



# Content

Version 230127

## *Manual*



Document: Manual - FORCAM FORCE  
EDGE 230127.docx



Release date: 2023-02-027



Document version: 1



Author: AEgilmez, STernes

## Content

<b>1</b>	<b>Concept .....</b>	<b>4</b>
<b>2</b>	<b>System Components .....</b>	<b>6</b>
2.1	Asset Connectivity & Model .....	6
2.2	Plugins .....	7
2.3	EDGE API .....	8
2.4	EDGE components .....	8
2.4.1	EDGE Configuration .....	8
2.4.2	Asset Repository .....	8
<b>3</b>	<b>System Architecture .....</b>	<b>9</b>
<b>4</b>	<b>Deployment .....</b>	<b>11</b>
<b>5</b>	<b>Basic settings .....</b>	<b>13</b>
5.1	User administration .....	14
5.2	Supplied Master Data .....	16
5.3	Licensing .....	19
5.4	Download area .....	20
5.5	Monitoring .....	20
5.6	Table Sorting .....	20
<b>6</b>	<b>EDGE Configurations .....</b>	<b>22</b>
6.1	Add EDGE node .....	24
6.2	Add asset .....	25
6.2.1	① Select template .....	26
6.2.2	② Basic information .....	27
6.2.3	③ Customer-specific settings .....	28
6.2.4	④ MDC controller .....	29
6.2.5	⑤ Signal .....	30
6.2.6	⑥ Script .....	31
6.2.7	⑦ DNC configuration .....	32
6.2.8	⑧ Overview .....	33
6.3	Northbound configuration .....	34
6.3.1	Signals & events from EDGE to superordinate system .....	35

## Content

---

6.3.2	Data & documents from superordinate system to EDGE .....	38
6.3.3	Configure an event .....	38
<b>7</b>	<b>Monitoring .....</b>	<b>39</b>
<b>8</b>	<b>Annex .....</b>	<b>41</b>
8.1	Table of changes .....	41
8.2	Document conventions .....	42
8.3	Abbreviations and terms .....	43
8.4	List of supported plugins .....	44
8.5	Standardized events .....	46
8.6	Script examples .....	48
8.6.1	Asset status and temperature .....	48
8.6.2	Temperature and humidity.....	49
8.6.3	Crane control .....	50
8.6.4	Signal package .....	52
8.7	Script functions .....	55

## 1 Concept

FORCAM FORCE EDGE offers manufacturing companies a solution for digitally connecting their heterogeneous assetry. Almost all assets can be digitized with FORCAM FORCE EDGE, regardless of age or technical status. An asset is a generic term for all objects that FORCAM FORCE EDGE can connect, such as machines, sensors, data beacons and IT systems. Thus, FORCAM supports the digital transformation of a manufacturing plant in the Brownfield environment.

FORCAM therefore delivers a product that addresses the main requirement of Industry 4.0 by extracting digital information from the production assetry. 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 asset connections and signals and delivers them as standardized events to superordinate systems. These can be ME (Manufacturing Execution) or MOM (Manufacturing Operation Management) 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 asset park. The assets are connected via a plugin concept for easier future expansion. Many common asset manufacturer-specific (proprietary) protocols are presently supported (such as HEIDENHAIN, Siemens S7 or FANUC & Co.) as well as many common communication standards (such as MTConnect, OPC UA or MQTT). The FORCAM I/O Controller is available as separate hardware for digitizing the asset if the asset is not network-capable. FORCAM FORCE EDGE is continually expanded with plugins in order to meet the challenge of digitally mapping every type of asset via the FORCAM FORCE EDGE.

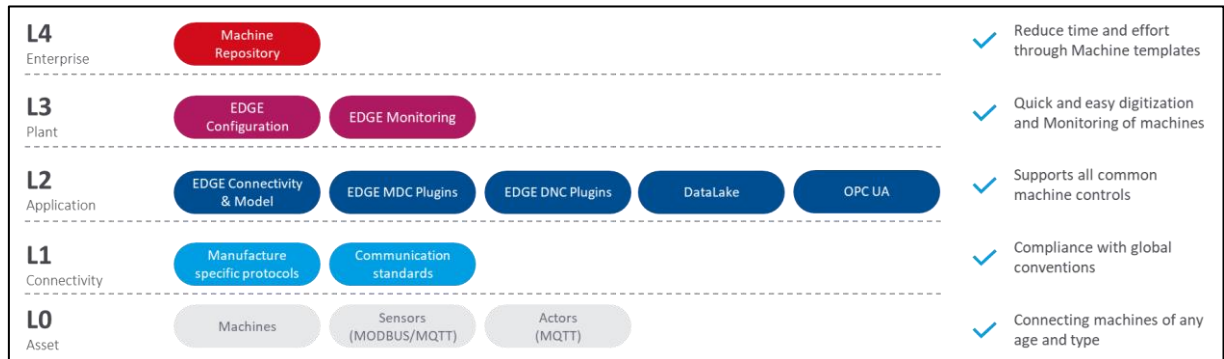
The asset connections are used to obtain a wide variety of information. This includes information about the current status of the connected assets or their sensor readings, such as temperatures, pressures or energy consumption. In the Brownfield environment, it is important not only to pick up the signals and transmit them, but also to interpret them for use. This task is performed by the EDGE Composition Layer. This makes it possible, for example, to interpret when an asset is actually in production or at a standstill. Another essential part of the solution is the handling of NC programs and the ability to transfer them to and from the asset.

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

The Asset Repository component makes it easy to create and use asset templates. This means that templates can be defined for asset connections or derived from existing connections and used for the connection of the same asset types. This further reduces the individual effort required to connect an asset, enabling the time- and resource-efficient implementation of digitization projects. The template structure ensures a standardized connection of identical assets, thus enabling the comparison of assets of the same type.

## Concept

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.



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

Figure 1 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.

## 2 System Components

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



**Fig. 2: Schematic structure of FORCAM FORCE EDGE**

### 2.1 Asset Connectivity & Model

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

#### EDGE MDC Layer

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

#### EDGE DNC Layer

The EDGE DNC Layer manages the actual connection of assets with an NC supply. The elementary components here are the selection of the suitable plugin for communication with the asset control, the configuration of the asset master data, the setting of the network connection and the configuration of the DNC transmission.

**EDGE Composition Layer**

The EDGE Composition Layer enables deriving logical asset 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 assets 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. In addition, options for individual events are also provided. The scripts also make it possible to react to events and write values to the asset's control unit if supported by the controller.

## 2.2 Plugins


Plugins in the FORCAM FORCE EDGE environment are used to establish communication links with specific asset controllers. They allow direct communication with various asset controllers, but also cover modern communication protocols such as MQTT, OPC 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 Asset Data Collection (MDC) and for Distributed Numerical Control (DNC).

MDC plugins include those designed for unidirectional readout of asset signals as well as for bidirectional signal transmission, i.e. for reading out and writing back signals.

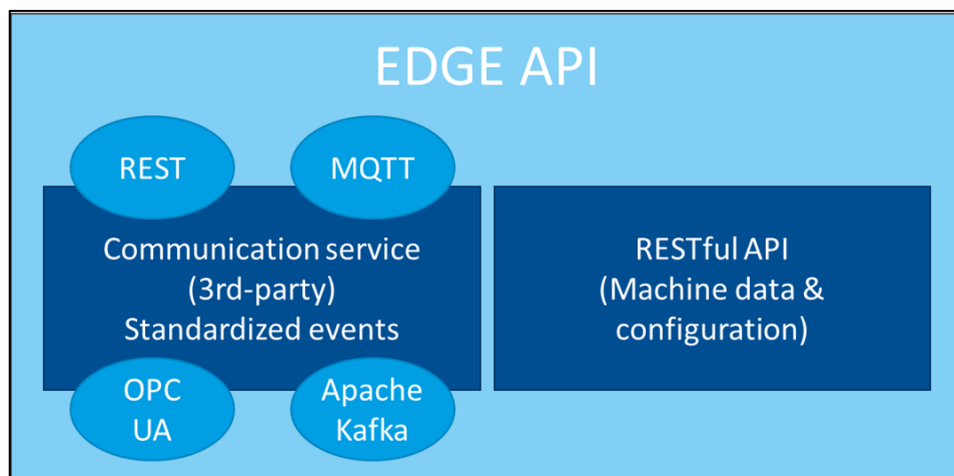
DNC plugins are used for transferring and reading NC files. They are used for transferring NC programs to the asset's file system or to query the program active on the asset.

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 asset master data and for configuring the asset connections. The EDGE API is also used to pass on asset data in the form of standardized events to higher-level systems (3rd party systems). Superordinate systems can be connected either via HTTP/REST, MQTT, Apache Kafka or OPC UA. The message content can be freely configured per connection and per event. When using MQTT or Apache Kafka, a broker is necessary 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.



**Fig. 3: 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 assets. 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 Asset Repository

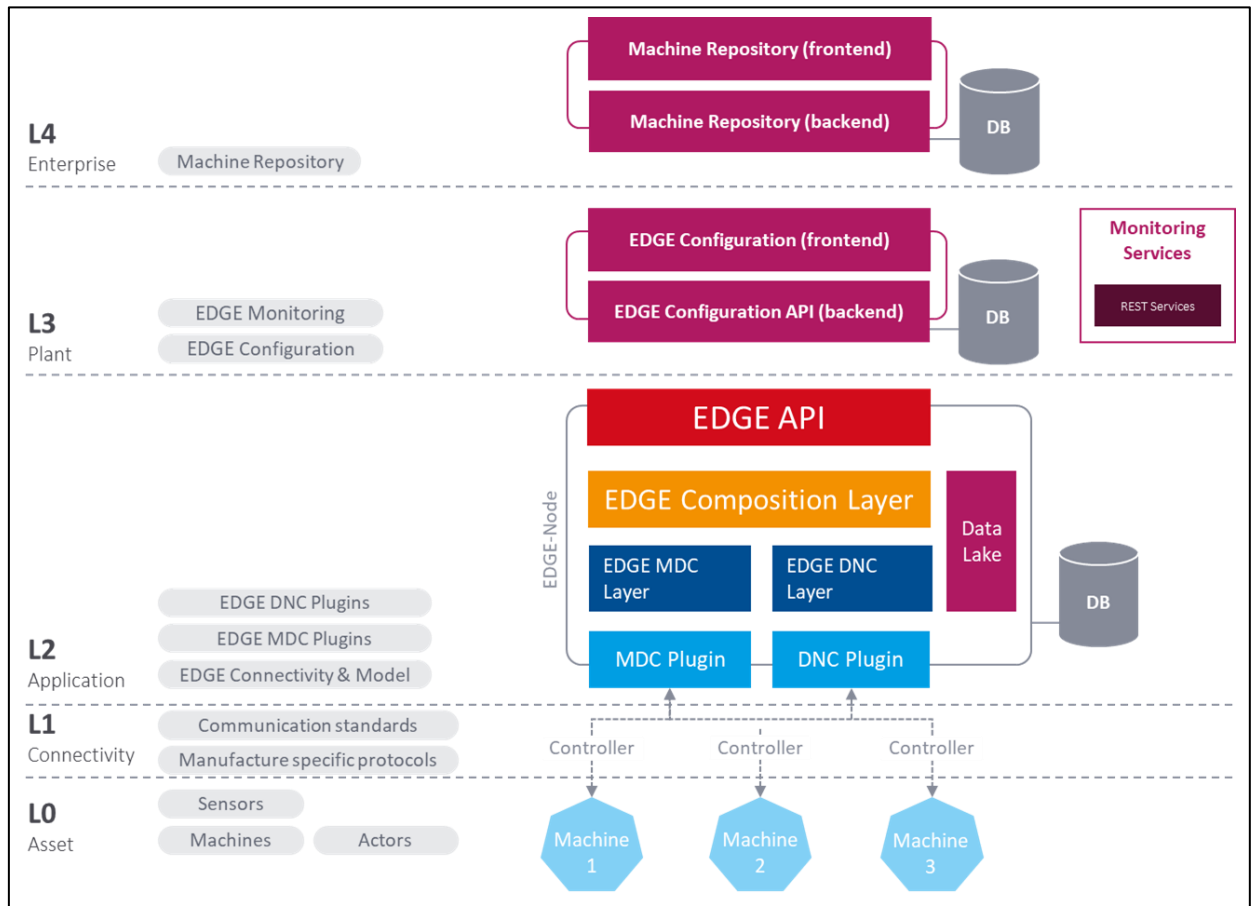
The Asset Repository lets you generate templates from existing asset connections or for new ones. These templates can be used to connect assets of the same type and the same usage type in a standardized manner. The template contains all configuration elements that are not asset-specific. Asset and asset connection-specific configuration elements are, for example, IP address, asset serial number, equipment number, etc.

