FORCE EDGE CONNECT

Version 230406

Handbuch

Dokument: Handbuch- FORCE EDGE CONNECT

Freigabedatum: 06.04.2023

Dokumentversion: 1

Autor: FORCAM GmbH

FB_8-41 ERSTELLT AM: 17.05.21 GEÄNDERT AM: 05.04.23 VERSION: V1
FREIGEGEBEN DURCH: HEAD OF TTE AUTOR: TECHNICAL WRITER VERTRAULICHKEITSSTUFE: ÖFFENTLICH

Inhaltsverzeichnis

1 Konzept*

FORCE EDGE CONNECT (im Folgenden nur noch EDGE CONNECT genannt) bietet produzierenden Unternehmen eine Lösung zur digitalen Anbindung ihres heterogenen Maschinenparks. Mit EDGE CONNECT lassen sich nahezu alle Assets digitalisieren, unabhängig von Alter und technischem Stand. Ein Asset ist ein Oberbegriff für alle Objekte, die EDGE CONNECT anbinden kann, wie beispielsweise Maschinen, Sensoren, Datenbaken und IT-Systeme. Dadurch unterstützt FORCAM die digitale Transformation einer Fertigungsanlage im Brownfield-Umfeld.

FORCAM liefert damit ein Produkt, welches die Kernanforderung von Industrie 4.0 durch das Gewinnen von digitalen Informationen aus dem Maschinenpark der Produktion adressiert. FORCAM leistet somit einen maßgeblichen Beitrag zur digitalen Transformation, indem die Kluft zwischen IT (Informationstechnik) und OT (operative Technologie) geschlossen wird.

EDGE CONNECT verschalt die Vielseitigkeit der Asset-Anbindungen und -signale und liefert diese als standardisierte Events an übergeordnete Systeme. Diese können unter anderem ME (Manufacturing Execution)- oder MOM (Manufacturing Operation Management)-Systeme wie beispielsweise SAP DMC/ME oder MII sein. Damit reduziert FORCAM den Aufwand bei der Digitalisierung und schafft eine standardisierte Schnittstelle zum Maschinenpark. Die Anbindung des Assets erfolgt über ein Plug-in-Konzept für die vereinfachte zukünftige Erweiterung. Aktuell werden viele gängige Asset-Hersteller-spezifische (proprietäre) Protokolle unterstützt (wie z. B. HEIDENHAIN, Siemens S7 oder FANUC & Co.) sowie viele gängige Kommunikationsstandards (wie z. B. MTConnect, OPC UA oder MQTT). Für nicht netzwerkfähige Assets steht der FORCAM I/O Controller als separate Hardware zur Digitalisierung des Assets zur Verfügung. EDGE CONNECT wird stetig um Plug-ins erweitert, um den Anspruch zu verwirklichen, jeden Assettypen über die EDGE CONNECT digital abbildbar zu machen.

Aus den Asset-Anbindungen werden unterschiedlichste Informationen gewonnen. Dazu zählen Informationen über den aktuellen Status des angebundenen Assets oder deren Sensormesswerte wie z. B. Temperaturen, Drücke oder Energieverbrauch. Im Brownfield-Umfeld ist es wichtig, nicht nur die Signale abzugreifen und weiter zu reichen, sondern diese auch für die Weiterverarbeitung zu interpretieren. Diese Aufgabe übernimmt der EDGE Composition Layer. Dadurch kann zum Beispiel interpretiert werden, wann sich eine Maschine tatsächlich in Produktion oder im Stillstand befindet. Ein weiterer wesentlicher Bestandteil der Lösung ist der Umgang mit NC-Programmen und die Möglichkeit, diese von und zum Asset zu übertragen.

Die moderne, klar strukturierte Menüführung von EDGE CONNECT erlaubt es, mit den vorhanden Steuerungs- und Signalinformationen schnell und effizient Assets digital anzubinden.

Die Machine Repository-Komponente ermöglicht die einfache Erstellung und Verwendung von Templates. Diese erlaubt es, für Asset-Anbindungen Templates zu definieren oder aus bestehenden Anbindungen abzuleiten und für die Anbindung von gleichen Assettypen zu verwenden. Dadurch wird der individuelle Aufwand für die Anbindung eines Asset noch einmal deutlich reduziert, wodurch die zeit- und ressourcenschonende Umsetzung von Digitalisierungsprojekten ermöglicht wird. Die Template-Struktur sorgt für eine standardisierte Anbindung von gleichen Assets und ermöglicht dadurch die Vergleichbarkeit von Assets des gleichen Typs.

* Aus Gründen der besseren Lesbarkeit wird im Text verallgemeinernd das generische Maskulinum verwendet. Diese Formulierungen umfassen jedoch gleichermaßen alle Geschlechter und sprechen alle gleichberechtigt an.

EDGE CONNECT ist flexibel einsetzbar und kann auf jedes produzierende Unternehmen angewandt werden. Die einzelnen Bausteine der Lösung können in verschiedene Bereiche und Ebenen verortet werden und bringen auf jeder Ebene Vorteile mit sich.

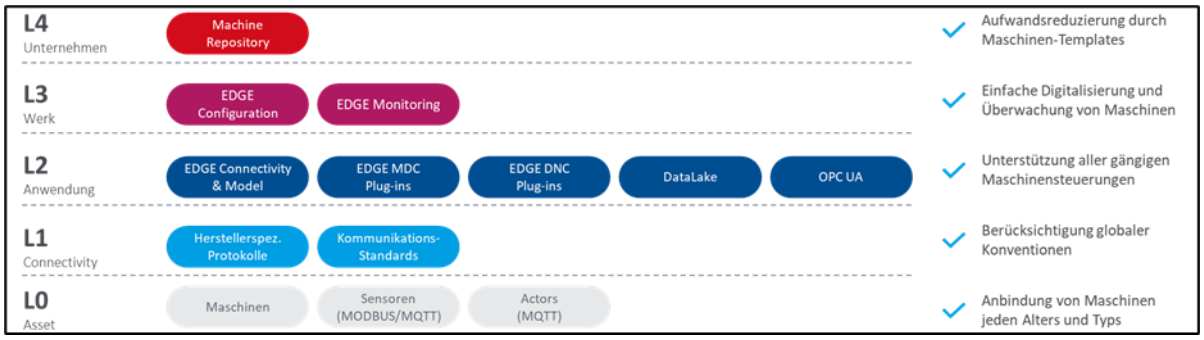


Bild 1: Verortung der Lösungsbausteine von EDGE CONNECT

Bild 1 zeigt die Referenzarchitektur der Open Industry 4.0 Alliance, die auch Grundlage der EDGE CONNECT Architektur ist. FORCAM leistet damit einen maßgeblichen Beitrag zur Digitalisierung in der Industrie und setzt den Fokus dabei auf den Kundennutzen. Die Vernetzung von Hardware durch intuitive und benutzerfreundliche Software zeichnet EDGE CONNECT besonders aus.

2 Systemkomponenten

In diesem Kapitel werden die einzelnen Komponenten von EDGE CONNECT und deren Aufgaben erläutert.

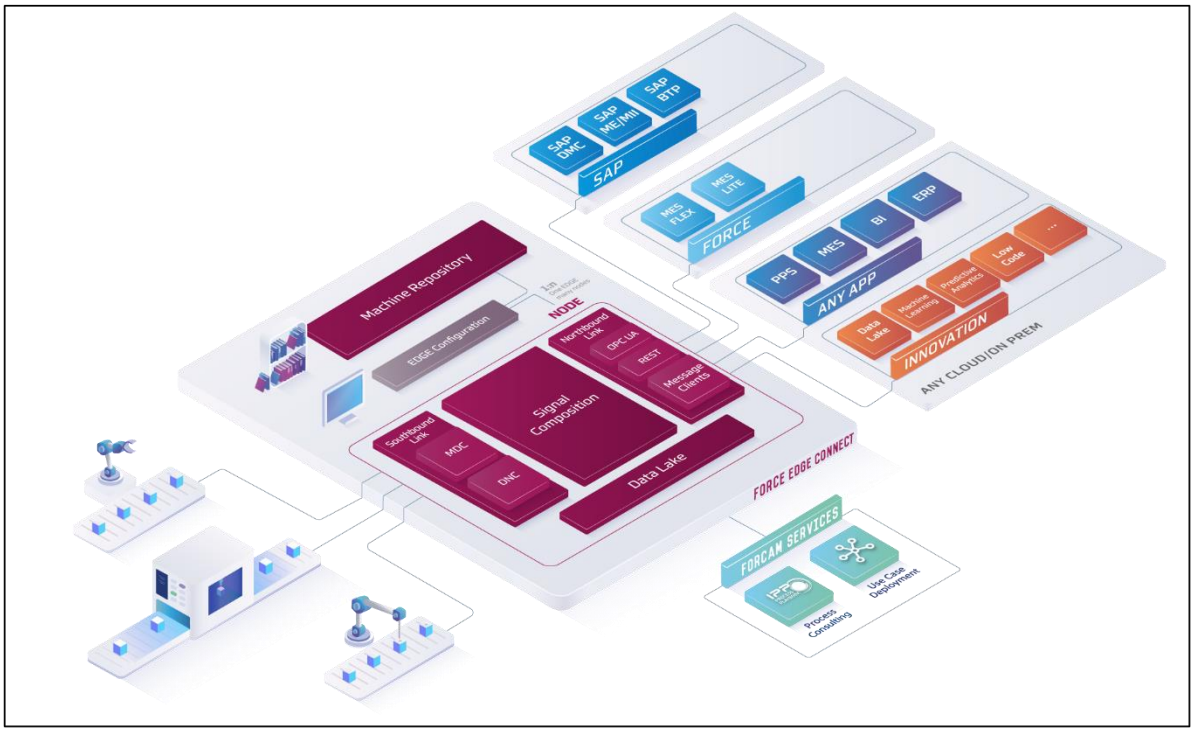


Bild 2: Schematischer Aufbau von EDGE CONNECT

2.1 EDGE Node

Der Node ist das wesentliche Element der EDGE CONNECT zur Asset-Anbindung. Dieser enthält folgende Subkomponenten:

2.1.1 Southbound Link

Der Southbound Link ist für die Kommunikation zwischen dem Asset und der EDGE CONNECT zuständig. Betrachtet man die Infrastruktur, in der die EDGE CONNECT verortet ist, so befindet sich diese oberhalb der Asset-Ebene (Shopfloor). Daher sprechen wir bei der Kommunikation zwischen Assets und der EDGE CONNECT von einer Kommunikation „in Richtung Süden“.
Dabei wird die Kommunikation von den drei Folgenden Komponenten verwaltet:

Plug-ins

Plug-ins im Umfeld von EDGE CONNECT implementieren die Kommunikationsverbindung mit spezifischen Asset-Steuerungen. Außerdem sorgen sie für eine Standardisierung der Daten, wodurch Auswertungen vergleichbarer werden.
Sie erlauben eine direkte Kommunikation mit verschiedenen Asset-Steuerungen, decken aber auch moderne Kommunikationsprotolle wie beispielweise MQTT, OPC UA und viele mehr ab. Das Plug-in-Konzept von EDGE CONNECT wird durch FORCAM stetig weiter ausgebaut.

Die Plug-ins gliedern sich in Plug-ins zur Maschinendatenerfassung (MDC) und für Distributed Numerical Control (DNC).
MDC-Plug-ins beinhalten sowohl solche zum unidirektionalen Auslesen von Asset-Signalen als auch für eine bidirektionale Signalübertragung, also dem Auslesen und Zurückschreiben von Signalen.
DNC-Plugins werden für das Übertragen und Auslesen von NC-Dateien verwendet. Mit ihrer Hilfe werden NC-Programme an das Dateisystem des Assets übertragen oder das an dem Asset aktive Programm abgefragt.

Für die gängigsten Steuerungstypen werden Plug-ins in EDGE CONNECT standardmäßig mitgeliefert. Eine Übersicht der aktuellen FORCAM Plug-ins ist in Kapitel 8.3 aufgelistet.

EDGE MDC Layer

Der EDGE MDC Layer verwaltet die konkrete Anbindung der Assets. Die wesentlichen Elemente sind die Auswahl des geeigneten Plug-ins zur Kommunikation mit der Asset-Steuerung, die Konfiguration der Asset-Stammdaten, die Einstellung der Netzwerkverbindung und die Definition der Asset-Signale. Zudem leitet der EDGE MDC Layer Asset-Signale an den EDGE Composition Layer weiter.

EDGE DNC Layer

Der EDGE DNC Layer verwaltet die konkrete Anbindung von Assets mit einer NC-Versorgung. Die elementaren Bestandteile sind hier die Auswahl des geeigneten Plug-ins zur Kommunikation mit der Asset-Steuerung, die Konfiguration der Asset-Stammdaten, die Einstellung der Netzwerkverbindung und die Konfiguration der DNC-Übertragung.

