CP8FD2		1334	4271	Press from the F series for low-maintenance production

MDC CONTROL

Controller Name
CP8FD2_controller
Type
MDE_UDP_CTRL_01
Machine address
Machine port
Log telegrams
0
Send immediately
false

SIGNALS

SIGNAL	TYPE
DONE	I

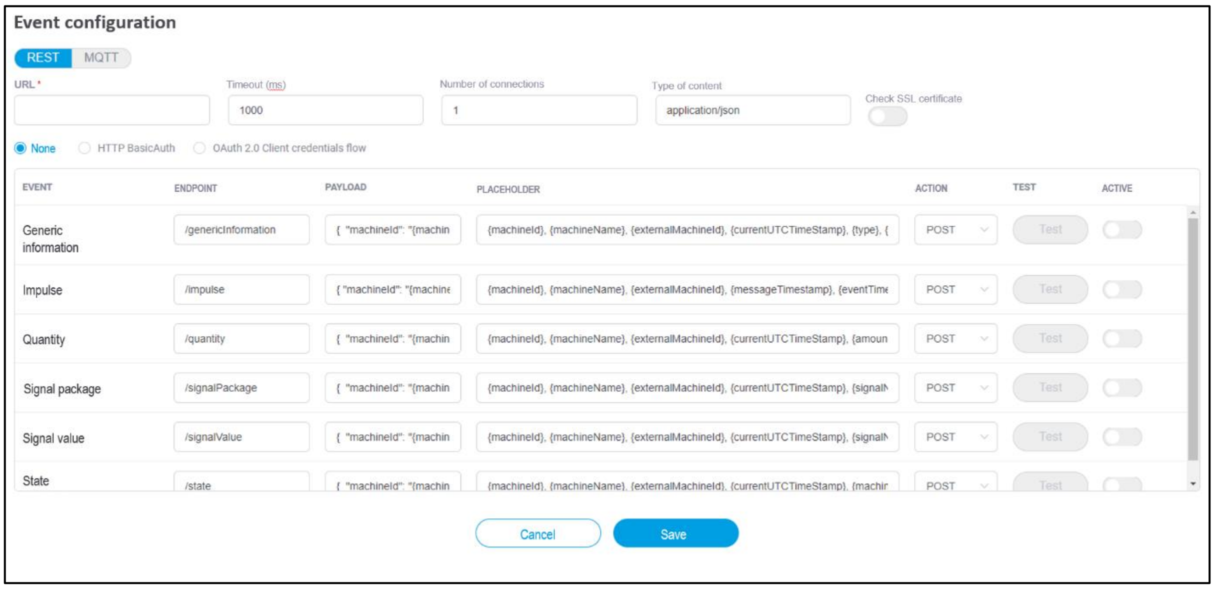
DNC CONTROL

Upload timeout (sec)
0
Download timeout (sec)
0
Automatic deletion
false
Plugin for machine configuration
MAZAK

Fig. 23: Add machine - Overview

6.3 Event configuration

The event configuration specifies how the signals are sent to a superordinate system. Payload and endpoint are predefined by default, but they can be customized.


Event configuration

REST | MQTT

URL * | Timeout (ms): 1000 | Number of connections: 1 | Type of content: application/json | Check SSL certificate

None | HTTP BasicAuth | OAuth 2.0 Client credentials flow

EVENT	ENDPOINT	PAYLOAD	PLACEHOLDER	ACTION	TEST	ACTIVE
Generic information	/genericInformation	{ "machineId": "{machin	{machineId}, {machineName}, {externalMachineId}, {currentUTCTimeStamp}, {type}, {	POST	Test	
Impulse	/impulse	{ "machineId": "{machine	{machineId}, {machineName}, {externalMachineId}, {messageTimestamp}, {eventTime	POST	Test	
Quantity	/quantity	{ "machineId": "{machin	{machineId}, {machineName}, {externalMachineId}, {currentUTCTimeStamp}, {amoun	POST	Test	
Signal package	/signalPackage	{ "machineId": "{machin	{machineId}, {machineName}, {externalMachineId}, {currentUTCTimeStamp}, {signalN	POST	Test	
Signal value	/signalValue	{ "machineId": "{machin	{machineId}, {machineName}, {externalMachineId}, {currentUTCTimeStamp}, {signalN	POST	Test	
State	/state	{ "machineId": "{machin	{machineId}, {machineName}, {externalMachineId}, {currentUTCTimeStamp}, {machir	POST	Test	

Fig. 24: Event configuration in FORCAM FORCE EDGE

Events are used in a script to trigger outgoing events. For this, there are script functions available that generate a corresponding event depending on the type.
In the sample from section 6.2.6, for example, the script function **sendStateProduction** was used, which sends the state **Production** as an event to a third-party system.
For each type of event there is a standardized **Event**. For example, the **Quantity** event type sends the quantity produced by the machine. All available events are listed in section 6.3.

Under **PAYLOAD** the JSON body defines what the message to the superordinate system should look like. Finally, the placeholders (wildcards) are replaced by the corresponding existing signals. Example of an event structure:

```
{
        machineId: $machineId$
        machineName: $machineName$
        externalMachineId: $externalMachineId$
        reference: $reference$
        timeStamp: $currentUTCTimeStamp$
        signalName: $signalName$
        value: $value$
        unit: $unit$
}
```

Script functions allow events to use **placeholders** (wildcards) which can be used to transmit different information. This can be used, for example, to get the machine ID or the timestamp formatted in UTC. Section 8.7 lists and explains all available script functions.