By using an existing template, the time required to connect an asset 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 assets logically, but also based on performance.



**Fig. 4: FORCAM FORCE EDGE in the System architecture**

#### Level 0 - Asset

The lowest layer is where the assets, sensors and actuators are connected.

#### 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.

## System Architecture

---

### 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 asset via plug-ins and forwards the signals to **EDGE Composition Layer**. Similarly, DNC-capable assets are connected by **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 assets, event management and notification of third party systems can be configured "northbound".
- **Data Lake** records all data, configuration changes as well as write operations and transferred NC data. The data can be accessed via the Data Lake API.
- **DB** contains all configurations related to the asset connections.

### 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 can run independently and without an active connection to other components of the FORCAM FORCE EDGE (e.g. due to temporary loss). 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 Asset Repository is a FORCAM FORCE EDGE extension that lets you create and manage asset connection templates.

## 4 Deployment

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




**Fig. 2: Options for installing FORCAM FORCE EDGE**

### On Premise - model

For an On Premise installation, an installer is provided containing the **EDGE Configuration**, **EdgeNode** and **Asset Repository**. These are installed by the customer themselves, or by a FORCAM service provider.

EDGE Configuration contains the entire user interface including all functions. EDGE Node 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. The asset repository contains the user interface along with its functions. It can support a large number of EDGE instances. An EDGE instance is composed of an EDGE Configuration with its associated EDGE Nodes.

 The Asset Repository must be purchased in addition to the FORCAM FORCE EDGE.

### SAP BTP

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, 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

## Deployment

---

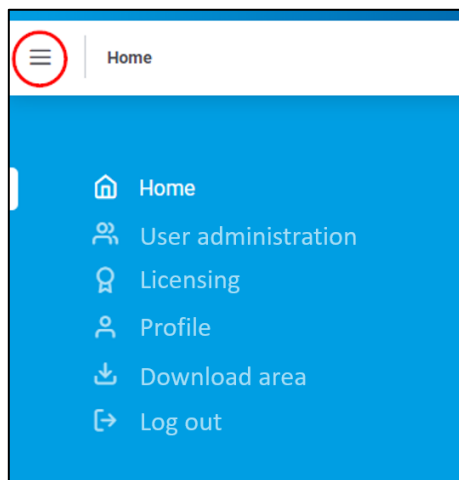
into the existing hardware environment. FORCAM guarantees the function and availability of the ASE and the software components.

## 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.

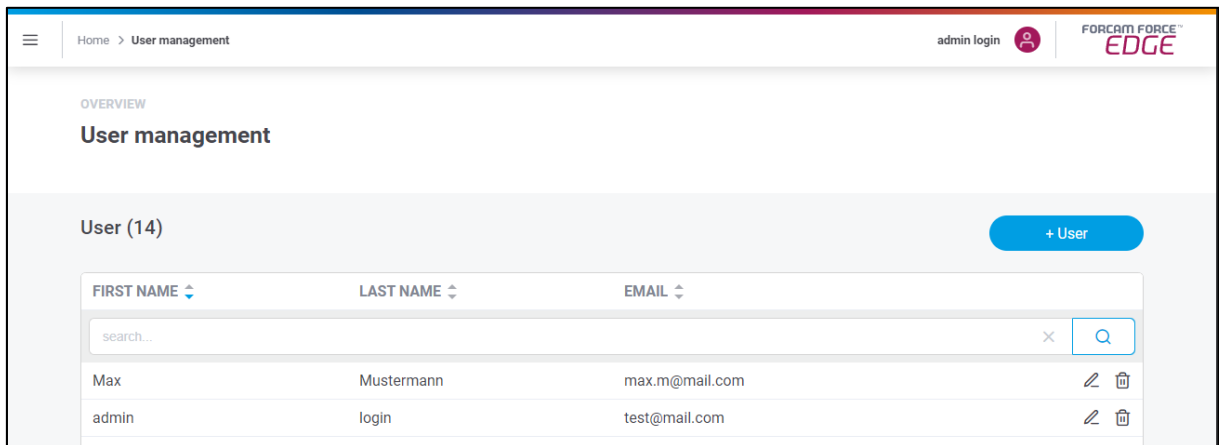


**Fig. 3: Calling up the Home section**

## 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 asset, 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.



**Fig. 4: FORCAM FORCE EDGE user administration with 2 users**

### To create a new user:

1. Click on **+ 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.

## Basic settings

**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.
Monitoring page	The user can view the monitoring area from the EDGE configuration. Changes cannot be made.
Restart Edge node	The user can restart an Edge node.
Supplied master data	The user can transfer asset master data from a third-party system to FORCAM FORCE EDGE.
Add nodes	The user can edit, delete EDGE Nodes and make changes in the event configurations.
Configure without template	The user can create assets without the use of MR templates.
Change template-based values	The user can modify MR templates (signals and script).
Configure with template	The user can create assets only by means of MR templates. Changes cannot be made.

Create User

×

Email \*

First name \*

Last name \*

New password \*

Confirm password \*

User rights:

General permissions:

☐ User management

☐ Monitoring page

☐ Restart EDGE node

☐ Supplied Master Data

☐ Configure nodes

Machine configuration permissions:

☐ Configure without template

☐ Change template-based values

☐ Configure with template

Cancel

Save

**Fig. 5: Dialog for creating a new user**

## 5.2 Supplied Master Data

The "Supplied Master Data" is an extension that allows asset master data to be transferred to FORCAM FORCE EDGE from third party applications.

Asset master data can be created via a third-party application such as SAP DMC. To avoid having to create this again in FORCAM FORCE EDGE, the data can be transferred to EDGE Configuration via the API interface. This reduces the effort required to create an asset in FORCAM FORCE EDGE and supports consistency or synchronization of asset master data.

- i Supplied master data can be used to create assets with basic information... Connection specific information can be entered through a template or manually. Deleting assets through the API is not possible.

When the master data is sent in the third-party application, EDGE Configuration creates an asset in FORCAM FORCE EDGE through the API. All new assets are displayed in the **Supplied Master Data** page and are initially given the status **New Master Data**. They can be configured ready here and assigned to an Edge node.

OVERVIEW							
Supplied Master Data							
Inbox							
ASSET NAME	ASSET TYPE	ASSET CLASS	MANUFACTURER	MODEL	SERIAL NUMBER	STATUS	ACTION
search...							
CL4FD10	Machine	Additive manufacturing system	FORCAM	3131	4444	<span style="color: green;">●</span> New master data	+
CL4FD9	Machine	Additive manufacturing system	BionticsAstra	77483	1336	<span style="color: blue;">●</span> Updated master data	
CL4FD8	Machine	Additive manufacturing system	BionticsAstra	77484	1337	<span style="color: red;">●</span> Deleted master data	

**Fig. 9: List of assets with Supplied Master Data**

The following master data can be created via the API:

- Asset name
- Asset type
- Asset class
- Asset manufacturer
- Asset model
- Serial Number

### Finish configuring the asset

The supplied master data can be created on an existing edge node. For this purpose, the configuration dialog for adding an asset opens (see section 6.2). The master data received through

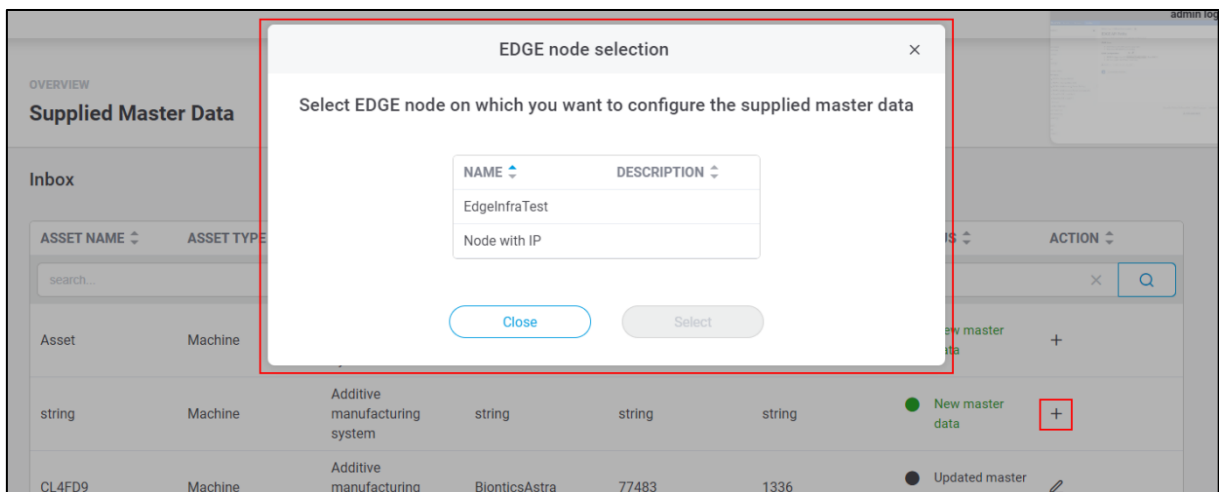


## Basic settings

the API will be prioritized higher: If a template is selected when configuring an asset created through the API, the master data passed through the API is used and that of the template is discarded.

### To create new master data on a node:

1. Click on the plus icon on the right of the desired master data.
2. In the subsequent dialog, select an Edge node on which the master data is to be created.
3. Click on Select.
  - The configuration dialog for adding an asset opens. Master data received through the API is pre-filled.
4. Make further configurations as desired (see section 6.2).