⚠ Das Bereitstellen, Bearbeiten oder Verwalten von NC-Programmen ist keine Funktion von EDGE CONNECT.

2.1.2 Signal Composition

Der EDGE Signal Composition Layer ermöglicht das Ableiten von logischen Asset-Zuständen. Mit einer Skriptsprache oder einer grafischen Lösung lassen sich aus Signalkombinationen standardisierte Events ableiten. Die Vereinheitlichung der Reportfähigkeit wird hierzu vom Composition Layer ermöglicht. Zusätzlich werden auch Möglichkeiten für individuelle Events geschaffen. Durch die Composition gibt es auch die Möglichkeit, auf Ereignisse zu reagieren und Werte in die Steuerungseinheit des Assets zu schreiben, sofern dies von der Steuerung und dem Protokoll unterstützt wird. Eine solche Composition kann in der EDGE CONNECT entweder über ein Skript oder eine grafische Lösung umgesetzt werden. Letzteres ermöglicht einen leichten Einstieg in die Welt der Signalkomposition.

2.1.3 Northbound Link

Der Northbound Link ist für die Kommunikation zwischen der EDGE CONNECT und einem beliebigen Drittsystem zuständig. Betrachtet man die Infrastruktur, in der die EDGE CONNECT verortet ist, so befindet sich das Drittsystem oberhalb der EDGE CONNECT. Daher sprechen wir bei der Kommunikation zwischen EDGE CONNECT und darüberliegenden Systemen von einer Kommunikation „in Richtung Norden“.

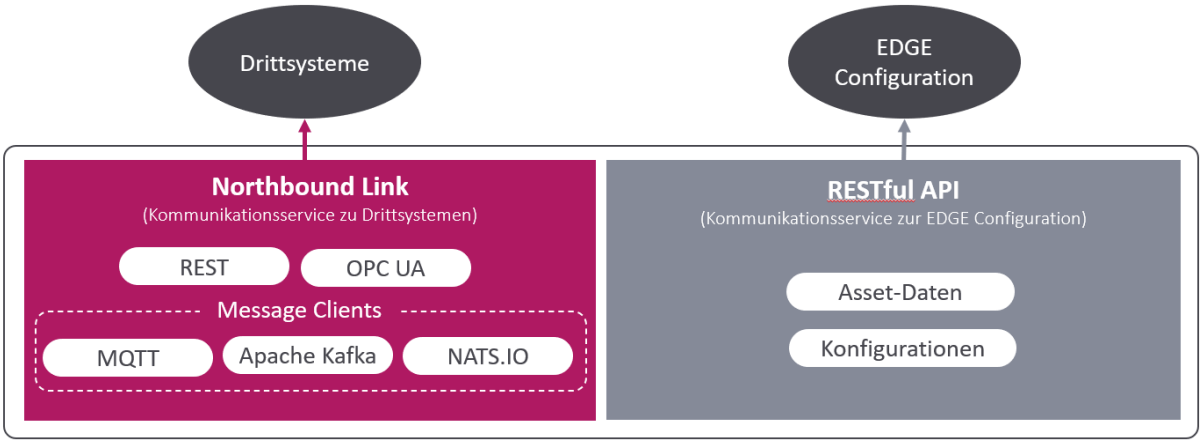


Bild 3: Northbound Link

RESTful API

Über die RESTful API erfolgt der Abruf von Asset-Stammdaten und die Konfiguration der Asset-Anbindungen. Sie ist hauptsächlich für die Kommunikation zwischen EDGE NODE und EDGE Configuration zuständig.

Northbound Link

Der Northbound Link dient zur Weitergabe von Asset-Daten in Form von standardisierten Events an übergeordnete Systeme (3rd-Party). Die Anbindung von übergeordneten Systemen kann über folgende Optionen erfolgen:

- HTTP/REST
- MQTT
- Apache Kafka
- OPC UA
- NATS.io

Der Nachrichteninhalt kann pro Anbindung und pro Event frei konfiguriert werden. Bei der Verwendung von MQTT, NATS.io und Apache Kafka ist ein Broker als Middleware notwendig.
Die EDGE API wird mit vorkonfigurierten Standardevents zur Kommunikation mit der MES- oder ERP-Ebene ausgeliefert. Diese lassen sich bei Bedarf weiter individualisieren.

⚠ Die Middleware muss separat bereitgestellt und eingerichtet werden. Sie ist nicht Bestandteil der EDGE CONNECT.

Data Lake

Um einen digitalen Schatten eines Assets oder einer Steuerung zu erhalten, ist es nicht nur wichtig, die Verbindung zum Asset herzustellen, die Signale zu interpretieren und an andere Anwendungen weiterzugeben, sondern auch die Daten zu speichern. Mit dem Data Lake werden alle Daten auf der Signalebene, der Interpretationsebene und der Event-Ebene gespeichert, einschließlich Konfigurationsänderungen, Schreibvorgängen und übertragenen NC-Dateien. Die Daten werden über die Data Lake API zur Verfügung gestellt. So können KI-Algorithmen, Visualisierungstools, aber auch Audit-Anforderungen davon profitieren.

2.2 EDGE Configuration

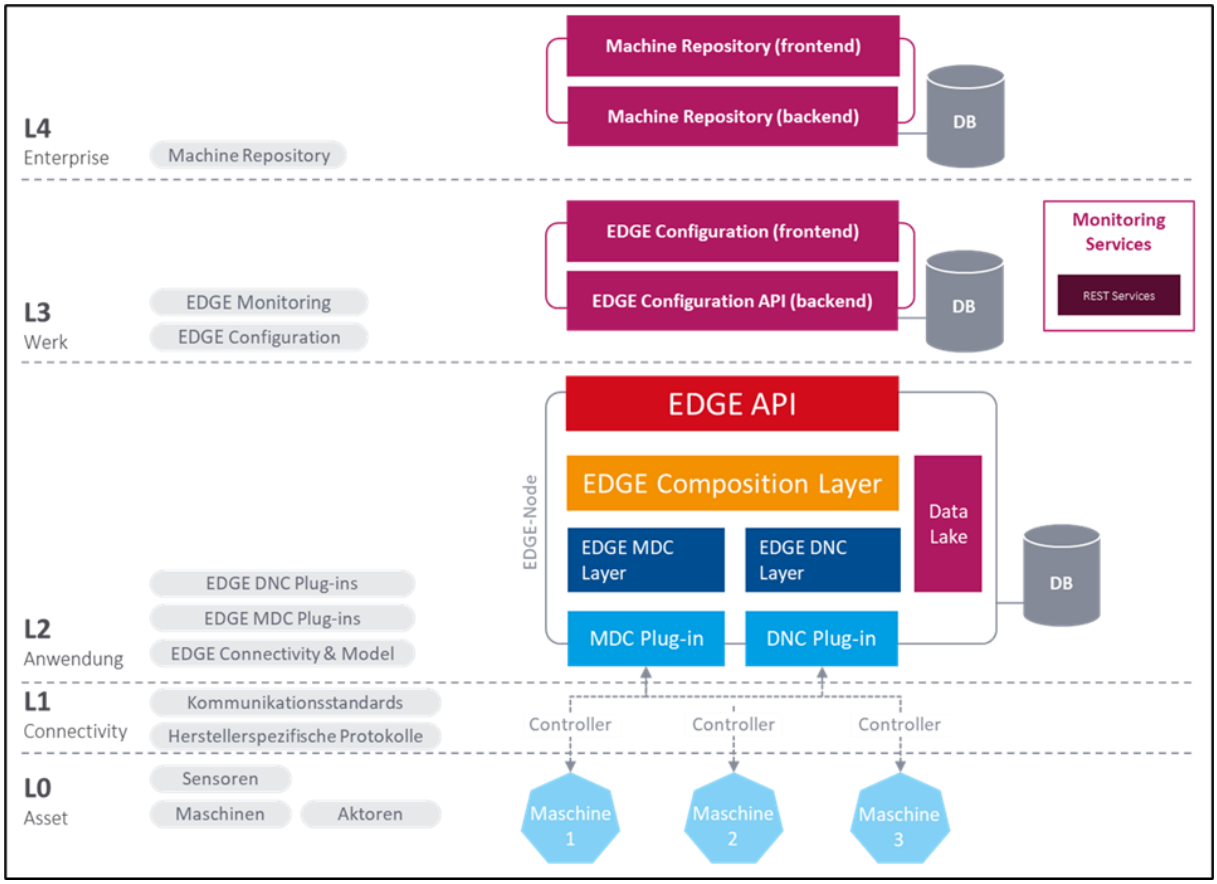
EDGE Configuration ist die Verwaltungsoberfläche für EDGE CONNECT. Mit ihr lassen sich mehrere EDGE-Knoten verwalten. Ein EDGE-Knoten ist die Bündelung der Signalerfassung von mehreren Assets. Je nach Datenmenge werden ein oder mehrere EDGE-Knoten pro Werk eingesetzt. Die Verwaltung der Knoten erfolgt zentral.

2.3 Machine Repository

Das Machine Repository erlaubt es, aus bestehenden oder für neue Asset-Anbindungen Templates zu generieren. Mit diesen Templates können Assets des gleichen Typs und der gleichen Nutzungsart einheitlich angebunden werden. Das Template enthält alle Konfigurationselemente, welche nicht asset-individuell sind. Individuelle Konfigurationselemente von Assets und Verbindungen sind beispielsweise IP-Adresse, Seriennummer, Equipment-Nummer etc. Durch das Verwenden eines bestehenden Templates wird der Zeitaufwand zur Anbindung eines Assets deutlich reduziert. Zusätzlich kann durch den Template Ansatz die Asset Konfiguration vereinheitlicht werden, und so eine bessere Vergleichbarkeit bei der Auswertung der Daten geschaffen werden.

3 Systemarchitektur

EDGE CONNECT ist architektonisch in Level (Schichten) unterteilt. Diese orientieren sich nach der betriebswirtschaftlichen Nutzung, was eine hohe Skalierbarkeit der einzelnen Komponenten ermöglicht. So können beispielsweise mehrere EDGE-Knoten gehostet werden, um die Assets logisch, aber auch Performance-orientiert aufzuteilen.



Level 0 - Assets

Auf der untersten Schicht erfolgt die Anbindung der Maschinen, Sensoren und Aktoren.

Level 1 - Connectivity

Die wachsende Auswahl an Plug-ins ermöglicht die Anbindung vielfältiger Steuerungen mit deren unterschiedlichen Kommunikationsstandards wie OPC UA oder MT Connect sowie herstellerspezifischen Protokollen.

Level 2 - Anwendung

Die Anzahl der möglichen EDGE-Knoten ist nicht begrenzt. Ein Knoten umfasst mehrere Schichten bzw. Aufgaben:

- **EDGE MDC Layer** steuert die Anbindung des Assets mittels **Plug-ins** und leitet die Signale an **EDGE Composition Layer** weiter. Analog dazu werden DNC-fähige Assets von **EDGE DNC Layer** angebunden.
- **EDGE Composition Layer** ist zuständig für die Signalinterpretation und das Komponieren der standardisierten Events.

- **EDGE API** ist die Programmierschnittstelle, über welche Assets, das Management von Events und die Benachrichtigung von Drittsystemen „northbound“ konfiguriert werden können.
- **Data Lake** hält alle Daten, Konfigurationsänderungen sowie Schreiboperationen und übertragene NC-Daten fest. Die Daten sind über die Data Lake API abrufbar.
- **DB** enthält alle Konfigurationen in Bezug auf die Asset-Anbindungen.

Level 3 - Werk

Die Konfigurationskomponente kann 1 bis n EDGE-Knoten bedienen. Es ist auch möglich, diese pro Werk, aber auch pro Fertigungslinie bereitzustellen.
Jede Komponente ist für sich eigenständig und ohne eine aktive Verbindung zu anderen Komponenten der EDGE CONNECT lauffähig (etwa durch zeitweisen Verlust). Dies ermöglicht unterschiedlichste Deployment-Modelle. So muss die EDGE-Konfiguration beispielsweise nicht zwingend in EDGE CONNECT selbst gehostet sein, sondern lediglich eine Verbindung zur jeweiligen API aufgebaut werden.
Grundlegend kommunizieren alle Komponenten über standardisierte Schnittstellen (HTTP/REST).

Level 4 - Enterprise

Das Machine Repository ist eine Extension von EDGE CONNECT, welches das Erstellen und Verwalten von Templates erlaubt.

4 Deployment

Bild 4: Möglichkeiten der Installation von EDGE CONNECT

Bei einer Installation wird ein Installer bereitgestellt, indem die **EDGE Configuration**, **EDGE Node und Machine Repository** enthalten sind. Diese werden vom Kunden selbst, oder von einem FORCAM Servicedienstleister installiert.
EDGE Configuration beinhaltet die gesamte Benutzeroberfläche inklusive aller Funktionen.
EDGE Node umfasst den EDGE-Knoten und kann beliebig oft installiert werden, die Anzahl der Knoten in EDGE CONNECT ist lediglich über die Lizenz begrenzt. Die Anzahl richtet sich nach dem erworbenen Subscription-Modell. Hier ist definiert, wie viele Knoten erstellt und wie viele Controller pro Knoten angeschlossen werden können.
Das Machine Repository enthält die Benutzeroberfläche mit samt ihren Funktionen. Es kann eine Vielzahl von EDGE-Instanzen versorgen. Eine EDGE-Instanz besteht aus einer EDGE Configuration mit den dazu angebundenen EDGE-Knoten.