If **ACTIVE** is enabled, the corresponding event will be sent. Events that are not enabled will not be sent.
An enabled event can also be tested by clicking **TEST**. In the subsequent dialog, values such as **machineID** (machine ID) or **Value** can be entered to generate and execute the signal as an example without influencing the actual machine connection. This allows events to be tested in advance without having to execute them in the live environment.

6.3.1 Signals & events from EDGE to superordinate system

There are two technical options for supplying signals and events from an EDGE node to a third-party application.

ⓘ The supply can be configured in the EDGE node itself.

HTTP**/**REST

To supply the external system, any REST endpoint provided there can be used. The HTTP methods POST and PUT are supported.
The following standards are implemented as HTTP authentication methods:

- Basic authentification: Authentication according to RFC 2617 by entering the username and password (see https://datatracker.ietf.org/doc/html/rfc2617).
- Client credential flow: Authentication according to OAuth 2.0 RFC 6749 via client ID and client secret known to the system. (see https://auth0.com/docs/flows/client-credentials-flow). This type of authentication is performed without user intervention, i.e. in the background.

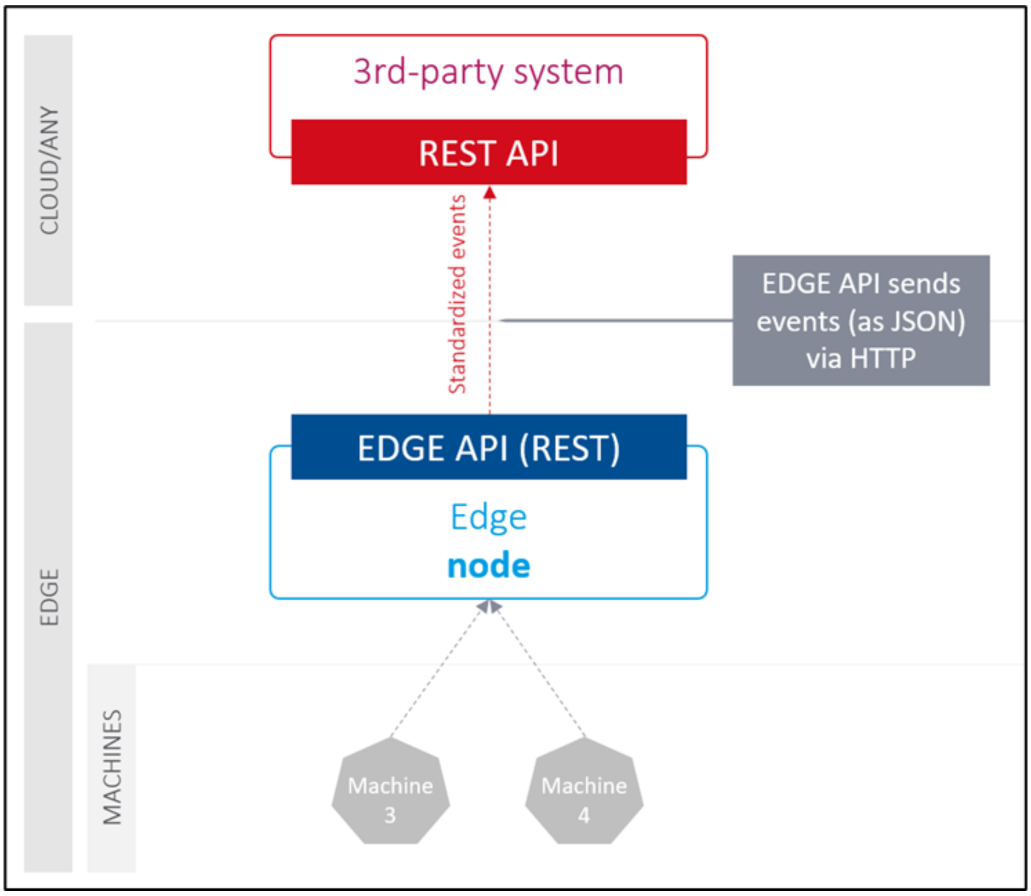


Fig. 25: Communication with superordinate systems via HTTP/REST

MQTT messaging

Any MQTT broker can also be used to supply the data, if provided by customers or partners.