**Fig. 10: Selection of an edge node for the creation of supplied master data**

### Change asset master data via API

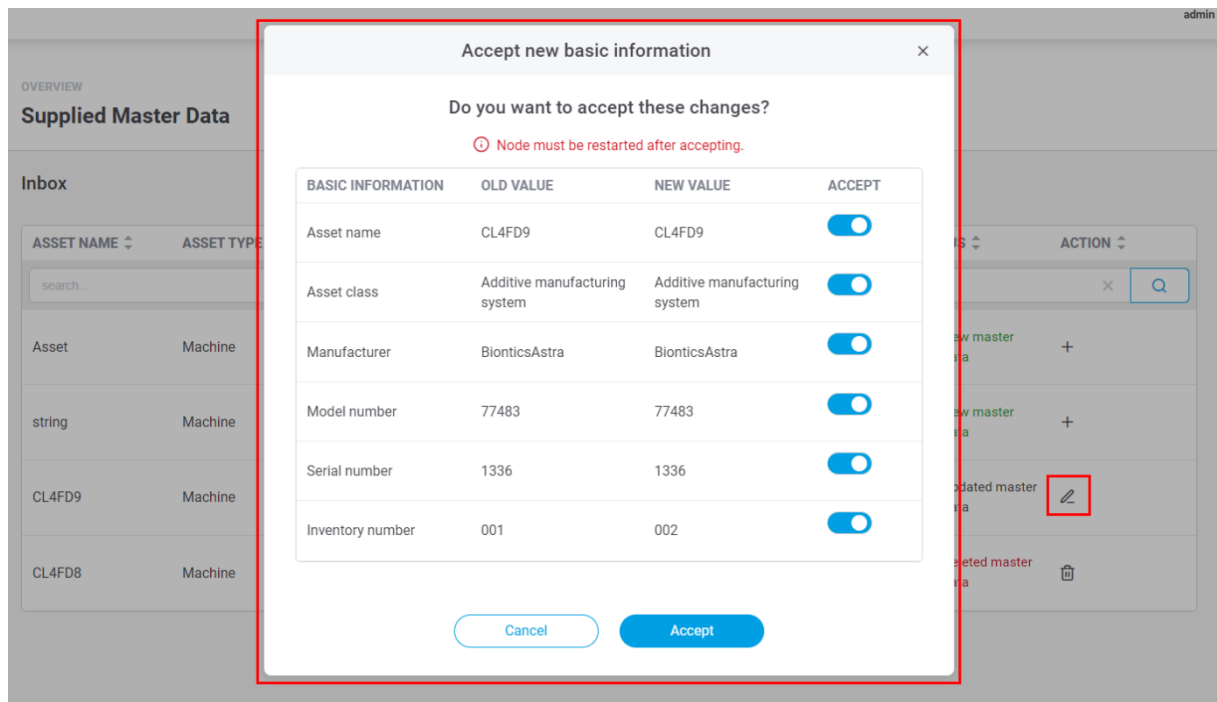
All asset master data can be changed via the API after it has already been initially sent. If the third-party application adjusts the master data, the status in the table of supplied master data changes to Modified master data. In addition, the user of FORCAM FORCE EDGE is informed by a message on the start page that master data has been changed.

The pencil icon can be used to decide which of the changes should be applied.

### To apply changes to master data:

1. Click on the pencil icon to the right of the desired master data.
  - The subsequent dialog lists all changes that have been made to the selected master data.
2. Deactivate the checkbox behind the desired data whose changes are *not* to be applied. By default, all switches are activated.
3. Click on **Accept**.
4. Result
5. Restart the corresponding node.

## Basic settings



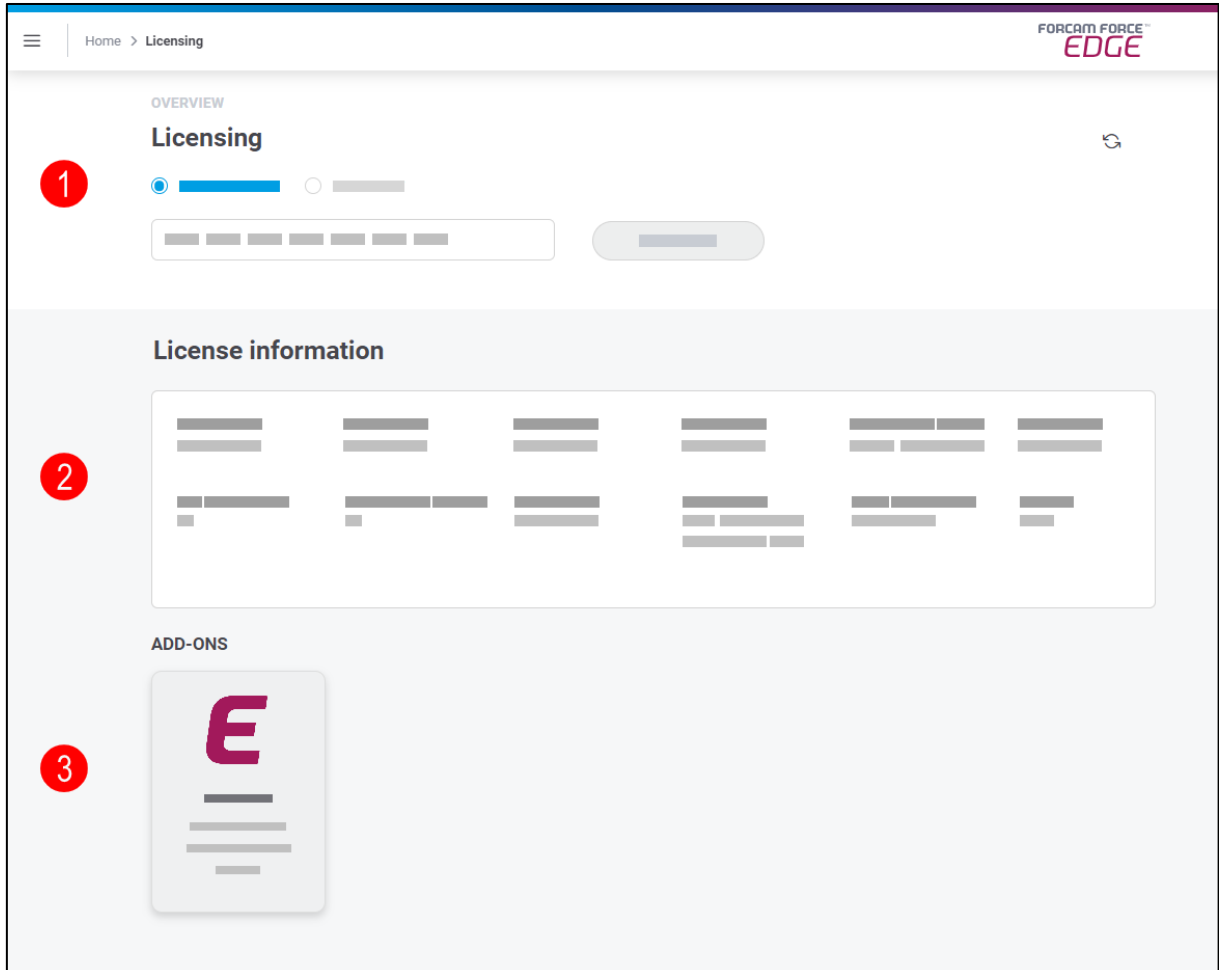
**Fig. 11: Confirmation of asset master data changes**

### Mark asset for deletion via API

When an asset is deleted in the third-party application, it receives the status **To be deleted** under the **Supplied Master Data** page. When you confirm the deletion in the Edge node, the asset is removed from the node as well as from the table of the page.

## 5.3 Licensing

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



**Fig. 12: 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 assets, maintenance, validity and other data.
- (3) All booked add-ons are listed here.

## 5.4 Download area

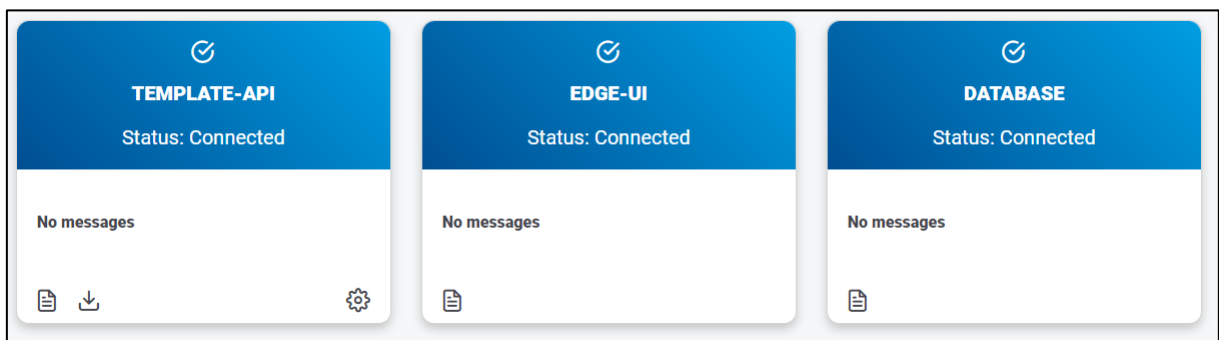
The current FORCAM FORCE EDGE documentation can be downloaded in several languages via the **General** tab. The manual and a product description are available. The manual is this document with detailed configuration instructions. The product description is a shorter document describing only the function 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 an asset via the corresponding plugin. It is located between the EDGE MDC layer and the asset and enables the bi-directional communication.

## 5.5 Monitoring

The monitoring in the home menu is used to monitor the edge configuration. The monitoring of the edge components is displayed on a separate page and is described in chapter 7. However, the structure of the monitoring tiles is the same.

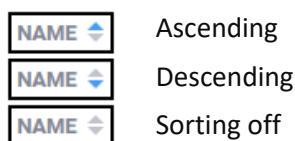
The following tile monitors the status of the transmissions of templates via the API. It specifies all the information that is logged in the process.



**Fig. 13: Monitoring template transmissions via the API**

## 5.6 Table Sorting







Most pages in FORCAM FORCE EDGE display data in the form of tables. To offer the familiar ease of use you know from other table tools, the sorting function of columns has been used here as well: You can sort the columns alphabetically ascending or descending.



**Fig. 14: Alphabetical sorting of columns**

## Basic settings

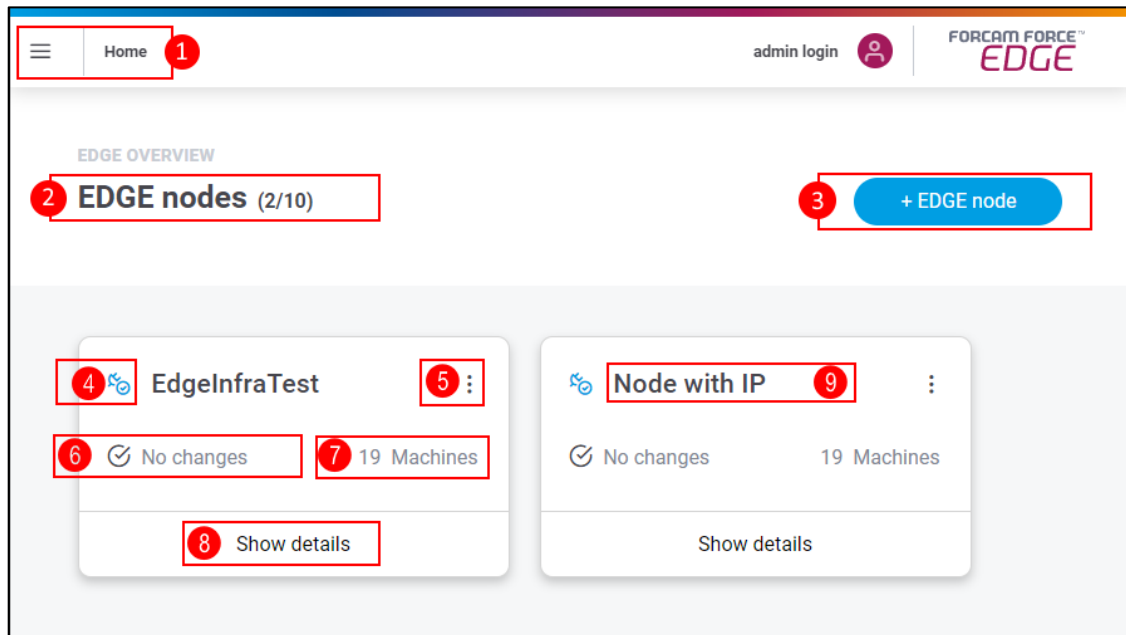
Columns that relate specifically to DNC and MDC specify a status instead of a string. The sorting arranges the statuses alphabetically and additionally groups them by content.

NAME 	MDC STATUS 
search...	
A_Test_DZ	 Connected
Enisco_Soft_PLC	 Connected
Reporting_Machine	 Connected
Script_Vorlage	 Connected

**Fig. 15: Alphabetical sorting and content grouping**

## 6 EDGE Configurations

The configuration of an edge node as well as an asset 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.