ⓘ Das Machine Repository muss zusätzlich zu EDGE CONNECT erworben werden.

SAP BTP

EDGE CONNECT kann als Solution-Erweiterung von **SAP Digital Manufacturing Cloud** (SAP DMC) auf der SAP Business Technology Plattform (SAP BTP) erworben werden.
Bei einem Erwerb liefert FORCAM die Hardware, auf der EDGE CONNECT betrieben wird. Hierzu wird ein Microsoft Azure Stack EDGE (ASE) verwendet, was von FORCAM vorkonfiguriert wird. FORCAM benötigt hierfür feste IP-Adressen, die kundenseitig manuell eingetragen oder den Servicedienstleistern von FORCAM mitgeteilt werden müssen. Die Eingliederung des ASE in die bestehende Hardwareumgebung obliegt dem Kunden. FORCAM gewährleistet die Funktion und Verfügbarkeit des ASE und der Softwarekomponenten.

5 Grundlegende Einstellungen

Für allgemeine Einstellungen zu EDGE CONNECT steht der Bereich **Home** zur Verfügung.
Neben einer Einführung in die Tabellenhandhabung geht dieses Kapitel auf die folgenden Punkte ein:

- Benutzerverwaltung
- Versorgte Stammdaten
- Lizenzierung
- Download-Bereich
- Monitoring

ⓘ Die im Profil vorgenommenen Einstellungen bzgl. Sprache und Darkmode werden im Benutzerprofil gespeichert und gelten nur für diesen Benutzer.

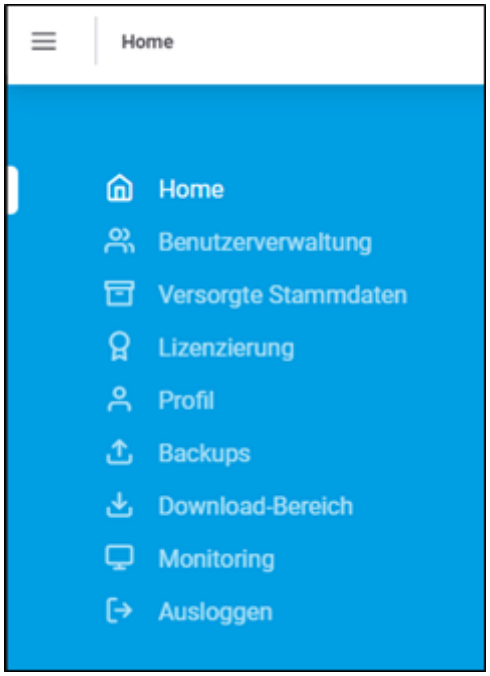


Bild 5: Aufruf des Home-Bereichs

5.1 Benutzerverwaltung

In der Benutzerverwaltung können Benutzer für EDGE CONNECT erstellt werden. Jedem Benutzer können Rechte zugewiesen werden, die nur die für ihn angemessenen oder beabsichtigten Funktionen enthalten (z. B. Maschine konfigurieren, Knoten neu starten etc.). Vorhandene Benutzeraccounts können auch nachträglich editiert werden.

ⓘ Wurden die Rechte eines eingeloggten Benutzers geändert, werden sie nach einem neuen Login sofort wirksam. Loggt sich der Benutzer nicht erneut ein, kann es bis zu 30 Minuten dauern, bis die Änderung aktiv ist.

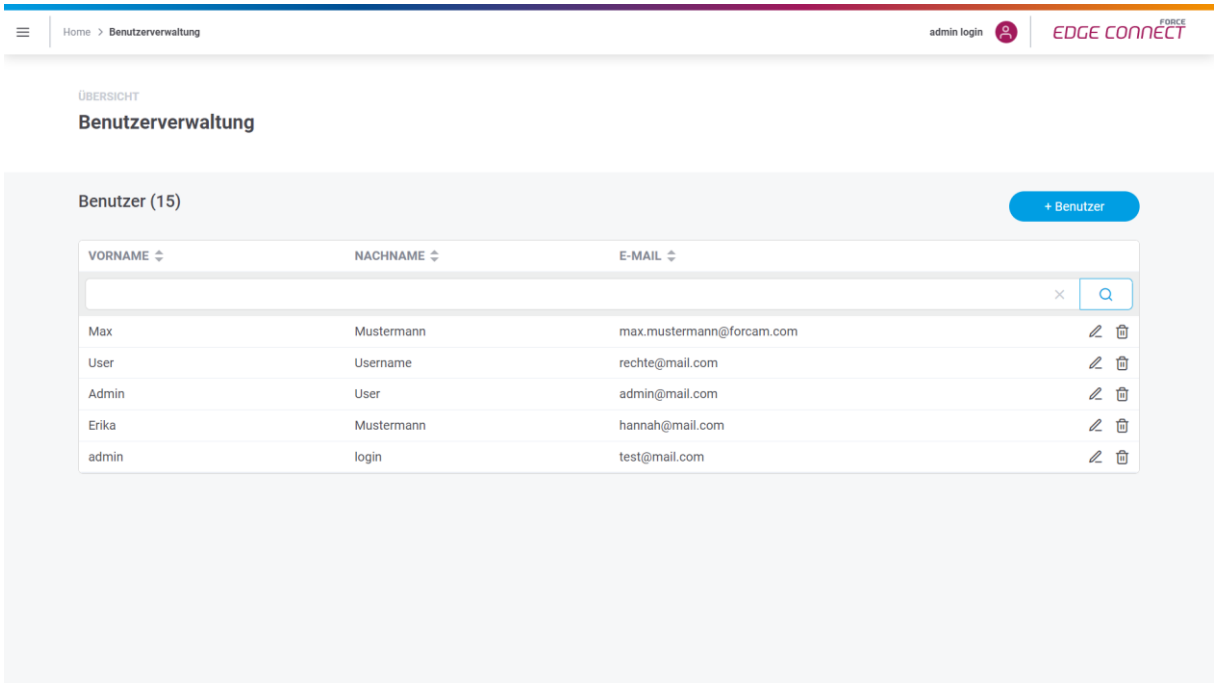


Bild 6: Benutzerverwaltung von EDGE CONNECT mit 2 Benutzern

Um einen neuen Benutzer zu erstellen:

1. Auf **+ Benutzer** klicken.
2. Im Folgedialog eine Mail-Adresse, Vor- und Nachnamen eintragen.
3. Gewünschtes Passwort setzen.
   Muss mind. 8 Zeichen lang sein, aus Groß- und Kleinbuchstaben bestehen und mind. eine Zahl und ein Sonderzeichen enthalten.
   Folgende Sonderzeichen sind erlaubt:
   ! „ # $ % & , ( ) * + , - . / : ; < = > ? @ [\ ] ^ _ ` { | } ~
4. Benutzerrechte vergeben (s.u.).
5. Speichern.

ⓘ Ein Benutzer kann mit denselben Daten kein weiteres Mal angelegt werden.

Tabelle 1: Benutzerrechte in EDGE CONNECT

Benutzerrecht	Beschreibung
Benutzerverwaltung	Der Benutzer kann die Benutzerverwaltung aufrufen, neue Benutzer erstellen und Rechte vergeben/entfernen.
Seite zur Überwachung	Der Benutzer kann in den Monitoring-Bereich einsehen von der EDGE-Konfiguration. Änderungen können nicht vorgenommen werden.
EDGE-Knoten neu starten	Der Benutzer kann einen EDGE-Knoten neu starten.
Versorgte Stammdaten	Der Benutzer kann Asset-Stammdaten aus einem Drittsystem an EDGE CONNECT übertragen
Knoten konfigurieren	Der Benutzer kann EDGE-Knoten bearbeiten, löschen und Änderungen in den Eventkonfigurationen vornehmen
Ohne Template konfigurieren	Der Benutzer kann Assets ohne MR-Templates anlegen.
Template-basierte Werte ändern	Der Benutzer kann die Daten ändern, die aus dem Template für das Asset übernommen wurden.
Mit Template konfigurieren	Der Benutzer kann Assets nur mit MR-Templates anlegen. Änderungen können nicht vorgenommen werden.

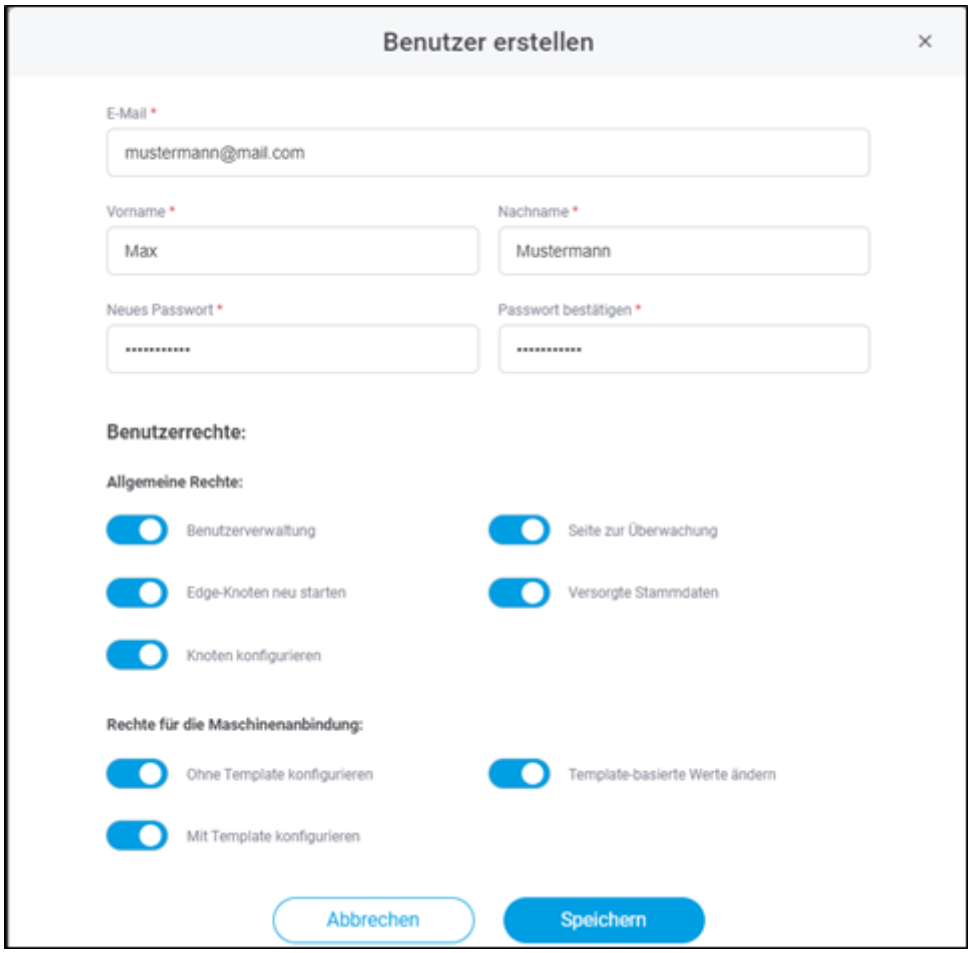


Bild 7: Dialog zu Erstellen eines neuen Benutzers

5.2 Versorgte Stammdaten

Die „Versorgte Stammdaten“ ist eine Erweiterung, die es ermöglicht, Asset-Stammdaten aus Drittanwendungen heraus an die EDGE CONNECT zu übertragen.

Stammdaten von Assets können über ein Drittanwendungen wie z. B. SAP DMC angelegt werden. Um diese in EDGE CONNECT nicht erneut anlegen zu müssen, können die Daten über die API-Schnittstelle an EDGE Configuration übertragen werden. Dadurch reduziert sich der Aufwand bei der Anlage eines Assets in EDGE CONNECT und die Durchgängigkeit bzw. Synchronität von Asset-Stammdaten werden unterstützt.

ⓘ Mit „Versorgte Stammdaten“ können Assets mit Basisinformationen angelegt werden. Anbindungsspezifische Informationen können durch ein Template oder manuell eingetragen werden.
Das Löschen von Assets über die API ist nicht möglich.

Werden die Stammdaten in der Drittanwendung verschickt, legt EDGE Configuration über die API ein Asset in EDGE CONNECT an. Alle neuen Assets werden in der Seite **Versorgte Stammdaten** angezeigt und erhalten initial den Status **Neue Stammdaten**. Sie können hier fertig konfiguriert und einem EDGE-Knoten zugewiesen werden.