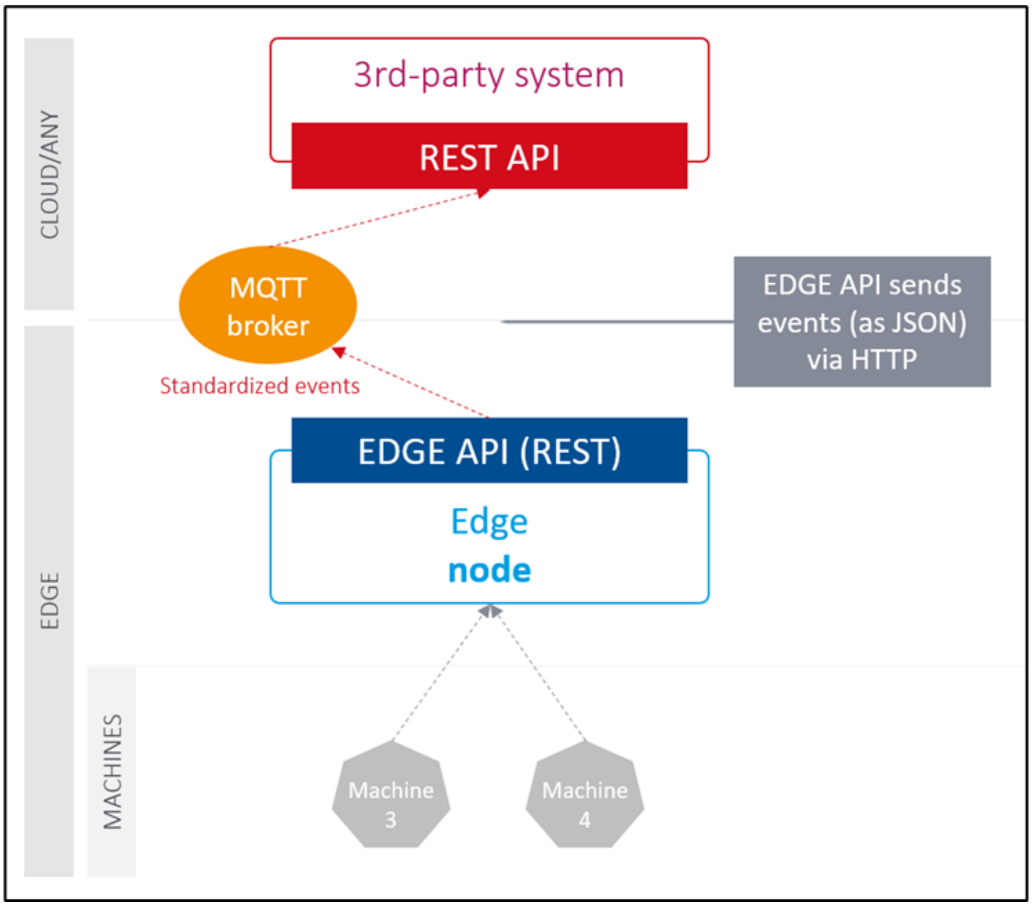


Fig. 26: Communication with superordinate systems via MQTT broker

6.3.2 Data & documents from superordinate system to EDGE

The EDGE API can be used to supply FORCAM FORCE EDGE with data and documents. Technically, this is only possible via **HTTP/REST**.
The following interfaces are provided:

Table 2: Interfaces for transferring data and documents

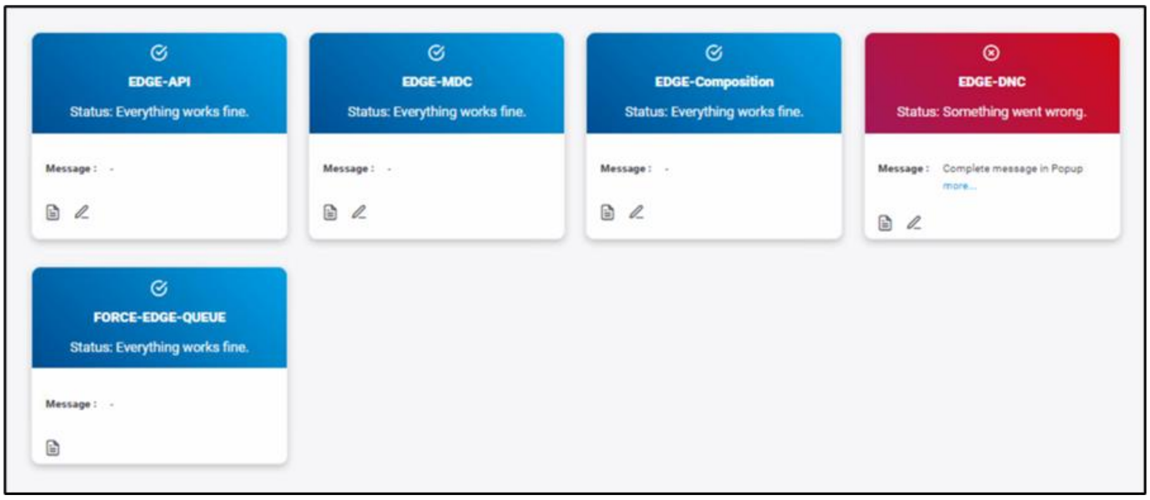
Interface	Description
Transfer of process and reference parameters	These business parameters can be used in the EDGE Composition Layer to supplement standardized events (e.g. order number or cycle time).
Transfer of signal parameters	Parameter values for specific signals can be transferred. These are written directly to the machine control.
Transfer of documents	NC programs can be transferred, which are also transferred to the machine control.

6.3.3 Configure an event

1. Click on the event configuration icon in the upper right-hand part of the detailed view of a configured machine (see Fig. 11).
2. In the subsequent dialog determine if **REST** or **MQTT** should be used.
3. In the upper bar, enter **URL** and optionally other details such as timeout, etc.
   Specifies where the events should be sent.
4. Set SSL certificate validation.
   If the switch **Check SSL certificate** is active, FORCAM FORCE EDGE can also be connected via REST to an application that does not have a valid or unsigned security certificate.
5. Select desired authentication and enter login data.
6. Configure events as desired.
7. Save.