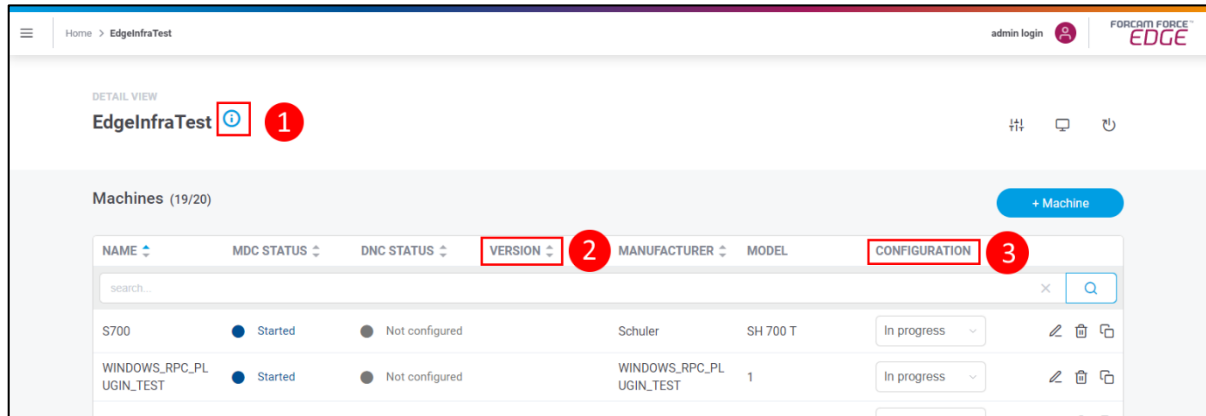


**Fig. 16: FORCAM FORCE EDGE entry and overview page**

- (1) FORCAM FORCE EDGE home menu
  - User Management
  - Supplied master data
  - Licensing
  - Profile
  - Download Area
  - Monitoring
- (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) Status of the EDGE node
- (5) Node settings menu:
  - Edit
  - Delete
- (6) Change display of the EDGE node if necessary, display that the node must be restarted
- (7) Number of connected assets
- (8) More detailed node information:
  - List of all connected assets and their status
  - Option to add a new asset
  - Monitoring connected assets

## EDGE Configurations

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



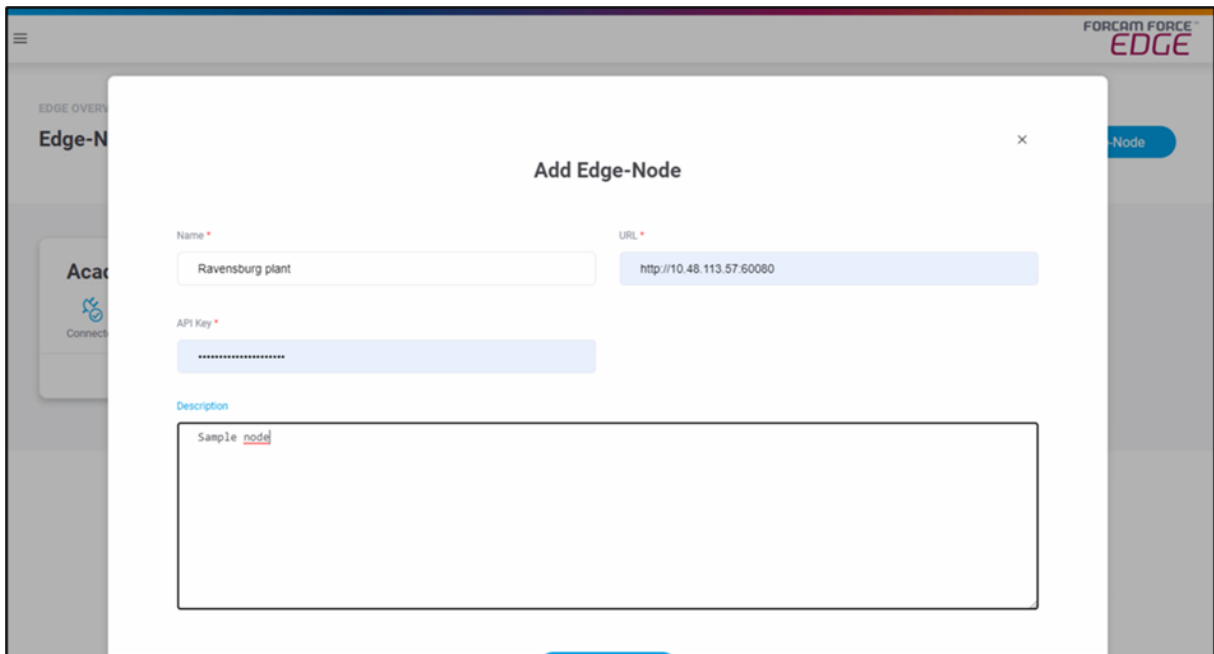
**Fig. 67: Asset overview as next page after clicking on "EDGE Node"**

1. The **i**-icon **i** can be used to display additional information about the Node.
2. The **VERSION** shows the latest implementation of the template.
3. **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 asset 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 asset workload is distributed practically.

- i** 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.



**Fig. 78: Dialog for adding a new node**

**To add a new EDGE node:**

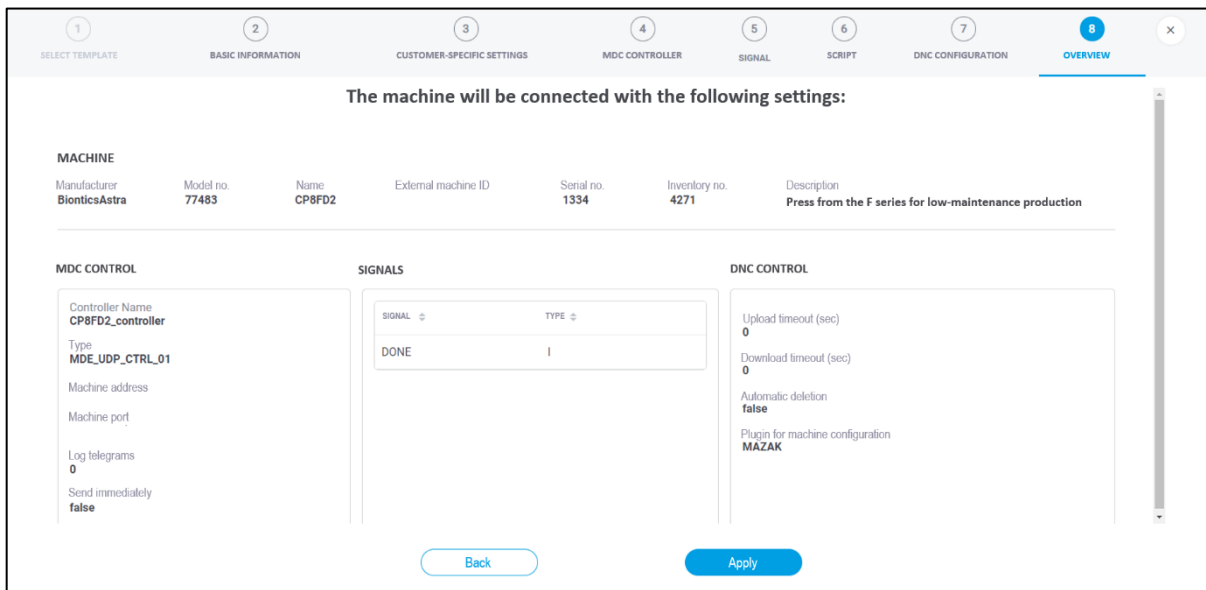
1. Click on **+ EDGE node**.
2. Fill in all mandatory fields in the next dialog:
  - Name:  
Appears in the node overview as the node title
  - URL:  
Consists of http + IP address + host. Only one edge node can be created per URL.
  - API key:  
Password that was assigned during the initial node installation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



### 6.2 Add asset

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

- ❗ Negative values are not permitted in the asset 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 an asset that has already been configured.



**Fig. 19: Dialog for configuring an asset in FORCAM FORCE EDGE**

#### To add an asset:

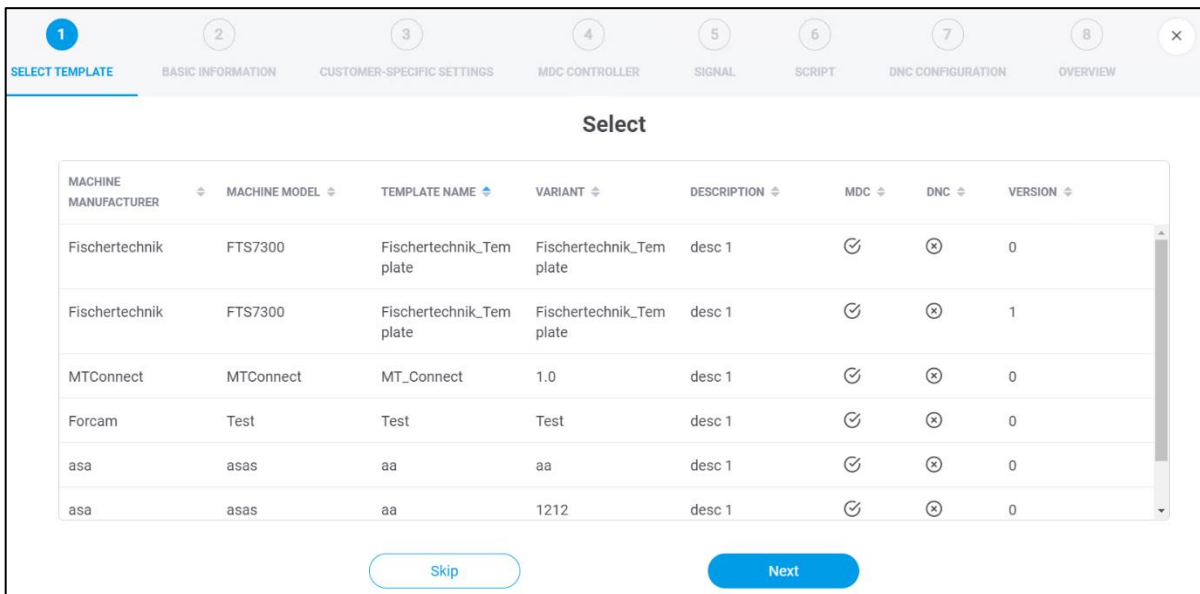
1. Click **+ Asset** in the Edge overview.
  - ➔ The subsequent dialog guides you through the following eight steps for configuring an asset.

### 6.2.1 ① Select template

In FORCAM FORCE EDGE, multiple assets of the same type do not have to be completely reconfigured every time: Once an asset has been configured, it can be entered as a template in the Asset Repository and will then be offered for the next asset connection in this mask. If the template is selected at this point, all settings are automatically used for this asset and all configuration fields that are not asset-specific are pre-filled. Only asset-specific (e.g. serial number) and information specific to the connection (e.g. IP address or port of the asset 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 extension is used. If no template is configured or MR is not in use, the asset connection will start with step ②.