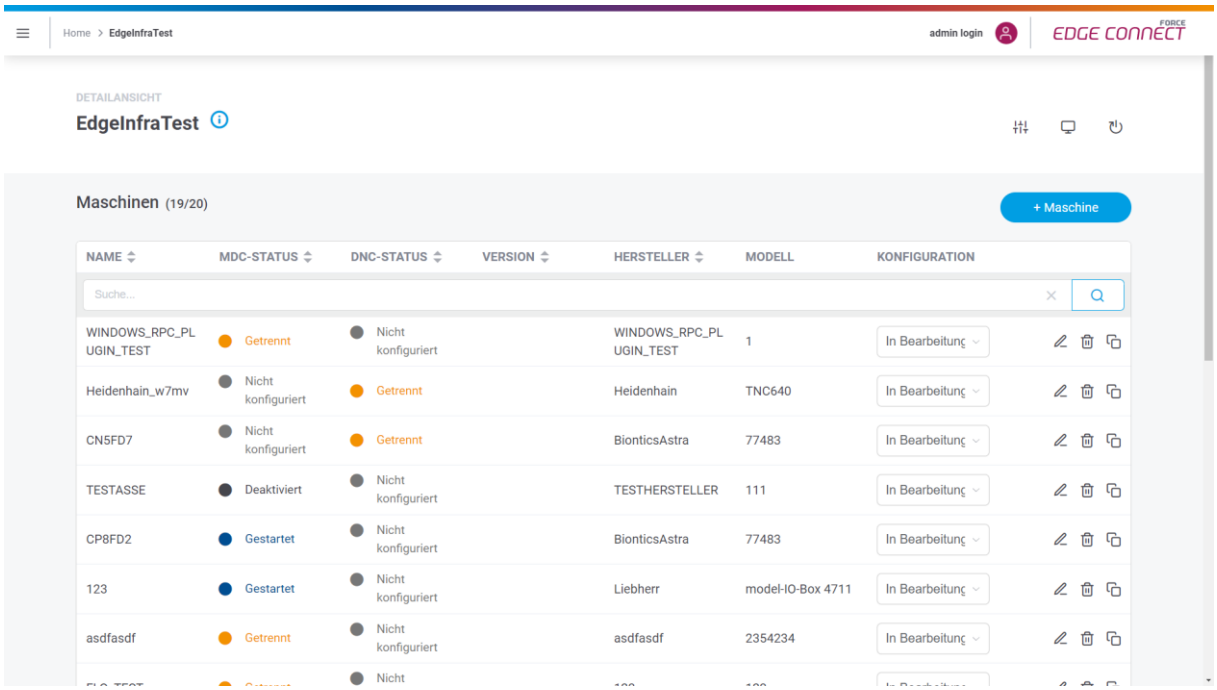


Bild 8: Liste von Assets mit versorgten Stammdaten

Folgende Stammdaten können über die API angelegt werden:

- Assetname
- Assettyp
- Assetklasse
- Hersteller
- Modell
- Seriennummer

Asset fertig konfigurieren

Die versorgten Stammdaten können auf einem bestehenden EDGE-Knoten angelegt werden. Dazu öffnet sich der Konfigurationsdialog zum Hinzufügen eines Asset (siehe Abschnitt 6.3). Die durch die API eingegangenen Stammdaten werden dabei höher priorisiert: Wird beim Konfigurieren eines über die API angelegten Assets ein Template ausgewählt, werden die über die API übergebenen Stammdaten verwendet und die des Templates verworfen.

Um neue Stammdaten auf einem Knoten anzulegen:

1. Bei den gewünschten Stammdaten rechts auf das Plus-Icon klicken.
2. Im Folgedialog einen EDGE-Knoten auswählen, auf dem die Stammdaten angelegt werden sollen.
3. Auf **Auswählen** klicken.
   - ➔ Der Konfigurationsdialog zum Hinzufügen eines Asset öffnet sich. Durch die API eingegangenen Stammdaten werden vorausgefüllt.
4. Weitere Konfigurationen wie gewünscht vornehmen (siehe Abschnitt 6.3).

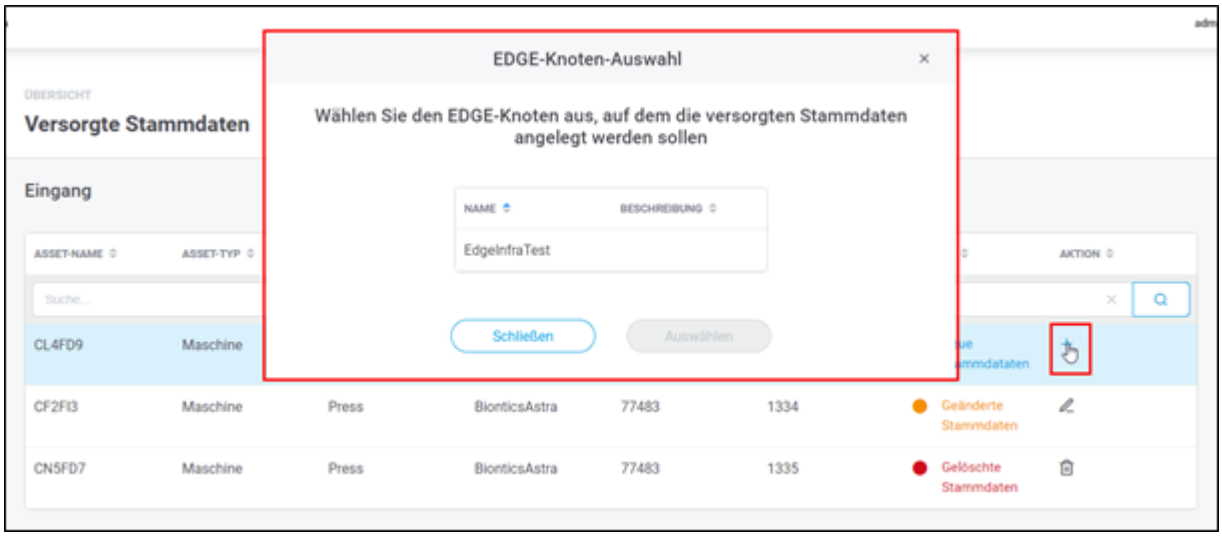


Bild 9: Auswahl eines EDGE-Knotens zum Anlegen von versorgten Stammdaten

Asset-Stammdaten über API ändern

Sämtliche Asset-Stammdaten können über die API geändert werden, nachdem sie bereits initial verschickt worden sind. Passt die Drittanwendung die Stammdaten an, ändert sich der Status in der Tabelle der versorgten Stammdaten auf **Geänderte Stammdaten**. Zudem wird der Benutzer von EDGE CONNECT auf der Startseite durch eine Meldung darüber informiert, dass Stammdaten geändert wurden.
Über das Stift-Icon kann entschieden werden, welche der Änderungen tatsächlich übernommen werden sollen.

Um Änderungen an Stammdaten zu übernehmen:

1. Bei den gewünschten Stammdaten rechts auf das Stift-Icon klicken.
   - ➔ Der Folgedialog listet alle Änderungen auf, die für die ausgewählten Stammdaten eingetroffen sind.
2. Den Schalter hinter den gewünschten Daten deaktivieren, deren Änderung *nicht* übernommen werden soll.
   Standardmäßig sind alle Schalter aktiviert.
3. Auf **Akzeptieren** klicken.

4. Resultat
5. Den entsprechenden Knoten neu starten.

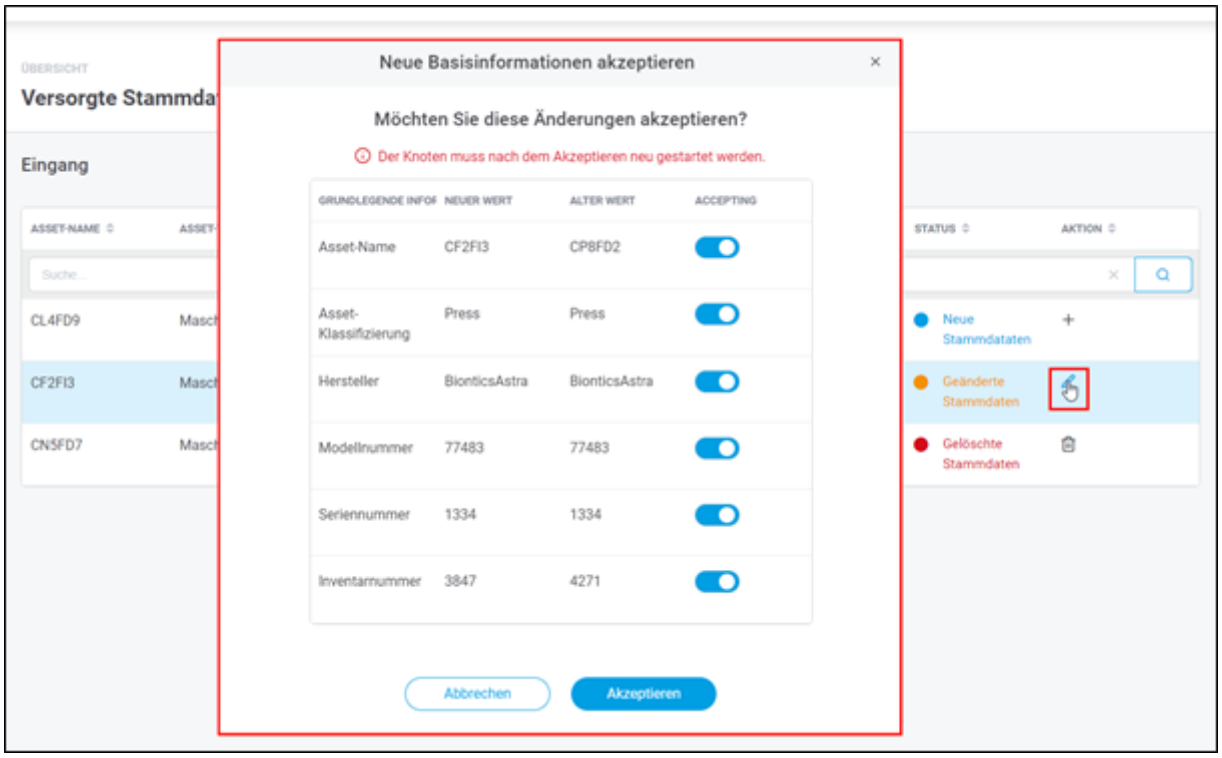


Bild 10: Bestätigung der Änderungen von Asset-Stammdaten

Asset über API für Löschung markieren

Wird ein Asset in der Drittanwendung gelöscht, erhält es unter der Seite **Versorgte Stammdaten** den Status **Zu löschen**. Wenn Sie die Löschung im EDGE-Knoten bestätigen, wird das Asset aus dem Knoten als auch aus der Tabelle der Seite entfernt.

5.3 Lizenzierung

Unter **Lizenzierung** können Lizenzen eingespielt und eingesehen werden.

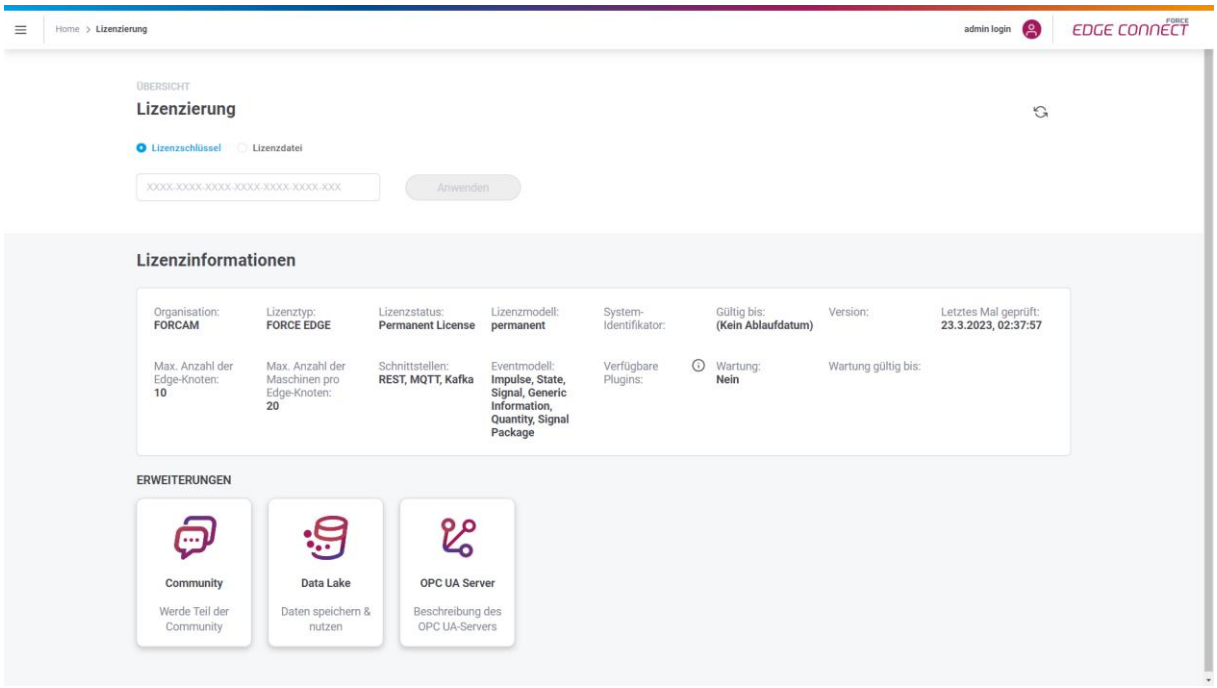


Bild 11: Lizenzierung und Übersicht

(1) Eine neue Lizenz kann als Datei hochgeladen oder direkt als Schlüssel eingetragen werden.
(2) Die Lizenzinformationen umfassen Typ und Status der Lizenz, Anzahl der lizenzierten Knoten und Assets, Wartung, Gültigkeit und weitere Daten.
(3) Alle gebuchten Add-ons werden hier aufgelistet.
Durch Klicken auf eine Add-on-Kachel erhalten Sie weitere Informationen wie bspw. bereitgestellt URLs.