7 Monitoring

FORCAM FORCE EDGE has the option to monitor individual components via the monitoring page. The page indicates if a component is running without errors or if there are any malfunctions.
The monitoring can be called up via the icon in the upper right area in the machine overview (see Fig. 11).


EDGE-API
Status: Everything works fine.
Message : -

EDGE-MDC
Status: Everything works fine.
Message : -

EDGE-Composition
Status: Everything works fine.
Message : -

EDGE-DNC
Status: Something went wrong.
Message : Complete message in Popup
more...

FORCE-EDGE-QUEUE
Status: Everything works fine.
Message : -

Fig. 27: Monitoring in FORCAM FORCE EDGE

Error messages and logs can be retrieved specifically for each component.

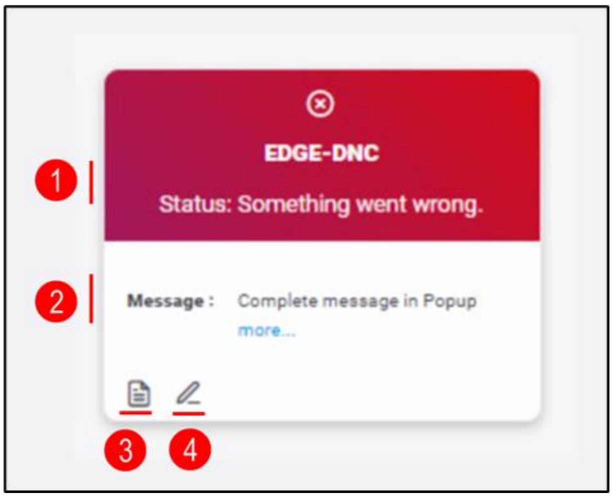

EDGE-DNC
Status: Something went wrong.
Message : Complete message in Popup
more...

Fig. 28: „EDGE-DNC“ component in the Monitoring page

(1) Current status of the component
(2) Message in the event of an error.
Clicking **more...** displays the full error message in a pop-up window.
(3) Displays the last warning and error message of the component for each case
(4) Enables downloading a log file from a specific day

8 Annex

8.1 Table of changes

Table 3: Changes in release version 210809 in document version 2

Date	EDGE version	Doc. version	Change
2021-08-11	210809	2	Replaced landing page (Fig. 3)
2021-08-11	210809	2	Updated list of supported plugins in 8.4: Added for MDC Weihenstephan and WUT; added for DNC Heidenhain, WUT and COM; and removed Soflex and Comportserver.
2021-08-11	210809	2	Added note in 6.2.6 regarding the Script Language manual
2021-08-11	210809	2	Added script 4 in 8.6 as an example for signal packages
2021-08-18	210514	2	Added section 5 Basic settings
2021-08-19	210514	2	Adapted mandatory information at step 1 in 6.2.2
2021-08-19	210514	2	Added note in 6.2 that negative values are not permitted
2021-08-24	210622	2	Added the sentence in 6.2 that configuration pages can be called-up directly during editing
2021-08-25	210717	2	FORCAM FORCE EDGE is now available in French.
2021-08-25	210717	2	Moved step about URL and optional specifications at 6.3.3 to position 3. Added step about SSL certificate verification in position 4. Moved step about authentication to position 5
2021-08-26	210802	2	Added note in 6.2.4 about logging telegrams
2021-08-26	210802	2	Added note in 5.1 about creating a user with the same data
2021-08-26	210802	2	Added note in 6.2.1 about overwriting configuration after a template is selected. Added additional screenshot.
2021-08-27	210809	2	Added Node-RED as a supported plugin (MDC) in 8.4
2021-08-30	210825	2	Added section 4 Deployment
2021-09-01	210825	2	Updated screenshots in section 6
2021-09-01	210825	2	Divided the script examples in 8.6 into individual subchapters
2021-09-14	210825	2	Updated screenshots in section 7

8.2 Document conventions

Table 4: Fonts, formatting and characters used

Conventions	Description
Boldface	Buttons and options names are written in boldface.
Italics	Highlighted words are in italics.
Icons	For a function that is represented by an icon, the icon is referenced as the object.
Action result	Action results are indicated by →.
Prerequisites	Prerequisites are indicated by ✓.
Warnings	Warnings are indicated by ⚠.
Notes	Notes are indicated by ⓘ.
Tips	Tips are indicated by ⓣ.