MACHINE MANUFACTURER	MACHINE MODEL	TEMPLATE NAME	VARIANT	DESCRIPTION	MDC	DNC	VERSION
Fischertechnik	FTS7300	Fischertechnik_Tem plate	Fischertechnik_Tem plate	desc 1	☑	⊗	0
Fischertechnik	FTS7300	Fischertechnik_Tem plate	Fischertechnik_Tem plate	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

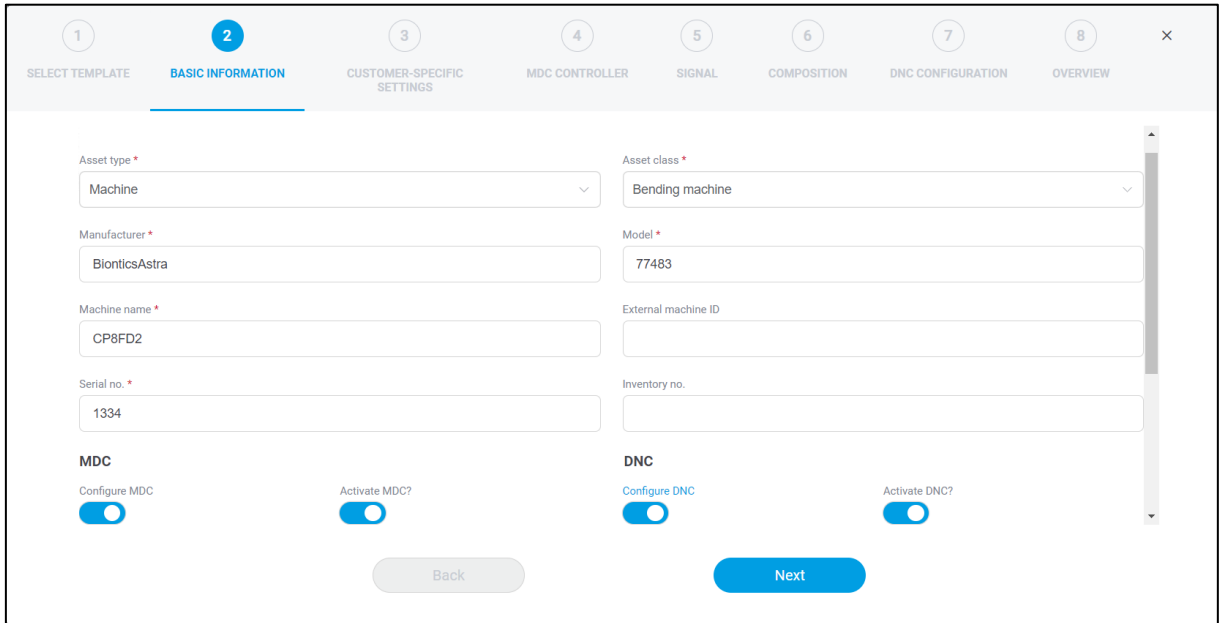
**Fig. 20: Add asset - select template**

1. Select desired template for asset binding in the list.
2. Click on **Next**.

- ❗ If you do not use a template for asset binding, click **Skip**.

## 6.2.2 ② Basic information

This is where basic information of the asset 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 asset and transmitted or written to it. NC files are transferred to the asset via the DNC controller.



**Fig. 21: Add asset - Basic Information**

1. Select **asset type** and **asset classification**.
2. Enter **manufacturer**, **model**, **asset name** and **serial number**.
3. Optional: Enter additional information as desired.
4. Select the **Configure MDC** and/or **Configure DNC** checkbox.
5. Optional: Determine whether MDC or DNC should be activated initially.
6. Optional: Activate **Data Lake**.
7. Click **Next**.

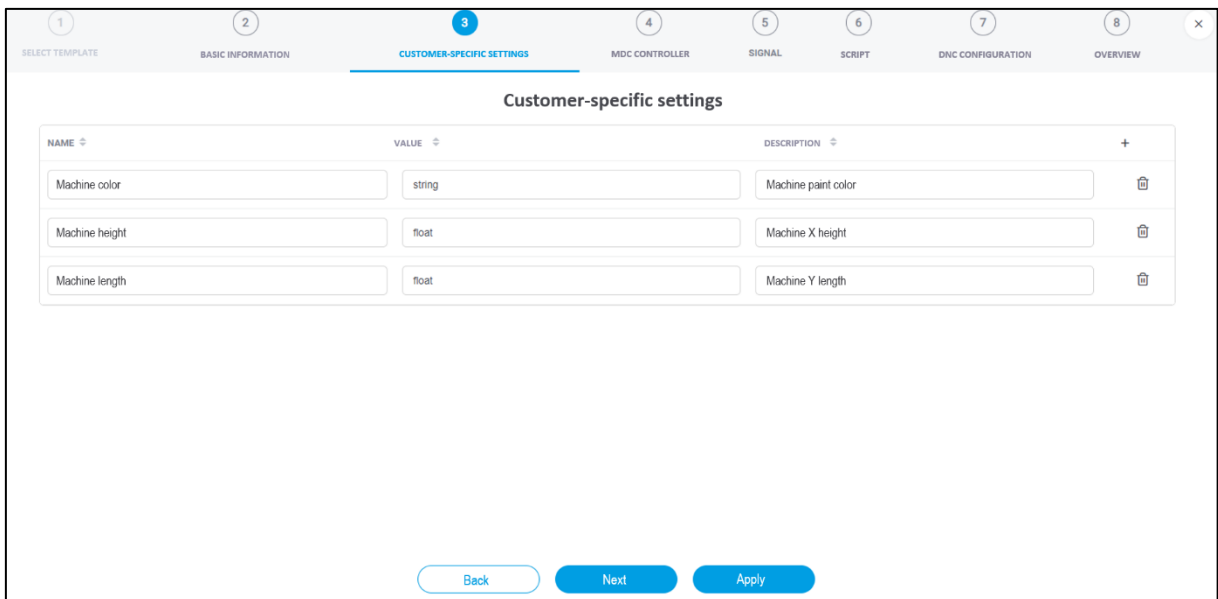
### 6.2.3 ③ Customer-specific settings

This enables individual, plant-specific information to be added to an asset to further supplement the asset'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 asset data to help accurately locate the asset in the event of a malfunction.

 This step is optional.



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

Back Next Apply

**Fig. 22: Add asset – Customer-specific settings**

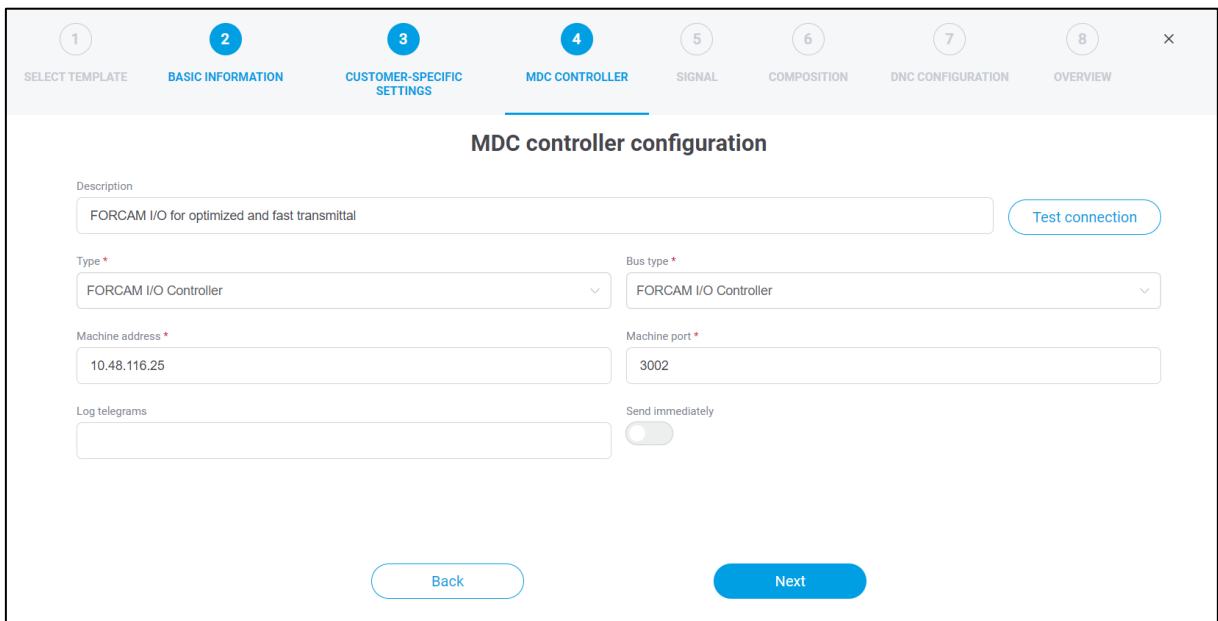
1. Click on the + icon.
2. Enter the desired parameters.
3. Click on **Next**.

## 6.2.4 ④ MDC controller

Option to configure an MDC controller. Specifies the type and method of connection to the asset. 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.

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



**Fig. 238: Add asset - MDC controller**

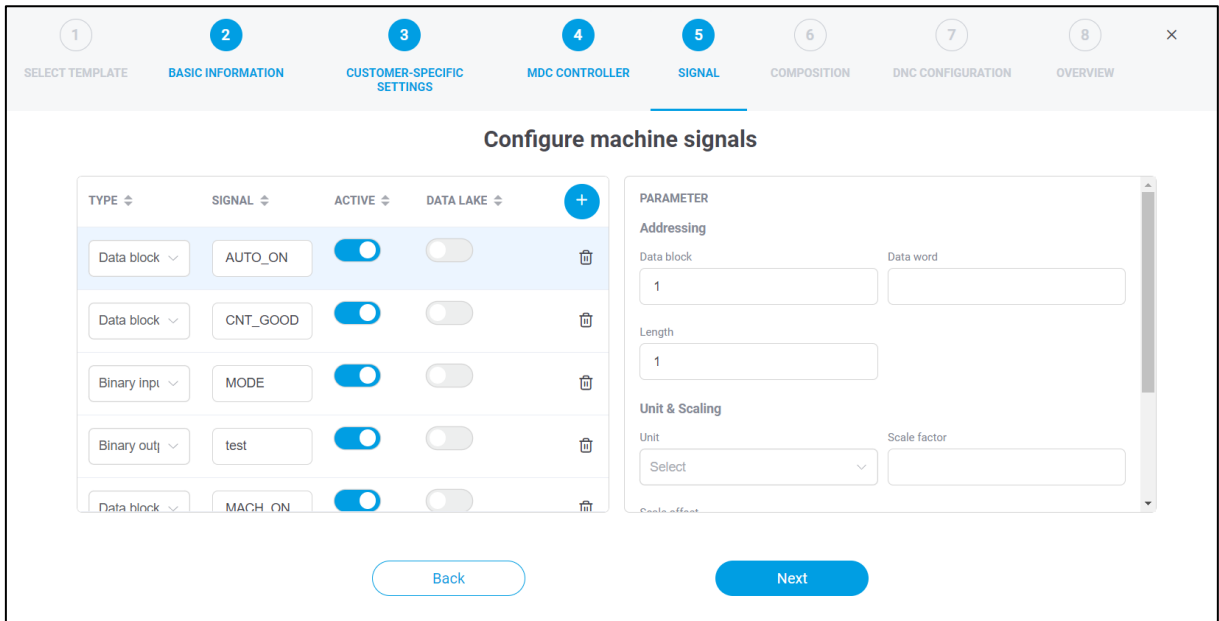
1. Optional: Enter the description of the control.
2. Select the **control type**.
- ➔ Additional configuration parameters appear depending on the selected type.
3. Select **bus type**.  
The possible selection depends on what type of control was previously selected.
4. Adjust the configuration according to the controller.
5. Optional: Checking the connection to the machine.
6. Click on **Next**.

**i** 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

This step defines which signals are read out from the controller. Depending on the configuration of the MDC control from step ④, different listings of the signal types are displayed. The Data Lake can be used to record and save all data. Data Lake storage can be switched on & off per signal. Units can be recorded on individual signals (e.g. degrees Celsius or liters per minute), and furthermore scaling factors can be defined.

**i** If the **ACTIVE** switch for the signal is deactivated, it cannot be used in the ⑥ script.



The screenshot displays the 'Configure machine signals' window. At the top, a navigation bar shows steps 1 through 8, with step 5 'SIGNAL' currently selected. The main area contains a table with the following columns: TYPE, SIGNAL, ACTIVE, and DATA LAKE. There is a '+' icon to add new signals and a trash icon to delete existing ones.