5.4 Download-Bereich

Unter dem Reiter **Allgemein** kann die aktuelle Dokumentation von EDGE CONNECT in mehreren Sprachen heruntergeladen werden. Zur Verfügung stehen derzeit das Handbuch und eine Produktbeschreibung. Das Handbuch ist das vorliegende Dokument mit genauen Anweisungen zur Konfiguration. Die Produktbeschreibung ist ein kürzeres Dokument, das lediglich Funktion und Nutzen der Anwendung beschreibt und die Leistung aufführt und abgrenzt.

In den Reitern **MDC Plug-ins** und **DNC Plug-ins** stellt FORCAM zusätzliche Anwendungen bereit. Diese werden benötigt, um über das entsprechende Plug-in mit einem Asset zu kommunizieren. Die Anwendungen stehen dabei softwareseitig zwischen dem EDGE MDC Layer und dem Asset und ermöglicht die bidirektionale Kommunikation.

5.5 Monitoring

Das Monitoring im Home-Menü dient zur Überwachung der EDGE Configuration. Die Überwachung der EDGE-Komponenten wird auf einer separaten Seite angezeigt und ist in Kapitel 7 beschrieben. Der Aufbau der Monitoring-Kacheln ist jedoch der gleiche.

Die folgende Kachel überwacht den Status der Transmissionen von Templates über die API. Sie gibt alle Informationen an, die dabei geloggt werden.

Bild 12: Monitoring von Template-Transmissionen über die API

5.6 Sortierung von Tabelleneinträgen

Die meisten Seiten in EDGE CONNECT zeigen die Daten in Form von Tabellen an. Um die gewohnte Benutzerfreundlichkeit anzubieten, die Sie aus anderen Tabellentools kennen, wurde die Sortierfunktion von Spalten auch hier verwendet: Sie können die Spalten alphabetisch auf- oder absteigend sortieren.

Aufsteigend

Absteigend

Sortierung aus

Bild 13: Alphabetische Sortierung von Spalten

Spalten, die speziell DNC und MDC betreffen, geben statt einer Zeichenkette einen Status an. Die Sortierung ordnet die Status alphabetisch an und gruppiert sie zusätzlich inhaltlich.

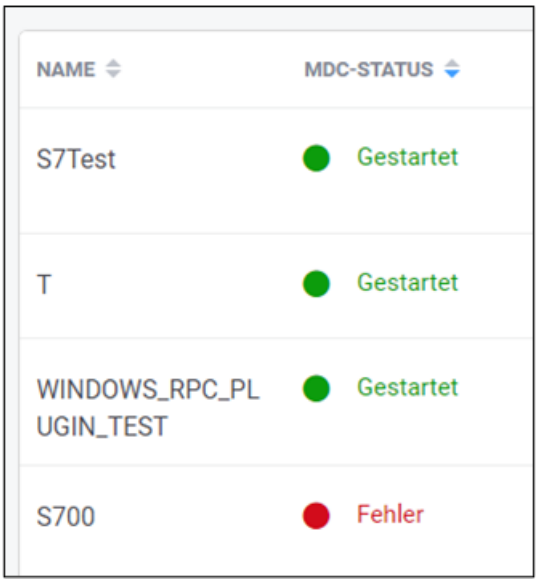


Bild 14: Alphabetische Sortierung und inhaltliche Gruppierung

6 EDGE-Konfiguration

Die Konfiguration eines EDGE-Knotens sowie eines Assets findet vollständig in der EDGE Configuration von EDGE CONNECT statt. Die benutzerfreundliche Oberfläche führt durch alle relevanten Einstellungen und zeigt in der Übersicht alle Knoten und die Status an.

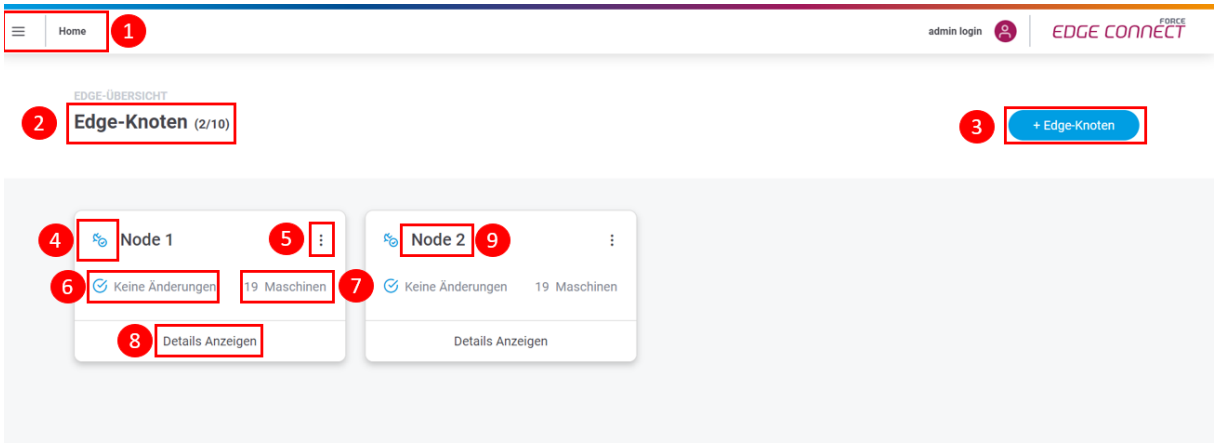


Bild 15: Einstiegs- und Übersichtsseite von EDGE CONNECT

(1) Home von EDGE CONNECT
- Benutzerverwaltung
- Versorgte Stammdaten
- Lizenzierung
- Profil
- Download-Bereich
- Monitoring

(2) Gibt an, wie viele EDGE-Knoten konfiguriert sind (erste Zahl) und wie viele Knoten gemäß Lizenz konfiguriert werden können (zweite Zahl)

(3) Fügt einen neuen EDGE-Knoten hinzu

(4) Status des EDGE-Knotens

(5) Einstellungsmenü des Knotens:
- Editieren
- Löschen

(6) Änderungsanzeige des EDGE-Knotens ggfs. Anzeige das der Note neu gestartet werden muss

(7) Anzahl der angebundenen Assets

(8) Weitere Detailinformationen des Knotens:
- Liste aller angebundenen Assets und deren Status
- Möglichkeit zum Hinzufügen eines neuen Assets
- Monitoring angebundener Assets

ⓘ Änderungen in der Benutzerverwaltung bzgl. Benutzerrechte können bis zu 30 Minuten benötigen, bis sie im gesamten System greifen.

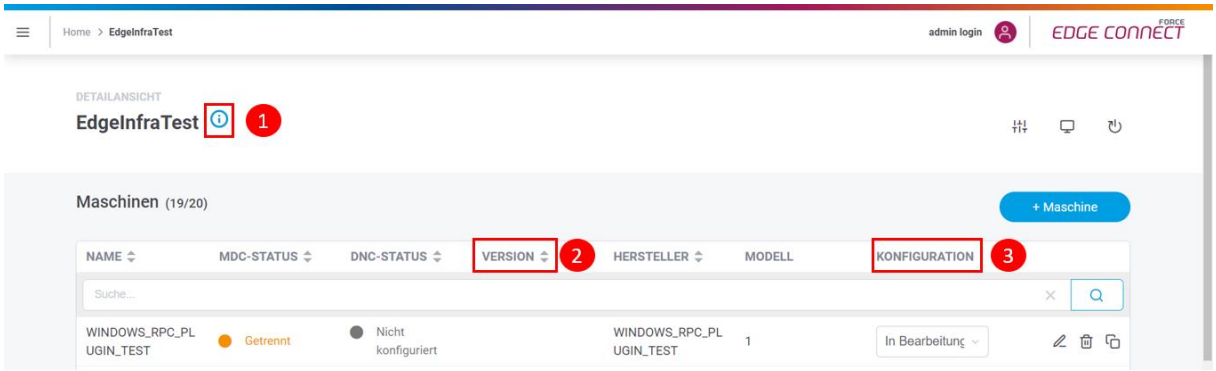


Bild 16: Asset-Übersicht als Folgeseite nach Klicken auf EDGE-Knoten

Über das **Icon** (1) lassen sich zusätzliche Informationen zum Knoten anzeigen.

Die **VERSION** (2) zeigt die aktuelle Ausführung des Templates an.

Unter **KONFIGURATION** (3) kann manuell bestimmt werden, welchen Status die Konfiguration haben soll, um etwa Mitarbeitenden einen Überblick zu verschaffen oder diese hinzuzuziehen:

- In Bearbeitung:
  Die Konfiguration ist noch nicht abgeschlossen und soll zu einer anderen Zeit fortgesetzt werden.
- In Validierung:
  Die Konfiguration des Assets soll auf Fehler und Konsistenz hin überprüft werden.
- Abgeschlossen:
  Die Konfiguration ist vollständig abgeschlossen. Nur in diesem Status kann der Lernzyklus des MR stattfinden und aus der Konfiguration ein Template generiert werden.

6.1 EDGE-Knoten hinzufügen

In EDGE CONNECT können Knoten in wenigen Schritten hinzugefügt werden. Ein EDGE-Knoten entspricht einer Instanz einer Anbindungsvariante. Pro Werk kann es mehrere Knoten geben. Sie werden logisch gebündelt, so dass die Last von Assets sinnvoll aufgeteilt wird.

ⓘ Wird ein konfigurierter EDGE-Knoten aus der Oberfläche entfernt, bleibt seine Konfiguration erhalten. Wird der Knoten wieder unter denselben Daten angelegt, übernimmt er die vorher konfigurierten Daten automatisch.

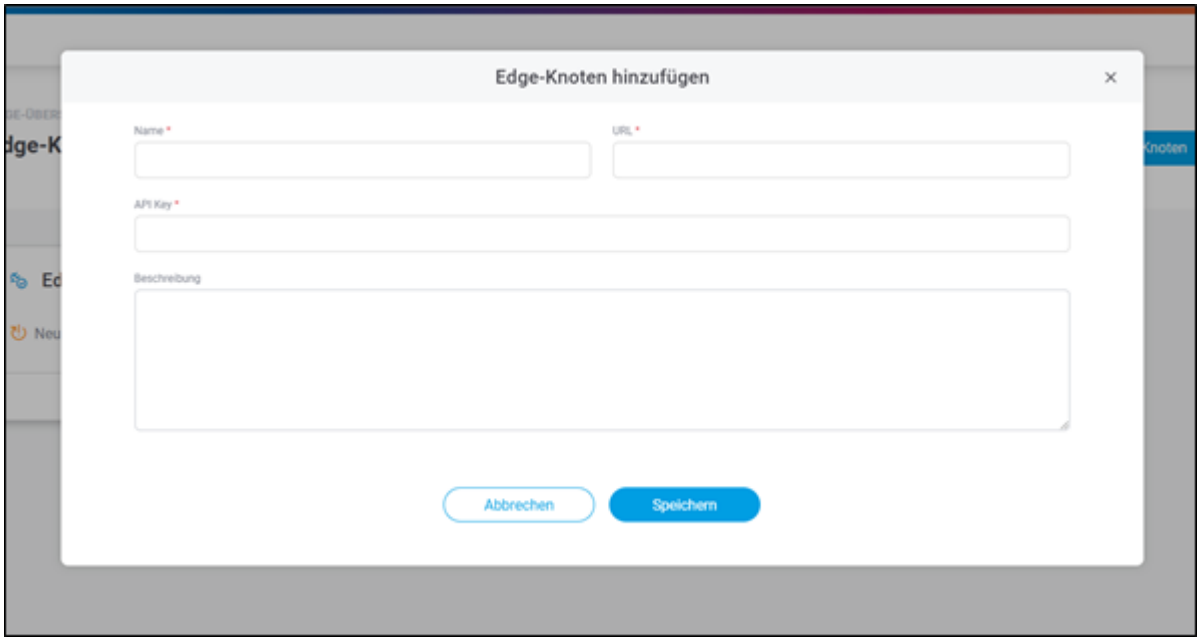


Bild 17: Dialog zum Hinzufügen eines neuen Knotens

Um einen neuen EDGE-Knoten hinzuzufügen:

1. In der Knotenübersicht (Home) auf **+ EDGE-Knoten** klicken.
2. Im Folgedialog alle obligatorischen Felder ausfüllen:
   - Name:
     Erscheint in der Knoten-Übersicht als Titel des Knotens
   - URL:
     Bestehend aus http + IP-Adresse+ Port 60067 (Bsp.: http://127.0.0.1:60067)
     Pro URL kann nur ein EDGE-Knoten erstellt werden.
   - API Key:
     Passwort, das bei der initialen Installation des Knotens vergeben wurde
3. Optional: Beschreibung hinzufügen.
4. Speichern.