8.3 Abbreviations and terms

Table 5: Abbreviations and terms used

Abbreviations	Explanation
Brownfield	An existing factory or manufacturing facility that has already been built and in operation for some time. The Brownfield approach in the context of Industry 4.0 means the digital transformation of an existing manufacturing plant.
CP	Communication processor
DNC	Distributed Numerical Control: NC systems that are connected to a computer. The individual systems can be centrally supplied with NC programs and then coordinated.
IT	Information technology
Machine	In FORCAM FORCE EDGE, the machine is a partial plant according to ISA 95. If there are no other partial plants (i.e. additional physical controls), we refer to it as a plant.
MDC	Machine data connection (machine data collection)
MQTT	Message Queuing Telemetry Transport: Open network protocol for machine-to-machine (M2M) communication that enables transmitting telemetry data in the form of messages between devices, despite high delays or network limitations.
MR	Machine Repository
Northbound	A northbound interface communicates with a higher level element in a particular network component.
OT	Operational technology
POST	POST is a method which is supported by HTTP and means that a web server accepts the data contained in the body of the message requested.
PUT	The PUT method is used to update a resource available on the server. Typically, it replaces everything existing at the target URL with something else.
REST	Programming paradigm for distributed systems (collection) of independent computers that present themselves to the user as a single system.)
RESTful API	API for data exchange based on HTTP requests via GET, PUT, POST and DELETE, which is subject to the requirements or restrictions of the REST architecture.
Signal	Values read from the machine control, such as temperature, pressure or certain statuses.
Southbound	An equivalent to the northbound interface, a southbound interface communicates with lower level components.
SPS	Programmable Logical Control
UTC	Coordinated Universal Time
Wildcard	Placeholder for other characters.

8.4 List of supported plugins

MDC Plugins

Table 6: List of all supported machine connection variants

Name	Read	Write	Transmission type Polling/Event based
AUDI SPS	X	X	X/
CSV file exchange	X		X/
Euromap 63	X		X/
Euromap 77 (via OPC UA)	X	X	/X
FANUC	X	X	X/
FORCAM I/O Controller	X	X	/X
FORCAM I/O Controller (hardware)	X		
Heidenhain	X	X	X/
MAKINO Pro 3/Pro 6	X		
Mazak	X		
MCIS RPC (SINUMERIK 810D/840D/840D)	X		X/X
Modbus	X		
MQTT	X	X	/X
MT Connect	X		X/
Node-RED	X	X	/X
OKUMA	X		
Omron	X		
OPC Classic	X	X	X/
OPC UA	X	X	/X
OPC XML	X		X/
Rockwell/Allen Bradley	X	X	X/

Name	Read	Write	Transmission type Polling/Event based
Siemens S5 with CP	X		
Siemens S5 without CP	X		
Siemens S7 with CP	X	X	X/
Siemens S7 without CP	X	X	X/
SQL database exchange	X		X/
Weihenstephan	X		X/
Wiesemann & Theis (WUT)	X		X/

DNC plugins

Table 7: List of all supported NC machine connection variants

Name	Read	Write
COM	X	X
Heidenhain	X	X
Mazak DNC	X	X
RPC Plugin	X	X
FTP Plugin	X	X
FANUC	X	X
File Handler (File Copy)	X	X
File Handler Server	X	X
MOXA-Box	X	X
Wiesemann & Theis (WUT)		

8.5 Standardized events

Table 8: Events and their function in FORCAM FORCE EDGE

Event type	Values	Function
General Information	– Machine ID – Machine name – External machine ID – Reference (any) – Timestamp – Type (any) – Value (any)	Any information
Impulse	– Machine ID – Machine name – External machine ID – Reference (any) – Timestamp – Count	For example: hit, shot etc.
Quantity	– Machine ID – Machine name – External machine ID – Reference (any) – Timestamp – Amount – Unit (optional) – QualityDetail (optional)	Produced quantity
Signal package	– Machine ID – Machine name – External machine ID – Reference (any) – Timestamp – ARRAY [SignalName, Value, TimeStampUTC, Unit (optional)]	Collection of signals (e.g. serial number, pressure and temperature)
Signal value	– Machine ID – Machine name – External machine ID – Reference (any) – Timestamp – SignalName – Value – Unit (optional)	Temperature, pressure, etc.

Event type	Values	Function
State	– Machine ID – Machine name – External machine ID – Reference (any) – Timestamp – State (production or downtime) – Statuscodes (optional list of statuses)	Machine state (production, or stoppage)

8.6 Script examples

8.6.1 Machine status and temperature

The following script sends the status of the machine (production or stoppage). In addition, the temperature is also indicated. As soon as the temperature changes, the updated temperature is sent.

```
var_local

begin
        oldState: boolean;
        oldTemperature: string;
end;

oncepersecond
begin

        if( oldState!= @|PLC|@:DONE) then
        begin
                oldState := @|PLC|@:DONE;
                if @|PLC|@:DONE then
                begin
                        sendStateProduction()
                end
                else
                begin
                        sendStateStoppage();
                end;
        end;

        if( oldTemperature != toString(@|PLC|@:TEMP)) then
        begin
                oldTemperature := toString(@|PLC|@:TEMP);
                sendSignalValue("TEMPERATURE", toString(@|PLC|@:TEMP), "Degrees");
        end;

end;
```

8.6.2 Temperature and humidity

The following script sends the current temperature and humidity. This occurs in intervals of 30 seconds, and as soon as a change of these values takes place.

```
var_local
begin
        oldTemperature : string;
        oldHumidity : string;
        seconds: number;
end;

oncepersecond
begin

        seconds := seconds + 1;

        if (seconds > 30) then
        begin
                seconds := 0;
                oldTemperature := "";
                oldHumidity := "";
        end;

        if (oldTemperature != @|PLC|@:TEMP ) then
        begin
                oldTemperature := @|PLC|@:TEMP;
                sendSignalValue("TEMP", toString(@|PLC|@:TEMP), "Degree");
        end;

        if (oldHumidity != @|PLC|@:HUMIDITY ) then
        begin
                oldHumidity := @|PLC|@:HUMIDITY;
                sendSignalValue("HUMIDITY", toString(@|PLC|@:HUMIDITY), "Degree");
        end;
end;
```

8.6.3 Crane control

This script collects a crane control with the black, green and red buttons.