TYPE	SIGNAL	ACTIVE	DATA LAKE
Data block	AUTO_ON	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Data block	CNT_GOOD	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Binary inpt	MODE	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Binary outq	test	<input checked="" type="checkbox"/>	<input type="checkbox"/>
Data block	MACH_ON	<input checked="" type="checkbox"/>	<input type="checkbox"/>

At the bottom of the table are 'Back' and 'Next' buttons. To the right of the table is a 'PARAMETER' configuration panel with the following sections:

- Addressing:** Data block (input: 1), Data word (input:).
- Length:** (input: 1)
- Unit & Scaling:** Unit (dropdown: Select), Scale factor (input:).
- Scale offset:** (input:)

**Fig. 24: Add asset - Signal**

1. Click on the + icon.
2. Select the **type**, enter the **signal name** and optionally activate the **Data Lake** switch.
3. Specify plugin-specific **signal parameters**.
4. Optional: **enter units & scaling** and **description**.
5. Click on **Next**.

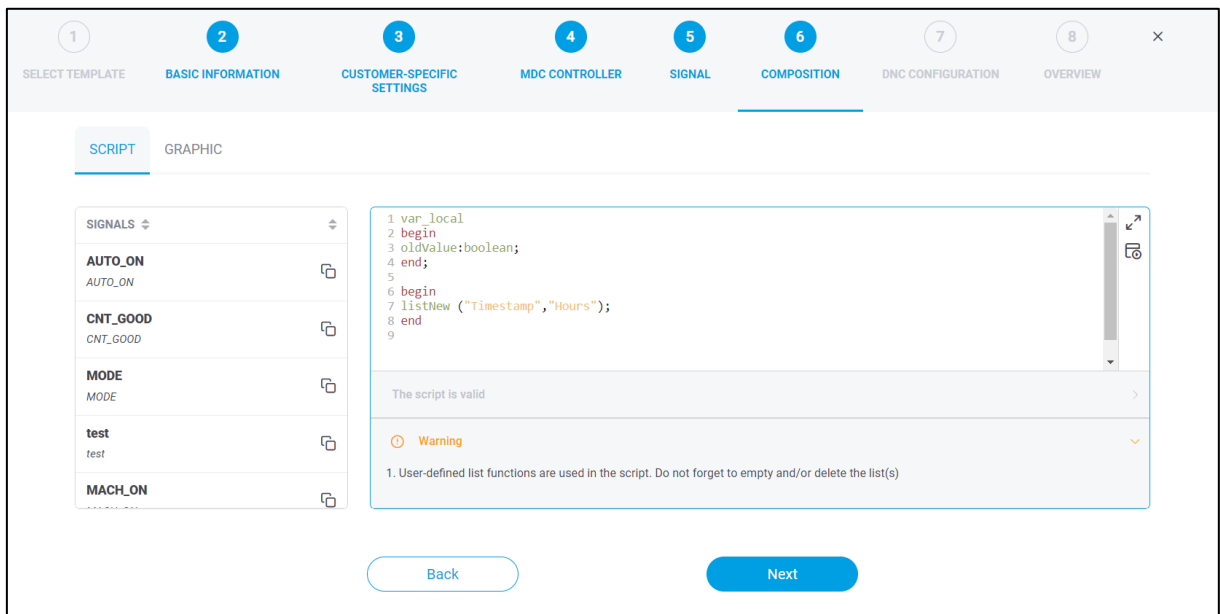
### 6.2.6 ⑥ Script

In this section, the acquired signals are interpreted, from which logical conclusions can be drawn regarding the behavior of the asset. This results, for example, in measured values, information for maintenance and different production statuses. The left-hand area of the screen lists all the signals configured in step ⑤. The central area in the middle is the input field for a script through which the actual asset logic is defined

❗ In the appendix you can find examples of the script as well as their functions (see chapter 8.6 resp. 8.7).

❗ For a listing of form elements and operators in scripts, see the Scripting Language Guide.

⚠ When using a list, do not forget to empty it (see Fig. 20).



**Fig. 25: Add asset - Script**

1. Enter the desired script into the central input field.
2. Optionally, run the script using the play icon at the top right to check its validity.
3. Click Next.

⚠ Only by running an error-free script can you proceed to the next step.

ⓘ In the left pane, the selected signal can be copied by the icon and pasted into the script. This way it is not necessary to type the signal variables manually

❗ The script editor can be set to full screen via the maximize icon.

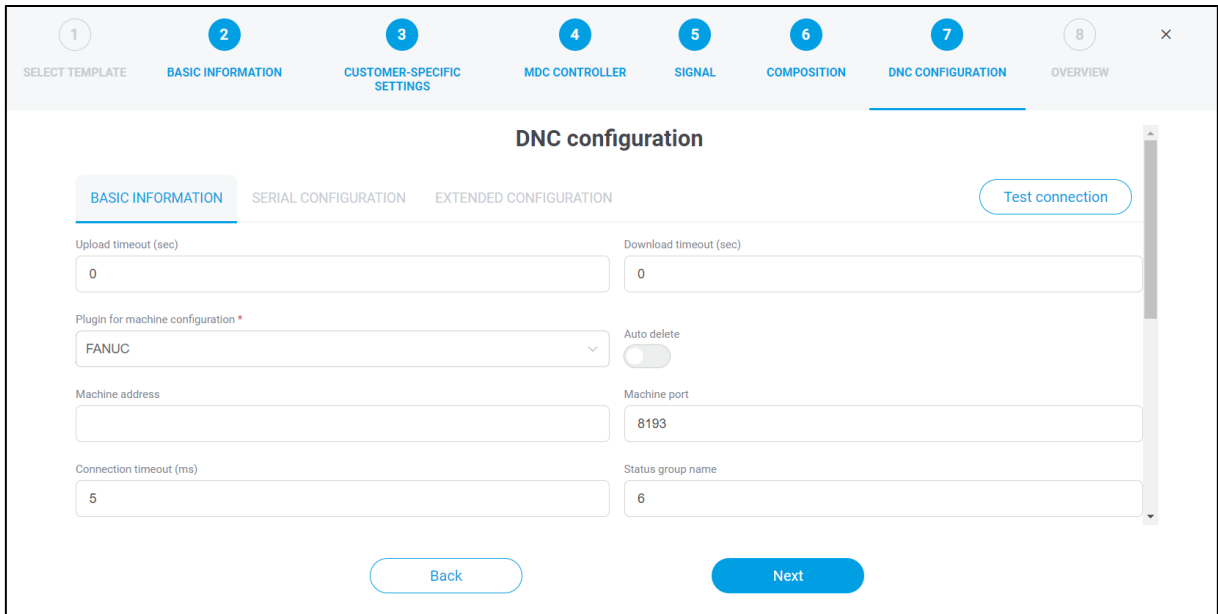
## 6.2.7 ⑦ DNC configuration

Option to configure a DNC control. Determines the way in which an NC file is to be transferred to the asset.

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.

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

**i** After the DNC-Configuration the connection can be tested.



**Fig. 9: Add asset - DNC configuration**

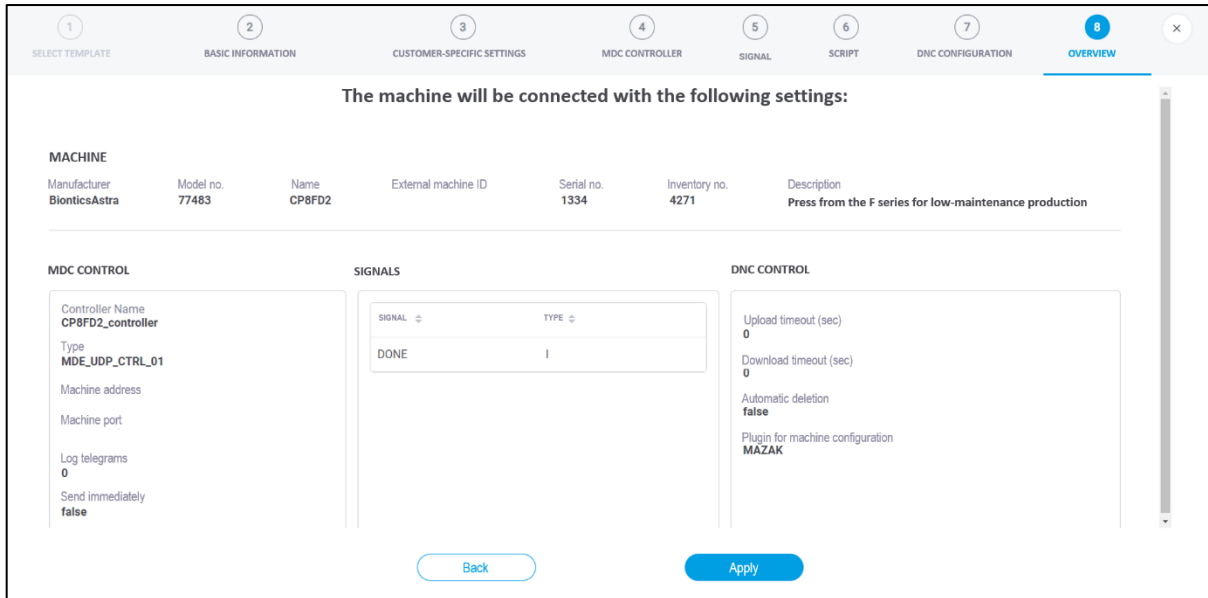
1. Optional: Enter the **Upload** and **Download timeout**.
2. Select **plugin for asset configuration**.
  - ➔ Additional configuration parameters may appear in further tabs depending on the selected plugin.
3. Optional: Set automatic deletion.  
If enabled, the NC file is automatically deleted from the asset after it has been read.
4. Enter additional configuration parameters depending on the selected plugin in the remaining tabs.
5. Verify the connection to the asset.
6. Click on **Next**.



## EDGE Configurations

### 6.2.8 ⑧ Overview

A summary of the previous configuration from all steps and a list of all defined signals. After confirming, the asset is mapped with the specified configuration and is therefore digitized. The configured asset appears under the specified name in the overview (see Fig. 617).