6.2 Data Lake eines Knotens bearbeiten

Natürlich ist es dem Nutzer möglich, einen Knoten nach seiner Erstellung zu bearbeiten. Die Einstellungen beim nachträglichen Bearbeiten eines Knotens enthalten zusätzlich zu den bereits genannten Feldern auch Informationen und Einstellungen zum Data Lake.

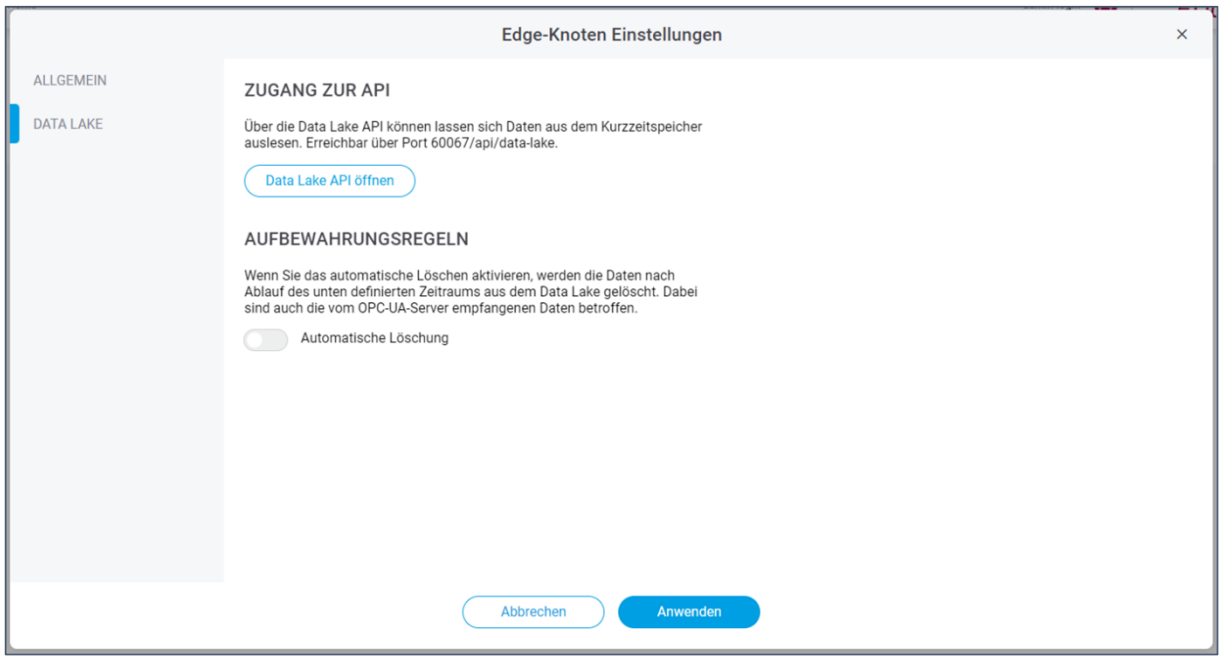


Bild 18: Dialog zum Bearbeiten eines bestehenden Knotens

Um einen bestehenden EDGE-Knoten zu bearbeiten:

1. Zur Knotenübersicht (Home) navigieren.
2. Auf die drei Punkte neben dem Namen des gewünschten Knoten klicken.
3. Auf die Option **Einstellungen** klicken.
4. Werte bearbeiten.
5. Speichern.

Während der Bearbeitung eines bestehenden Knotens ist es dem Nutzer möglich eine Retention Policy für den Data Lake zu bestimmen. Die Retention Policy beschreibt eine Regel, nach welcher die Daten innerhalb des Data Lakes behandelt werden sollen. Dabei handelt es sich ausschließlich um das Löschen von Asset-generierten Daten (Konfigurationsdaten o.ä. sind nicht betroffen). Hier kann ein zeitlicher Rahmen gegeben werden, wie lange die Asset-generierten Daten auf dem Data Lake gespeichert werden dürfen. Die Zeit ist dabei in Tagen anzugeben. Eine Retention Policy betrifft genau einen EDGE-Knoten und muss für andere separat erstellt werden, da sie standardmäßig nicht aktiviert ist. Sobald eine Retention Policy erstellt wurde, tritt diese sofort in Kraft und beginnt damit betroffene „alte“ Daten zu löschen. Dieser Prozess kann nicht abgebrochen werden, jedoch kann der zeitliche Rahmen der Policy im Nachhinein geändert werden. Dies ist nicht gleichzusetzen mit einem Prozessabbruch, da das System bereits damit begonnen hat, die initiale Policy auszuführen und damit gerechnet werden muss, dass Daten verloren gehen.

ⓘ Der Data Lake sollte als Kurzzeitspeicher verstanden werden, die technischen Anforderungen aus den System Requirements sind zu beachten!

6.3 Asset hinzufügen

Der Dialog zum Hinzufügen eines Assets leitet durch acht Schritte durch, die für eine Anbindung nötig sind. Hier werden unter anderem MDC-/DNC-Steuerungen konfiguriert und Asset-Signale definiert.

ⓘ Negative Werte sind in der Asset-Konfiguration nicht erlaubt.

ⓣ Ist ein Schritt abgeschlossen, wird er in der oberen Leiste blau markiert.
Durch Klicken auf einen abgeschlossenen Schritt kehrt man zu diesem zurück.
Beim Editieren eines bereits konfigurierten Assets kann jede Konfigurationsseite direkt ausgewählt und aufgerufen werden.

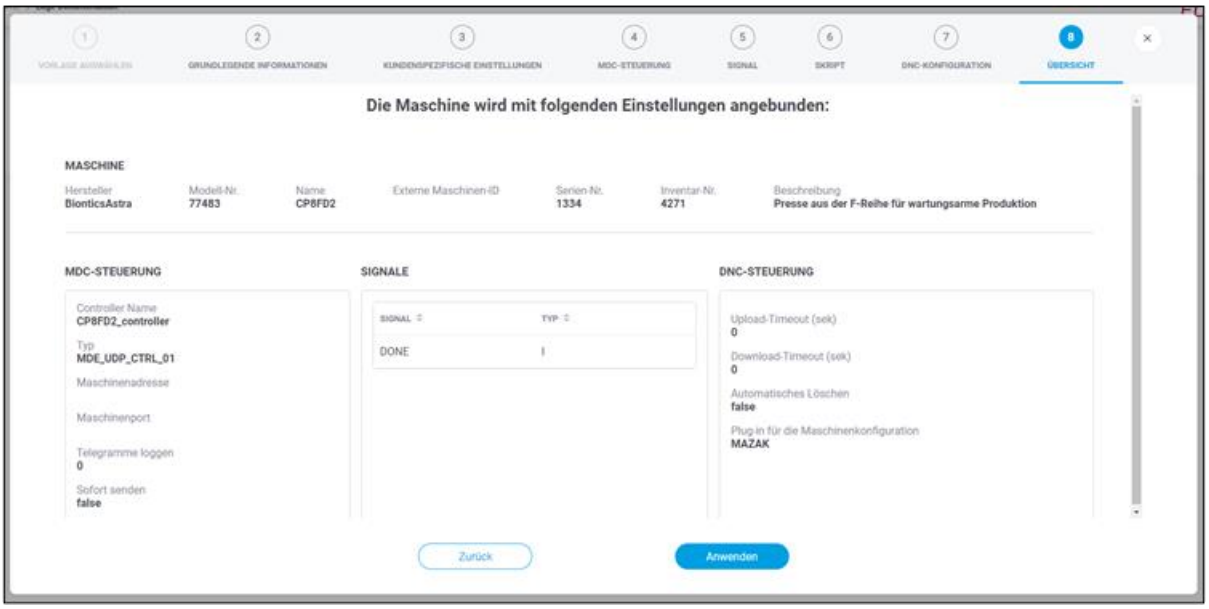


Bild 19: Dialog zur Konfiguration eines Assets in EDGE CONNECT

Um ein Asset hinzuzufügen:

1. In der Node-Detailansicht auf **+ Asset** klicken.
   ➔ Der Folgedialog führt durch die nachfolgenden acht Schritte zur Konfiguration eines Assets.

6.3.1 ① Vorlage auswählen

Mehrere Assets vom selben Typ müssen in EDGE CONNECT nicht jedes Mal komplett neu konfiguriert werden: Ein einmal konfiguriertes Asset kann als Template im Machine Repository erfasst werden und wird dann bei der nächsten Asset-Anbindung in dieser Maske angeboten. Wird das Template hier ausgewählt, werden alle Einstellungen automatisch für dieses Asset übernommen und alle nicht Asset-spezifischen Konfigurationsfelder sind vorausgefüllt. Lediglich Asset-spezifische (z. B. Seriennummer) und verbindungsspezifische Informationen (z. B. IP-Adresse oder Port des Assets bzw. Steuerung) müssen noch angepasst werden.

Die **VERSION** gibt an, in welcher Ausführung es sich befindet. Wird ein Template überarbeitet, wird die Versionsnummer automatisch hochgezählt um 1 und die frühere Version überschrieben. Version 0 bedeutet, dass im entsprechenden Template kein Skript konfiguriert ist.

ⓘ Dieser Schritt ist nur verfügbar, wenn die MR-Extension verwendet wird. Ist kein Template konfiguriert oder MR nicht in Verwendung, beginnt die Asset-Anbindung mit Schritt ②.

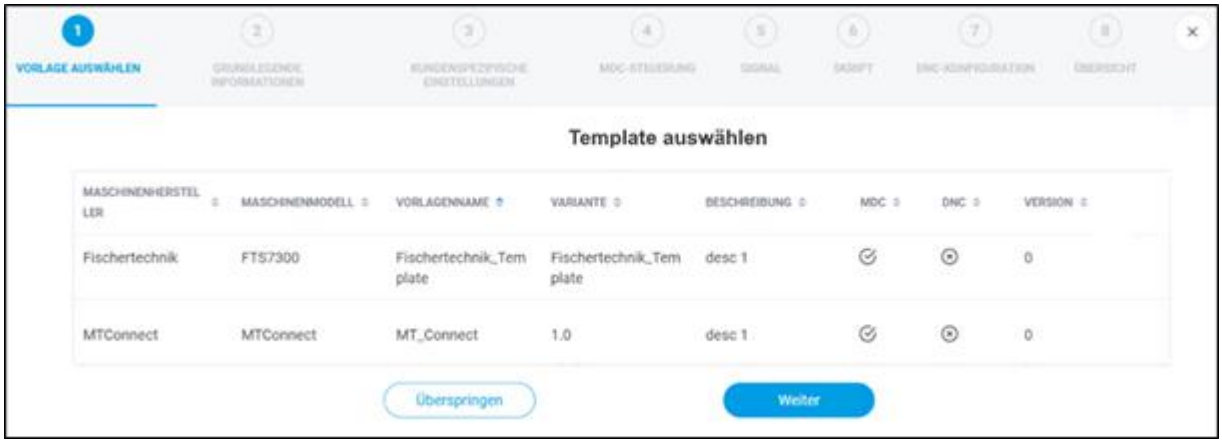


Bild 20: Asset hinzufügen - Vorlage auswählen

1. Gewünschtes Template zur Asset-Anbindung in der Liste auswählen.
2. Auf **Weiter** klicken.

ⓘ Wenn Sie kein Template verwenden zur Asset-Anbindung, auf **Überspringen** klicken.

6.3.2 ② Grundlegende Informationen

Basisinformationen vom konfigurierenden Asset wie z. B. Name oder Seriennummer. Hier wird zudem bestimmt, ob eine MDC- oder eine DNC-Steuerung konfiguriert werden soll – oder beides. Mit dem MDC-Controller werden Signale von dem Asset abgeholt und weitergegeben oder auf diese geschrieben. Über die DNC-Steuerung werden NC-Dateien an das Asset übertragen.