- The black button turns the machine on and off
- The red button triggers an emergency
- The green button sends a pulse for piece counts then counts this number up

```
var_local
begin
        // GENERAL LOGIC VARIABLES
        seconds:  number;
        // MACHINE STATE
        state: number;
        stateOld: number;
        // MACHINE STATUS REASON
        status_reason: string;
        status_reasonOld: string;
        // PIECE COUNT VARIABLES
        counter: number;
        counterOld: number;
        counterSend: number;
end;

begin
        //DEFINE LISTS START
        ListNew("STATUSCODES", "S");
        //DEFINE LISTS END
end;

begin
        // INITIALIZE SCRIPT VARIABLES START
        if not initialized and not offline(@|PLC|@) then
        begin
                status_reason := " ";
                status_reasonOld := " ";
                counter := @|PLC|@:Good_count;
                counterOld:= counter;
                ListClear("STATUSCODES");
                // Set initialized to perform initializing once
                initialized := true;
        end
        else if initialized then
        begin
                counter := @|PLC|@:Good_count;
                ListClear("STATUSCODES");
        end;
        // INITIALIZE SCRIPT VARIABLES END

        // ACTIONS ONCE PER SECOND START
        oncePerSecond
        begin
                seconds := seconds + 1;
        end;
        // ACTIONS ONCE PER SECOND END

        // DEFINITION STATE / STATUS_REASON START
        if offline(@|PLC|@) then
        begin
                state := "1";
                status_reason :='NOT_CONNECTED';
                seconds := 0;
        end
        else if not @|PLC|@:Emergency_ON then
        begin
                state := "1";
                status_reason :='EMERGENCY_ON';
                seconds := 0;
        end
        else if not @|PLC|@:Machine_ON then
```

```
begin
        state := "2";
        status_reason := 'PRODUCTION'
        seconds := 0;
end
else
begin
        if seconds > waiting period then
        begin
                state := "1";
                status_reason := 'UNDEFINED_STOPPAGE'
                seconds := 0;
        end
end;
// DEFINITION state END

// DEFINITION COUNTER START
if counter >= counterOld then                   // Part counter on PLC is incremented
begin
        counterSend := counter - counterOld;
        counterOld := counter;
end
else if counter < counterOld then               // Part counter on PLC changes to negative
begin
        counterSend := 32768 - counterOld;
        counterOld := counter;
end;
// DEFINITION COUNTER END

// SEND state status_reason START
if state <> stateOld or status_reason <> status_reasonOld then
begin
        if state == 2 then
        begin
                ListAdd("STATUSCODES", status_reason);
                sendStateProduction("STATUSCODES");
        end
        else
        begin
                if == 1 then
                begin
                        ListAdd("STATUSCODES", status_reason);
                        sendStateStoppage("STATUSCODES");
                end
        end;
        debugOut("@|PLC|@" + "Send state: " + state);
        stateOld := state;
        status_reasonOld := status_reason;
end;
// SEND state status_reason END


// SEND STROKES / QUANTITY START
if counterSend > 0 and packetNo <> packetNoOld then
begin
        debugOut("@|PLC|@" + "Send quantity: " + toString(counterSend));
        SendQuantity(counterSend);
        counterSend := 0;
end;
// SEND STROKES / QUANTITY END

// LOGGING SIGNALS WHEN CHANGED START
logstring := "@|PLC|@ Signals: " + " offline: "            + toString(offline(@|PLC|@))
                                 + " State: "              + toString(state)
                                 + " status Reason: "      + toString(status_reason)
                                 + " Machine_ON: "         + tostring(@|PLC|@:Machine_ON)
                                 + " Emergency_ON: "       + tostring(@|PLC|@:Emergency_ON )
                                 + " COUNTER: "            + tostring(@|PLC|@:Good_count)
                                 + " seconds: "            + toString(seconds);

if logString <> logstringOld then
begin
        debugOut(logString);
```

```
                logstringOld := logString;
        end;
        // LOGGING SIGNALS WHEN CHANGED END
end;
```