**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

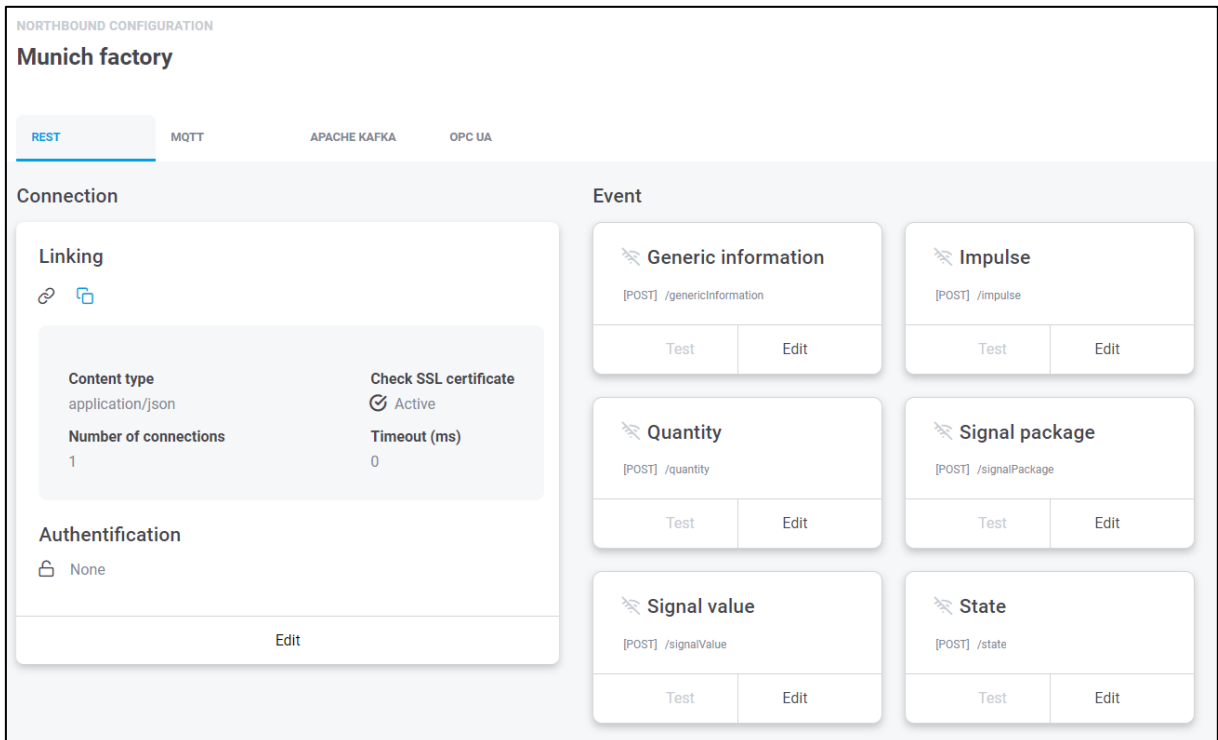
Plugin for machine configuration  
MAZAK

Back Apply

Fig. 107: Add asset - Overview

## 6.3 Northbound configuration

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



**Fig. 28: Northbound 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.

For each type of event there is a standardized **Event**. For example, the **Quantity** event type sends the quantity produced by the asset. 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:

```
{
  assetId: $assetId$
  assetName: $assetName$
  externalAssetId: $externalAssetId$
  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 asset ID or the timestamp formatted in UTC. Section 8.7 lists and explains all available script functions.


## EDGE Configurations

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 **assetID** (asset ID) or **Value** can be entered to generate and execute the signal as an example without influencing the actual asset 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 four technical options for supplying signals and events from an EDGE node to a third-party application.

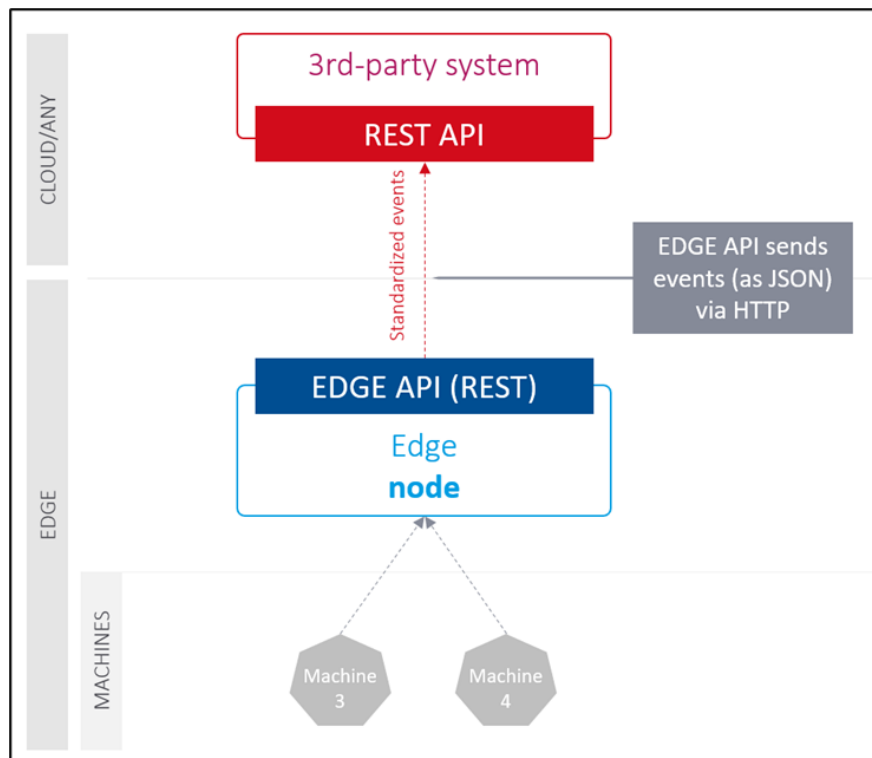
-  The supply can be configured in the EDGE node itself.

#### 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 authentication: 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.

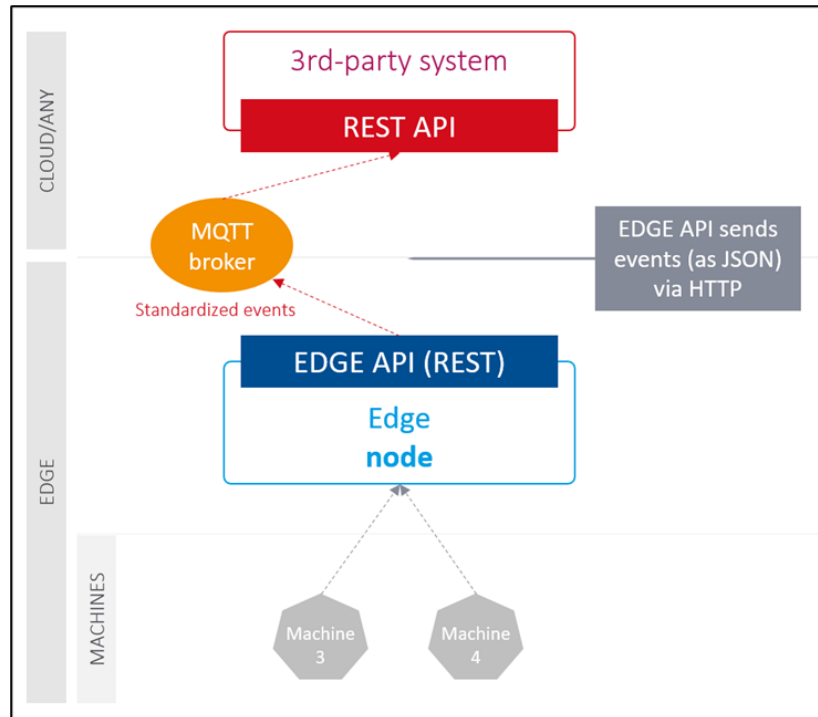


**Fig. 29: Communication with superordinate systems via HTTP/REST**

## EDGE Configurations

### MQTT messaging

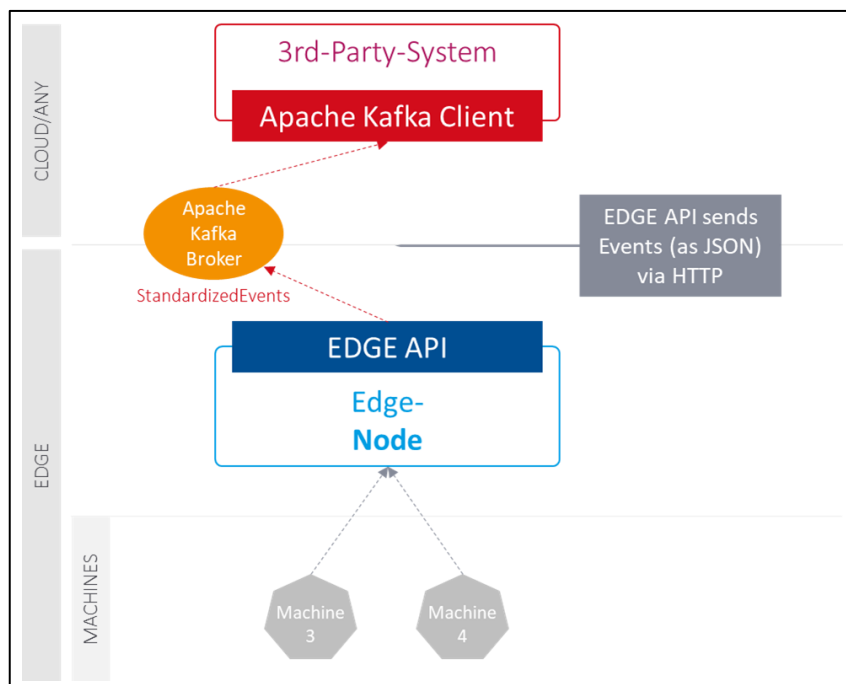
Any MQTT broker can also be used to supply the data, if provided by customers or partners.



**Fig. 30 Communication with superordinate systems via MQTT**

### APACHE KAFKA

The third-party system can be powered by Apache Kafka support if it is provided by the customer or partner.



**Fig. 31: Communication with superordinate systems via Apache Kafka**

## OPC UA

FORCAM provides an OPC UA server with the functionality "Data Access". Through this extension, various asset data is provided via the defined OPC UA interface. The information models are dynamically prepared based on the existing configured assets in the edge node.

Using the specified URL, the user connects to the server to retrieve the desired data. The necessary client for the retrieval is assumed.

It is not only possible to retrieve the current values of an event or signal, but also the history. To be able to process these historical data sets, FORCAM FORCE EDGE supports the **Historian** functionality of OPC UA. The Historian acts as a data logger with SQL databases. It logs historical data and can additionally be used as a gateway to access real-time data from all underlying OPA UP servers.

In FORCAM FORCE EDGE, the user has the option to assign login credentials. This data must then be entered correctly when connecting to the server via the client.

**NORTHBOUND CONFIGURATION**

**Munich factory**

REST



MQTT

APACHE KAFKA


**OPC UA**

**Connection**

**Server URL**

 `opc.tcp://10.10.4.1:4840` 

**Server Security**

 None

Edit

Fig. 32: Event Configuration OPC UA

### 6.3.2 Data & documents from superordinate system to EDGE

Via the EDGE API, the FORCAM FORCE EDGE can be supplied with data and documents.

Technically, the transmission of NC files is possible via **HTTP/REST**. Writing business parameters and signal values is also possible via **MQTT** in addition to **HTTP/REST**.

The following interfaces are provided:

**Table 2: Interfaces for transferring data and documents**

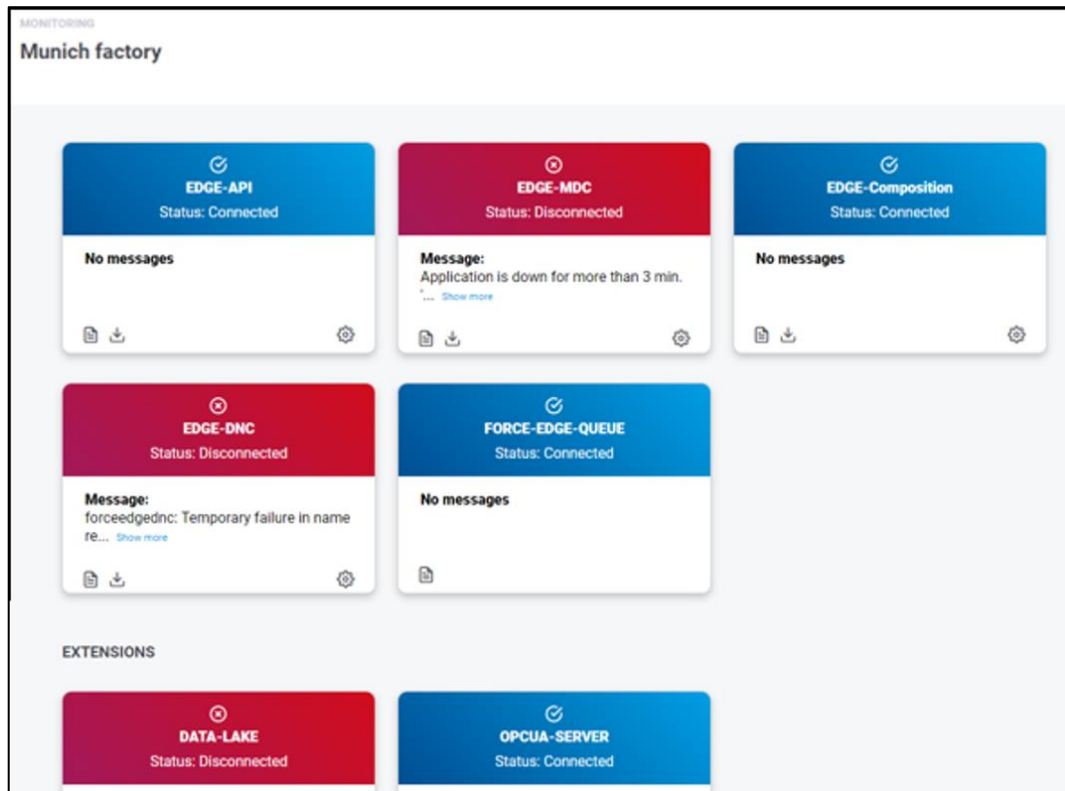
Interface	Description
<b>Transfer of process and reference parameters</b>	These business parameters can be used in the EDGE Composition Layer to supplement standardized events (e.g. order number or cycle time).
<b>Transfer of signal parameters</b>	Parameter values for specific signals can be transferred. These are written directly to the asset control.
<b>Transfer of documents</b>	NC programs can be transferred, which are also transferred to the asset 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 asset (see Fig. 67).
2. In the subsequent dialog determine if **REST** or **MQTT** or **KAFKA** 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.

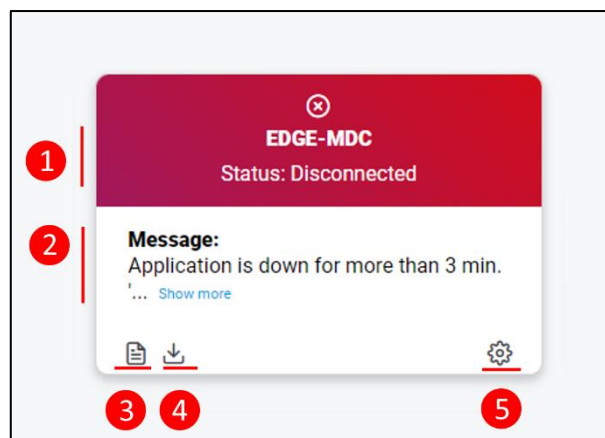
## 7 Monitoring

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 asset overview (see Fig. 67).



**Fig. 33: Monitoring in FORCAM FORCE EDGE**

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



**Fig. 34: „EDGE-DNC“ component in the Monitoring page**

(1) Current status of the component

## Monitoring

---

- (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
- (5) Setting the log-level.



## 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. )
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
2023-01-26	230127	1	Added sections 5.2, 5.5, 5.6. and in section 6.3.1 the content MQTT and KAFKA

## Annex

<b>2023-01-26</b>	230127	1	Updated smaller content in section 4 and 6.2.4 – 6.2.7
<b>2023-01-26</b>	230127	1	Updated screenshots in section 1-3 and 5-7
<b>2023-02-01</b>	230127	1	Updated section 6.3.2

## 8.2 Document conventions

**Table 4: Fonts, formatting and characters used**

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

## 8.3 Abbreviations and terms

**Table 5: Abbreviations and terms used**

Abbreviations	Explanation
<b>Apache Kafka</b>	Apache Kafka is a distributed messaging system that uses the publish-subscribe method.
<b>Asset</b>	Generic term for all objects that FORCAM FORCE EDGE can connect (e.g. machines, sensors, databases, IT systems, etc.).
<b>Brownfield</b>	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.
<b>CP</b>	Communication processor
<b>DNC</b>	Distributed Numerical Control: NC systems that are connected to a computer. The individual systems can be centrally supplied with NC programs and then coordinated.
<b>IT</b>	Information technology
<b>Asset</b>	In FORCAM FORCE EDGE, the asset 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.
<b>MDC</b>	Asset data connection (asset data collection)
<b>MQTT</b>	Message Queuing Telemetry Transport: Open network protocol for asset-to-asset (M2M) communication that enables transmitting telemetry data in the form of messages between devices, despite high delays or network limitations.
<b>MR</b>	Asset Repository
<b>Northbound</b>	A northbound interface communicates with a higher level element in a particular network component.
<b>OT</b>	Operational technology
<b>POST</b>	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.
<b>PUT</b>	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.
<b>REST</b>	Programming paradigm for distributed systems (collection) of independent computers that present themselves to the user as a single system.)
<b>RESTful API</b>	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.
<b>Signal</b>	Values read from the asset control, such as temperature, pressure or certain statuses.
<b>Southbound</b>	An equivalent to the northbound interface, a southbound interface communicates with lower level components.
<b>SPS</b>	Programmable Logical Control
<b>UTC</b>	Coordinated Universal Time
<b>Wildcard</b>	Placeholder for other characters.

## 8.4 List of supported plugins

### MDC Plugins

Table 6: List of all supported asset 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/

## Annex

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 asset 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

## 8.5 Standardized events

**Table 8: Events and their function in FORCAM FORCE EDGE**

Event type	Values	Function
<b>General Information</b>	<ul style="list-style-type: none"> <li>Asset ID</li> <li>Asset name</li> <li>External asset ID</li> <li>Reference (any)</li> <li>Timestamp</li> <li>Type (any)</li> <li>Value (any)</li> </ul>	Any information
<b>Impulse</b>	<ul style="list-style-type: none"> <li>Asset ID</li> <li>Asset name</li> <li>External asset ID</li> <li>Reference (any)</li> <li>Timestamp</li> <li>Count</li> </ul>	For example: hit, shot etc.
<b>Quantity</b>	<ul style="list-style-type: none"> <li>Asset ID</li> <li>Asset name</li> <li>External asset ID</li> <li>Reference (any)</li> <li>Timestamp</li> <li>Amount</li> <li>Unit (optional)</li> <li>QualityDetail (optional)</li> </ul>	Produced quantity
<b>Signal package</b>	<ul style="list-style-type: none"> <li>Asset ID</li> <li>Asset name</li> <li>External asset ID</li> <li>Reference (any)</li> <li>Timestamp</li> <li>ARRAY [SignalName, Value, TimeStampUTC, Unit (optional)]</li> </ul>	Collection of signals (e.g. serial number, pressure and temperature)
<b>Signal value</b>	<ul style="list-style-type: none"> <li>Asset ID</li> <li>Asset name</li> <li>External asset ID</li> <li>Reference (any)</li> <li>Timestamp</li> <li>SignalName</li> <li>Value</li> <li>Unit (optional)</li> </ul>	Temperature, pressure, etc.

## Annex

Event type	Values	Function
<b>State</b>	<ul style="list-style-type: none"> <li>Asset ID</li> <li>Asset name</li> <li>External asset ID</li> <li>Reference (any)</li> <li>Timestamp</li> <li>State (production or downtime)</li> <li>Statuscodes (optional list of statuses)</li> </ul>	Asset state (production, or stoppage)

## 8.6 Script examples

### 8.6.1 Asset status and temperature

The following script sends the status of the asset (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;
        end;

    if( oldTemperature != toString(@|PLC|@:TEMP)) then
        begin
            oldTemperature := toString(@|PLC|@:TEMP);
            sendSignalValue("TEMPERATURE", toString(@|PLC|@:TEMP), "Degrees");
        end;
    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 asset 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;
    // ASSET STATE
    state: number;
    stateOld: number;
    // ASSET 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|@:Asset_ON then
```