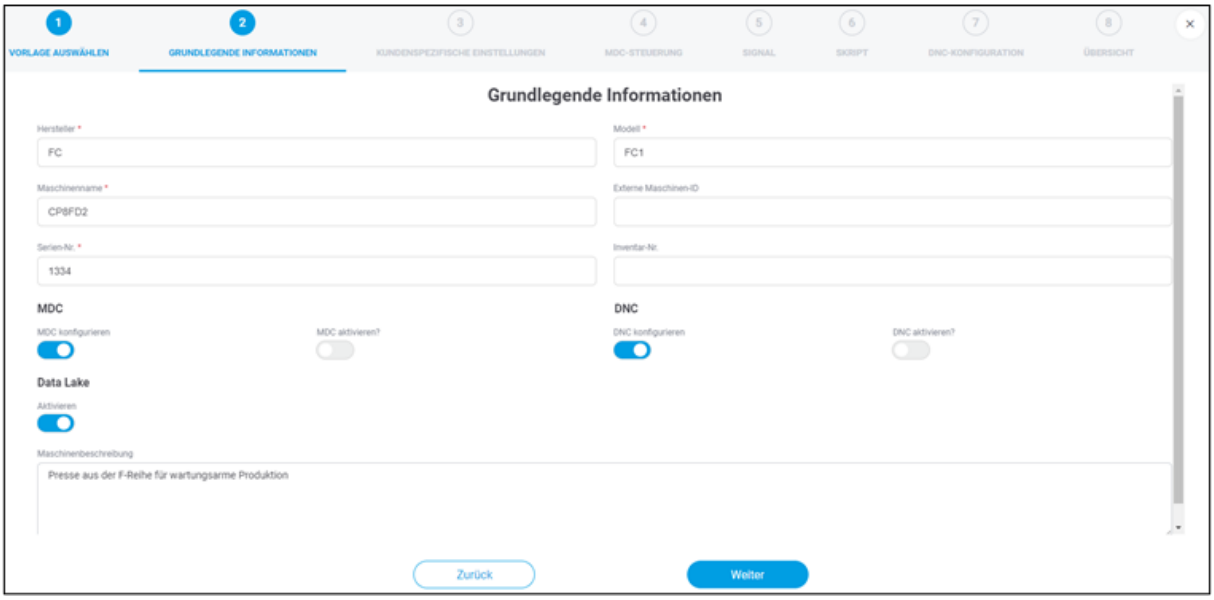


Bild 21: Asset hinzufügen - Grundlegende Informationen

1. **Asset-Typ** und **Asset-Klassifizierung** wählen.
2. **Hersteller, Modell, Asset-Name**, und **Serien-Nr**. eintragen.
3. Optional: Weitere Informationen wie gewünscht eintragen.
4. Schalter **MDC konfigurieren** und/oder **DNC konfigurieren** aktivieren.
5. Optional: Bestimmen, ob MDC oder DNC initial aktiviert werden soll.
6. Optional: Data Lake aktivieren.
7. Auf **Weiter** klicken.

6.3.3 ③ Kundenspezifische Einstellungen

Möglichkeit zum Hinterlegen individueller, werkspezifischer Informationen eines Assets mit dem Zweck, Asset-Daten zusätzlich anzureichern. Diese Daten können später von der API abgerufen werden, um einem Drittsystem mehr Informationen bereitzustellen.

Beispiel: Name = Ort, Wert = Halle 2.
Hier wird den Asset-Daten ein zusätzlicher örtlicher Aspekt verliehen, um das Asset bei einer eventuellen Störung genau zu lokalisieren.

ⓘ Dieser Schritt ist optional.

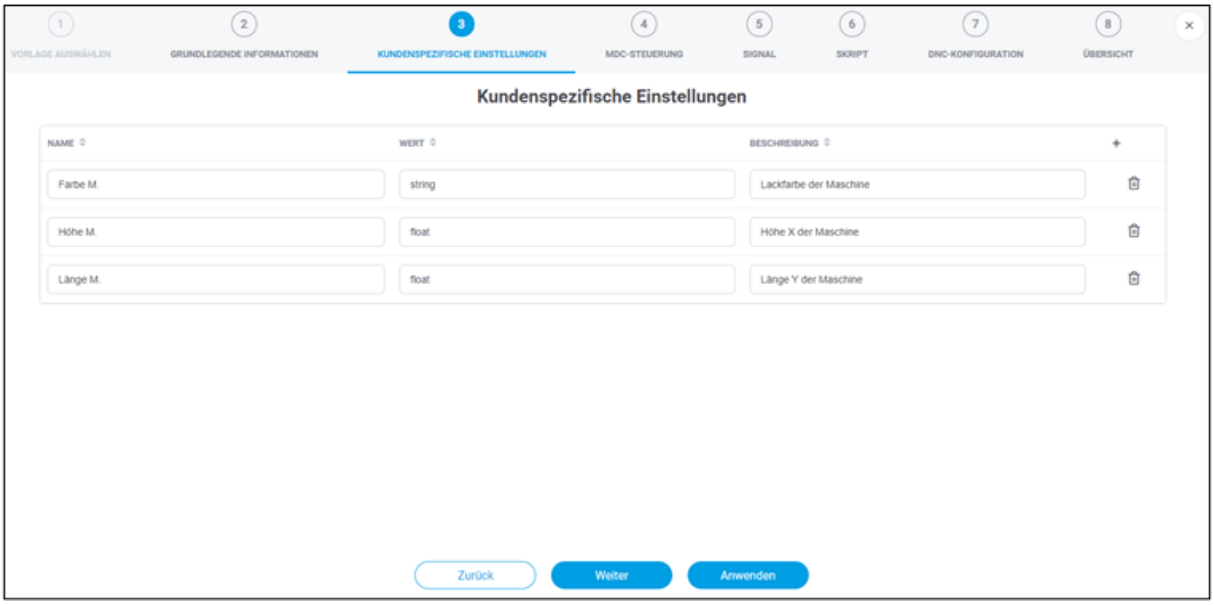


Bild 22: Asset hinzufügen - Kundenspezifische Einstellungen

1. Auf das **+** Icon klicken.
2. Gewünschte Parameter eintragen.
3. Auf **Weiter** klicken.

6.3.4 ④ MDC-Steuerung

Möglichkeit zur Konfiguration einer MDC-Steuerung. Bestimmt die Art und Weise, wie mit dem Asset verbunden werden soll.
FORCAM unterstützt alle gängigen Steuerungen auf dem Markt und ist bemüht, die Verfügbarkeit stetig auszubauen. Eine Übersicht der aktuellen FORCAM Plug-ins ist in Abschnitt 8.3 aufgelistet.

Der Bus-Typ ist ein bestimmtes Kommunikationsprotokoll des Steuerungstyps. Bei vielen Steuerungen gibt es lediglich ein Protokoll und somit nur einen Bus-Typ zur Auswahl (z. B. Bus-Typ **FORCAM IO Box connection** für den Controller **FORCAM IO Box**). Bei der Steuerung Siemens S7 sind beispielsweise mehrere Protokolle möglich, weswegen mehrere Bus-Typen verfügbar sind.

ⓘ Dieser Schritt ist nur verfügbar, wenn in Schritt ② **MDC konfigurieren** aktiviert wurde.

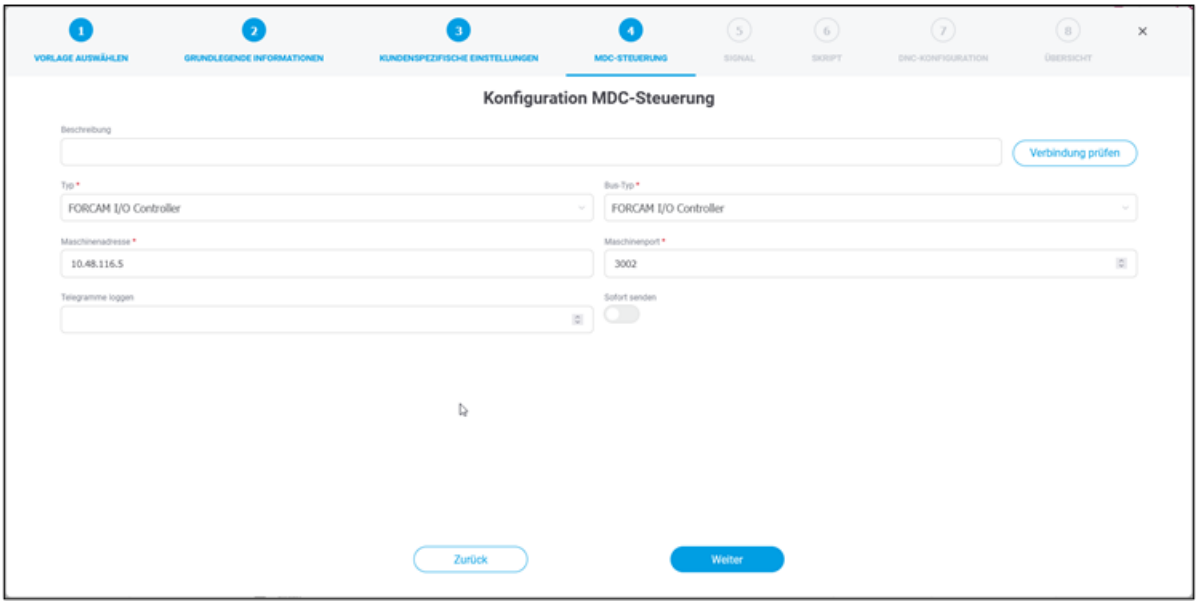


Bild 23: Asset hinzufügen - MDC-Steuerung

1. Optional: Beschreibung der Steuerung eintragen.
2. **Steuerungstyp** auswählen.
   ➔ Je nach ausgewähltem Typ erscheinen zusätzliche Konfigurationsparameter.
3. **Bus-Typ** auswählen.
   Die Auswahlmöglichkeit richtet sich nach dem zuvor ausgewählten Steuerungstyp.
4. Steuerungsspezifische Konfiguration anpassen.
5. Optional: Prüfen der Verbindung zur Maschine
6. Auf **Weiter** klicken.

ⓘ Die Funktion **Telegramme loggen** dient zum Mitloggen der UDP-Telegramme im DCU-Log. In dem Eingabefeld wird die Anzahl der zu loggenden Zeichen jedes Telegramms angegeben.

6.3.5 ⑤ Signal

In diesem Schritt wird festgelegt, welche Signale aus der Steuerung ausgelesen werden. Je nach Konfiguration der MDC-Steuerung (Schritt ④) werden unterschiedliche Auflistungen der Signaltypen angezeigt. Mit dem Data Lake können alle Daten festgehalten und gespeichert werden. Pro Signal kann die Data Lake Speicherung an- und abgeschaltet werden. Es können Einheiten auf einzelne Signale erfasst werden (z. B. Grad Celsius oder Liter die Minute), und zudem können Skalier-Faktoren festgelegt werden.

ⓘ Falls der **Aktiv** Schalter für das Signal deaktiviert ist, so kann es in Schritt ⑥ COMPOSITION nicht verwendet werden.