8.6.4 Signal package

The following script is an example of signal packages:

```
//
// Task: Send machine state / status_reason / quantities to runtime
// Created: 2021-05-12
// Version: 1.0
// Author:  FORCAM MDC
//
// --------------------------------------------------------------
//
// Incoming signals
// Reg1 = holding register  1
//
//
// Outgoing information
// //state         = machine state
// //STATUSCODES   = Status reason
// Reg1SEND   = just display holding register
////--------------------------------------------------------------

// VARIABLES
var_local
begin
// GENERAL  VARIABLES
    seconds: number;
    logstring: string;
    logstringOld: string;
//  SIGNAL VARIABLES
    H1Old: number;
    H2Old: number;
    H3Old: number;
    H1Old: number;
    H5Old: number;
    H1Old: number;
    H1Old: number;
    H8Old: number;
    H9Old: number;
    H10Old: number;
 // SCRIPT INIZIALIZING VARIABLES
    initialized: boolean;
end;

begin
     if not initialized and not offline(@|PLC|@) then
    begin
    //DEFINE LISTS START (S=strin B=boolean N=number)
        ListNew("Signals", "S");
        ListNew("Values", "S");
//        ListNew("Timestamps", "S");
    //DEFINE LISTS END
    end;

// INITIALIZE SCRIPT & VARIABLES START
    if not initialized and not offline(@|PLC|@) then
    begin
          H1Old := 0;
          H2Old := 0;
          H3Old := 0;
          H4Old := 0;
          H5Old := 0;
          H6Old := 0;
          H7Old := 0;
          H8Old := 0;
          H9Old := 0;
          H10Old := 0;
          ListClear("Signals");
          ListClear("Values");
 //         ListClear("Timestamps");
//    set initialized to perform initializing once
      initialized := true;
    end
    else if initialized then


// ACTIONS ONCE PER SECOND START
   oncePerSecond
   begin
     seconds:= seconds + 1;

// ACTIONS ONCE PER SECOND END
// send one package for all 10 holding registers  for now always
// Reg1Content Start
    if( H1Old <> @|PLC|@:H1 ) then
   begin

 //fill lists
      H1Old := @|PLC|@:H1;
       ListAdd("Signals", "H1");
      ListAdd("Values", toString(@|PLC|@:H1));
      H2Old := @|PLC|@:H2;
      ListAdd("Signals", "H2");
      ListAdd("Values", toString(@|PLC|@:H2));
      H3Old := @|PLC|@:H3;
      ListAdd("Signals", "H3");
      ListAdd("Values", toString(@|PLC|@:H3));
      H4Old := @|PLC|@:H4;
      ListAdd("Signals", "H4");
      ListAdd("Values", toString(@|PLC|@:H4));
 //     H5Old := @|PLC|@:H5;
 //     ListAdd("Signals", "H5");
 //     ListAdd("Values", toString(@|PLC|@:H5));
 //     H6Old := @|PLC|@:Reg6;
 //     ListAdd("Signals", "H6");
 //     ListAdd("Values", toString(@|PLC|@:H6));
 //     H7Old := @|PLC|@:H7;
 //     ListAdd("Signals", "H7");
 //     ListAdd("Values", toString(@|PLC|@:H7));
 //     H8Old := @|PLC|@:H8;
 //     ListAdd("Signals", "H8");
 //     ListAdd("Values", toString(@|PLC|@:Reg8));
 //     H9Old := @|PLC|@:H9;
 //     ListAdd("Signals", "H9");
 //     ListAdd("Values", toString(@|PLC|@:Reg9));
 //   H10Old := @|PLC|@:H10;
 //    ListAdd("Signals", "H10");
 //     ListAdd("Values", toString(@|PLC|@:H10));
//      sendSignalValue("HoldingReg1", toString(@|PLC|@:H1));
 //     sendSignalValue("HoldingReg2", toString(@|PLC|@:H2));
 //     sendSignalValue("HoldingReg3", toString(@|PLC|@:H3));
//      sendSignalValue("HoldingReg4", toString(@|PLC|@:H4));
 //     sendSignalValue("HoldingReg10", toString(@|PLC|@:H10));
//  send Signal Package with lists
                                               SendSignalPackage("Signals", "Values")
       //initialize list
      begin
         ListClear("Signals");
         ListClear("Values");
      end;

// SENDING Holding Register  END

// LOGGING SIGNALS WHEN CHANGED START
    logstring := "@|PLC|@ Signals: "
                                    + " Reg 1: "            + tostring(@|PLC|@:H1)
                                    + " Reg 2: "            + tostring(@|PLC|@:H2)
```

```
                                       + " Reg 3: "                + tostring(@|PLC|@:H3)
                                       + " Reg 4: "                + tostring(@|PLC|@:H4)
   //                                    + " Reg 5: "                + tostring(@|PLC|@:H5)
   //                                    + " Reg 6: "                + tostring(@|PLC|@:H6)
   //                                    + " Reg 7: "                + tostring(@|PLC|@:H7)
   //                                    + " Reg 8: "                + tostring(@|PLC|@:H8)
   //                                    + " Reg 9: "                + tostring(@|PLC|@:H9)
   //                                    + " Reg 10: "               + tostring(@|PLC|@:H10)
                                                                         ;

    if logString <> logstringOld then
    begin
      debugOut(logString);
      logstringOld := logString;
    end;
// LOGGING SIGNALS WHEN CHANGED END
end;
end;
end;
```