## Annex

```

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)
                                     + " Asset_ON: "        + toString(@|PLC|@:Asset_ON)
                                     + " Emergency_ON: "    + toString(@|PLC|@:Emergency_ON )
                                     + " COUNTER: "         + toString(@|PLC|@:Good_count)
                                     + " seconds: "         + toString(seconds);

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

```

```

        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 asset 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      = asset state
// //STATUSCODES = Status reason
// Reg1SEND    = just display holding register
////-----

// VARIABLES
var_local
begin
// GENERAL VARIABLES
    seconds: number;
    logstring: string;
    logstringOld: string;
// SIGNAL VARIABLES
    H10ld: number;
    H20ld: number;
    H30ld: number;
    H10ld: number;
    H50ld: number;
    H10ld: number;
    H10ld: number;
    H80ld: number;
    H90ld: number;
    H100ld: 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
                H10ld := 0;
                H20ld := 0;
                H30ld := 0;
                H40ld := 0;
                H50ld := 0;
                H60ld := 0;
                H70ld := 0;
            end;
        end;
    end;
end;

```

## Annex

```

      H80ld := 0;
      H90ld := 0;
      H100ld := 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( H10ld <> @|PLC|@:H1 ) then
    begin
      //fill lists
      H10ld := @|PLC|@:H1;
      ListAdd("Signals", "H1");
      ListAdd("Values", toString(@|PLC|@:H1));
      H20ld := @|PLC|@:H2;
      ListAdd("Signals", "H2");
      ListAdd("Values", toString(@|PLC|@:H2));
      H30ld := @|PLC|@:H3;
      ListAdd("Signals", "H3");
      ListAdd("Values", toString(@|PLC|@:H3));
      H40ld := @|PLC|@:H4;
      ListAdd("Signals", "H4");
      ListAdd("Values", toString(@|PLC|@:H4));
      //      H50ld := @|PLC|@:H5;
      //      ListAdd("Signals", "H5");
      //      ListAdd("Values", toString(@|PLC|@:H5));
      //      H60ld := @|PLC|@:Reg6;
      //      ListAdd("Signals", "H6");
      //      ListAdd("Values", toString(@|PLC|@:H6));
      //      H70ld := @|PLC|@:H7;
      //      ListAdd("Signals", "H7");
      //      ListAdd("Values", toString(@|PLC|@:H7));
      //      H80ld := @|PLC|@:H8;
      //      ListAdd("Signals", "H8");
      //      ListAdd("Values", toString(@|PLC|@:Reg8));
      //      H90ld := @|PLC|@:H9;
      //      ListAdd("Signals", "H9");
      //      ListAdd("Values", toString(@|PLC|@:Reg9));
      //      H100ld := @|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)

```

## Annex

---

```

//
//
//
//
//
//
+ " Reg 3: "
+ " Reg 4: "
+ " Reg 5: "
+ " Reg 6: "
+ " Reg 7: "
+ " Reg 8: "
+ " Reg 9: "
+ " Reg 10: "

+ toString(@|PLC|@:H3)
+ toString(@|PLC|@:H4)
+ toString(@|PLC|@:H5)
+ toString(@|PLC|@:H6)
+ toString(@|PLC|@:H7)
+ toString(@|PLC|@:H8)
+ toString(@|PLC|@:H9)
+ 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])	Sends 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
Helper	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 asset status.	-
Helper	GetMachineData(ParameterName)	Indicates asset 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	IPADDRESS	The IP address of the Composition.	-
Helper	HOSTNAME	Hostname of the Composition.	-
Helper	SQRT(args)	Root function MATH.	-
Helper	SIN(args)	Sine function MATH.	-
Helper	COS(args)	Cosine function MATH.	-
Helper	TAN(args)	Tangent function MATH	-
Helper	RISINGEDGE(args)	At the beginning the variable is FALSE, the EDGE checks if the values have changed. If this is the case, the variable is corrected to TRUE.	-



Application	Script function Parameters in [...] are optional	Description	Output event
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.	-
Helper	FORMATTIME(timeformatStr, timeOffset, [, timeunit])	<p>Formats the current time with the time unit as one of the following:</p> <p>MILLISECOND SECOND MINUTE HOUR TAG MONTH YEAR MSABSOLUTE (current time)</p> <p>"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.</p>	-

Application	Script function Parameters in [...] are optional	Description	Output event
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	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.	-
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).	-