8.7 Script functions

Application	Script function Parameters in [..] are optional	Description	Output event
Default	SendImpulse(ImpulseCount, [Reference])	Sends impulses.	Impulses
Default	SendQuantity(Quantity, [Unit], [QualityDetail], [Reference])	Sendet quantity.	Quantity
Custom	SendState(State, [StatusCodesListName], [Reference])	Sends status.	State
Default	SendStateProduction([StatusCodesListName], [Reference])	Sends production status.	State
Default	SendStateStoppage([StatusCodesListName], [Reference])	Sends the stop state.	State
Default	SendSignalValue(SignalName, Value, [Unit], [Reference], [CustomerSpecificSetting], [Timestamp])	Sends the value of a signal. Data type "Long" (L) must be used for the timestamp list.	SignalValue
Default	SendSignalPackage(SignalNamesListName, ValuesListName, [UnitsListName], [Reference], [CustomerSpecificSetting], [TimestampsListName])	Sends signal values as a package. Data type "Long" (L) must be used for the timestamp list.	SignalPackage
Custom	SendGenericInformation(ParamName, ParamValue, [Reference])	Sends generic information.	GenericInformation
Helfer	ListNew(ListName, DataType)	Creates a new list with the name ListName and list elements of the data type DataType (S for string, B for boolean, N for number).	-
Helper	ListAdd(ListName, Value)	Adds an element to the list.	-
Helper	ListClear(ListName)	Empties the list.	-
Helper	ListDelete(ListName)	Deletes the list.	-

Application	Script function Parameters in [..] are optional	Description	Output event
Helper	GetMachineStatus()	Indicates the machine status.	-
Helper	GetMachineData(ParameterName)	Indicates machine data for the specified parameter.	-
Helper	SetParameter(ParameterName, ParameterValue)	Sets a new value for the specified parameter.	-
Helper	GetParameter(ParameterName)	Fetches the value for the specified parameter.	-
Helper	DeleteParameter(ParameterName)	Deletes the parameter.	-
Helper	DeleteAllParameters()	Deletes all parameters.	-
Helper	OFFLINE	Indicator whether the controller is offline or not.	-
Helper	CONTROLLERERROR1 / 2	Error number 1 and 2 for the controller.	-
Helper	IPADDRESS	The IP address of the DACQ.	-
Helper	HOSTNAME	Hostname of the DACQ.	-
Helper	OFFLINESTRING	Controller offline error string.	-
Helper	SQRT(args)	Root function MATH.	-
Helper	SIN(args)	Sine function MATH.	-
Helper	COS(args)	Cosine function MATH.	-
Helper	TAN(args)	Tangent function MATH	-

Application	Script function Parameters in [..] are optional	Description	Output event
Helper	S5TIMETONUMBER(args)	Siemens S5 time (integer) to number (integer).	-
Helper	NUMBERTOS5TIME(args)	Number (integer) to Siemens S5 time (integer).	-
Helper	RISINGEDGE(args)	Checks if the last inspected value was false and if it is true.	-
Helper	FALLINGEDGE(args)	Checks if the last inspected value was true and if it is false.	-
Helper	SUBSTRING(str, startIndex[, endIndex])	Substring of the specified string.	-
Helper	TONUMBER(str)	String to number (double), replaces comma to period in string.	-
Helper	TOSTRING(str or number[, formatSpecifier])	Specifies the format of the form width. The default formatting is used for empty strings. Width is the minimum length of the result string. Precision is the number of decimal places. If not specified, 0 is used. If the format specification starts with 0, the result string is prefixed with filled zeros. If the format specification ends with X, the number is converted to hexadecimal, using upper or lower case letters with upper or lower case x. In this case, the decimal places are always cut off.	-
Helper	LENGTH(obj)	The length of an object as a string value.	-

Application	Script function Parameters in [..] are optional	Description	Output event
Helper	FORMATTIME(timeformatStr, timeOffset, [, timeunit])	Formats the current time with the time unit as one of the following: MILLISECOND SECOND MINUTE HOUR TAG MONTH YEAR MSABSOLUTE (current time) "R" at Format is specified as a number in milliseconds, otherwise the format is used and the offset and time unit are used to calculate the time.	-
Helper	STDLOG(ignored, logLevel, suffixNumber, logText)	The first parameter is ignored. The log level should be W = warning, C or F = error and everything else for the debug level. The suffix number, if not 0, is added to the end of the log text as "(<SuffixNumber>)" with script loggers.	-
Helper	FILELOG(filepath, textToAppend)	Creates or appends a file with a file path ('/' used as separator) with a specified text to be appended.	-
Helper	SENDTOFORCAM(str1[, str2]) / SENDTOCLIENT (str1[, str2])	If a string already exists, "clientsend" or "forcamsend" is used as the first argument, otherwise the first string is used as the first argument to send the data of the second string to the TCP connections configured in Javis.INI..	-
Helper	DEBUGOUT(text)	Logs the text at debug log level with parser logger.	-
Helper	COPYFILE(inFile, outFile)	Copies data from in-file to out-file. Arguments can be file paths. If successful, the last modified out-file is also updated as in-file.	-

Application	Script function Parameters in [..] are optional	Description	Output event
Helper	COPYREPLACE(inFile, outFile, searchStr, replaceStr)	Copies from in-file to out-file as with function COPYFILE, replacing all incidences of search-string with replace-string.	-
Helper	ATTIME(seconds, obj)	Calculates the object every day at specified times in seconds (seconds represents time fraction of the current day in seconds).	-
Helper	FROMASCII(num)	Returns a string that has the numeric value specified as num.	-
Helper	SLEEP(ms)	Pauses the current thread for a specified time in milliseconds (ms).	-