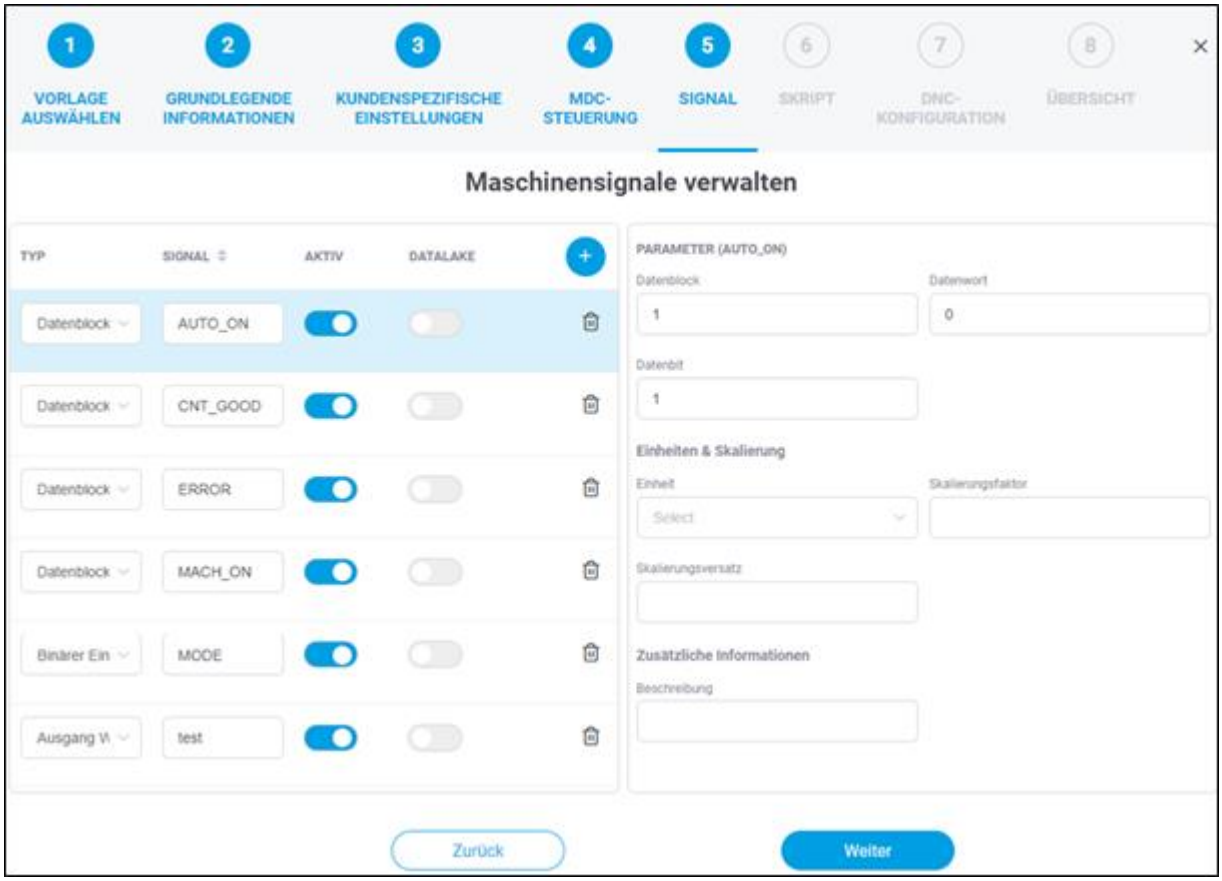


Bild 24: Asset hinzufügen – Signale

1. Auf das **+** Icon klicken.
2. **Typ auswählen**, **Signalnamen** eintragen sowie optional den Schalter für **Data Lake** aktivieren.
3. Plug-in spezifische Signal**-Parameter angeben.**
4. Optional**: Einheiten & Skalierung und Beschreibung** angeben.
5. Auf **Weiter** klicken.

6.3.6 ⑥ Composition

In diesem Schritt werden die erfassten Signale interpretiert, woraus logische Schlussfolgerungen bezüglich des Verhaltens des Assets gewonnen werden können. Daraus resultieren beispielsweise Messwerte, Informationen für die Wartung und unterschiedliche Produktionsstatus. Hier in Schritt ⑥ **Composition** werden Bedingungen für die Interpretation von Signalen definiert. Bedingungen können auf zwei Arten erfasst werden: Zum einen wird unter **SCRIPT** der Text-basierte Code angezeigt und bearbeitet (siehe Bild 26), zum anderen unter **GRAPHIC** die grafischen Blöcke (siehe Bild 25). Das sind Programmier-Bausteine, die ähnlich wie Puzzleteile zusammengesetzt werden können. Vorteil des Baukastensystems ist, dass auch Einsteiger ohne Programmiererfahrung Befehle erstellen können. Im linken Bereich der Maske werden alle Funktionskategorien farblich sortiert aufgelistet. Per Drag-and-drop können die gewünschten Blöcke nach rechts in den Arbeitsbereich gezogen und in die richtige Reihenfolge gebracht werden. Hier wird die tatsächliche Asset-Logik definiert.

⚠ Beim Einsatz einer Liste, nicht vergessen diese zu leeren.

⚠ Ein Wechsel von SCRIPT zu GRAPHIC ist nicht möglich. Wurde die Grafik zu einem Skript konvertiert, kann man diese dort bearbeiten aber nicht zurück in die Blockform setzten.

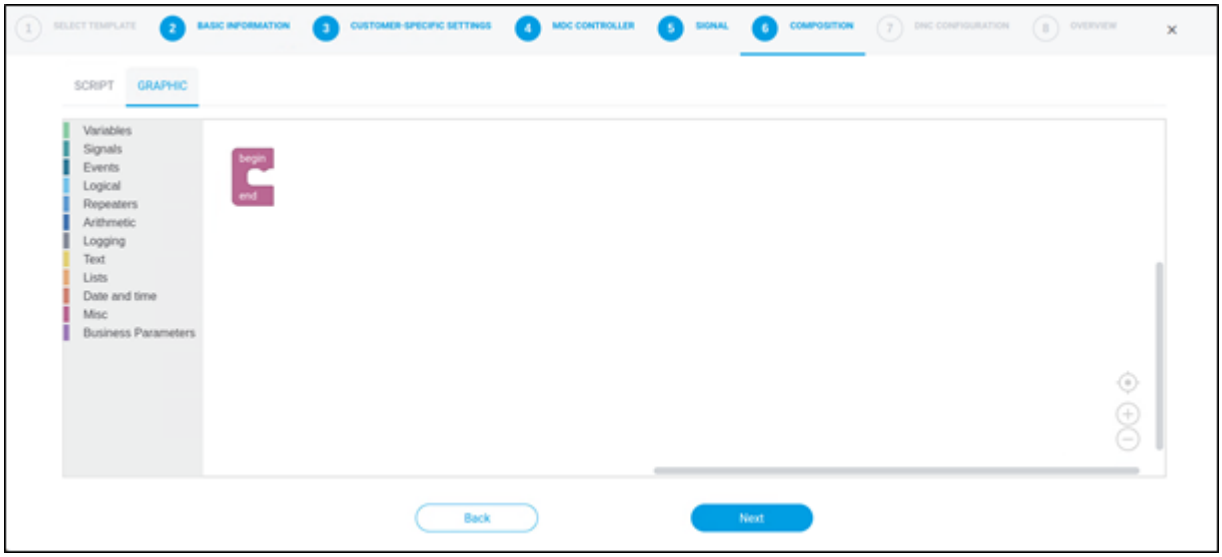


Bild 25: Asset hinzufügen – GRAPHIC

ⓘ Im Anhang finden Sie Beispiele zum Skript sowie ihre Funktionen (siehe Abschnitt 8.5 bzw. 8.6).

ⓘ Für eine detaillierte Erklärung der Funktionskategorien der Blöcke, siehe **Handbuch - Grafische Komposition**.

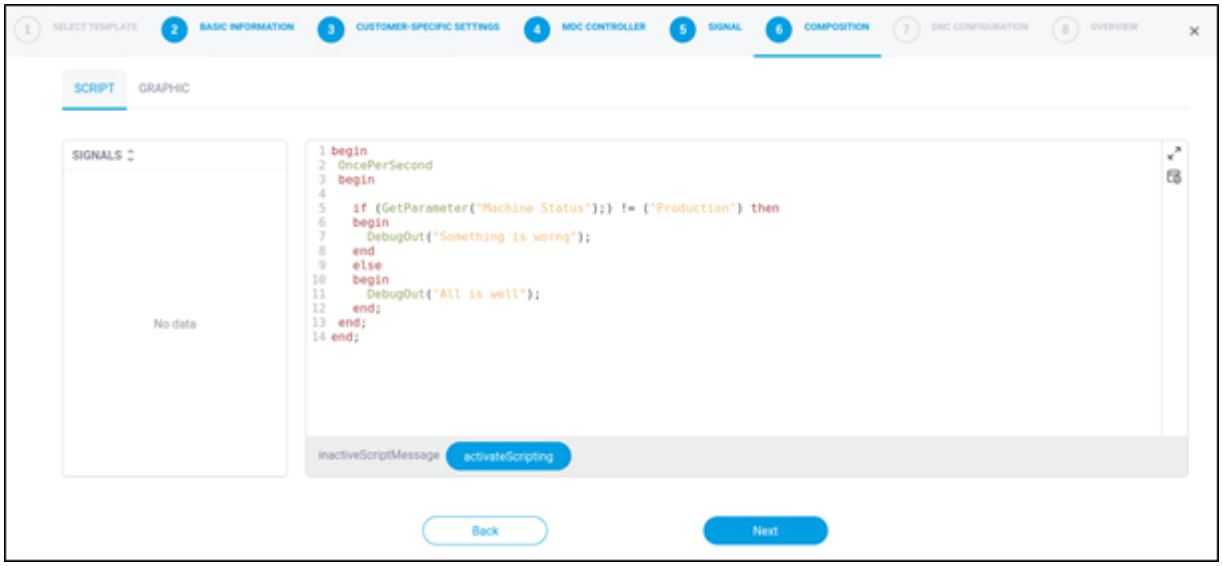


Bild 26: Asset hinzufügen - SCRIPT

1. Gewünschtes Skript in das zentrale Eingabefeld eintragen.
2. Optional: Skript unter dem Reiter **SCRIPT** am rechten oberen Rand auf Validität zu prüfen.
3. Auf **Weiter** klicken.

⚠ Ein fehlerhaftes Skript ist nicht möglich. Der nächste Konfigurationsschritt ist nur mit fehlerfreiem Skript erreichbar.

ⓘ Der Skript Editor kann über das Maximieren Icon auf Vollbild gestellt werden.

6.3.7 ⑦ DNC-Konfiguration

Möglichkeit zur Konfiguration einer DNC-Steuerung. Bestimmt die Art und Weise, wie eine NC-Datei an das Asset übertragen werden soll.
FORCAM unterstützt alle gängigen Steuerungen auf dem Markt und ist bemüht, die Verfügbarkeit stetig auszubauen. Eine Übersicht der aktuellen FORCAM Plug-ins ist in Abschnitt 8.3 aufgelistet.

ⓘ Dieser Schritt ist nur verfügbar, wenn in Schritt ② **DNC konfigurieren** aktiviert wurde.

ⓘ Nach der DNC-Konfiguration kann die Verbindung überprüft werden

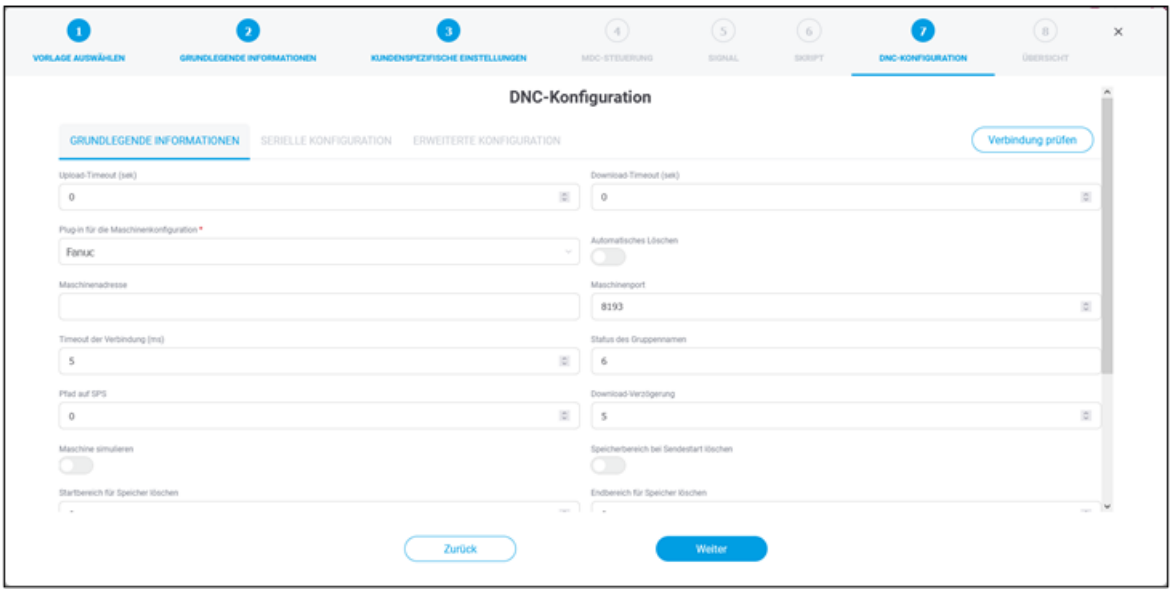


Bild 27: Asset hinzufügen - DNC-Konfiguration

1. Optional: **Upload**- und **Download-Timeout** eintragen.
2. **Plug-in für die Asset-Konfiguration** auswählen.
   ➔ Je nach ausgewähltem Plug-in erscheinen zusätzliche Konfigurationsparameter auf der Seite.
3. Optional: Automatisches Löschen bestimmen.
   Ist der Schalter aktiviert, wird die NC-Datei nach dem Lesen aus dem Asset automatisch gelöscht.
4. Weitere Konfigurationsparameter abhängig vom ausgewählten Plug-in in den übrigen Reitern eintragen.
5. Prüfen der Verbindung zum Asset.
6. Auf **Weiter** klicken.

6.3.8 ⑧ Übersicht

Fasst die bisherige Konfiguration aus allen Schritten zusammen und listet alle definierten Signale auf. Nach der Bestätigung wird das Asset mit der angegebenen Konfiguration abgebildet und somit digitalisiert. Das konfigurierte Asset erscheint unter dem definierten Namen in der Übersicht (siehe Bild 16).

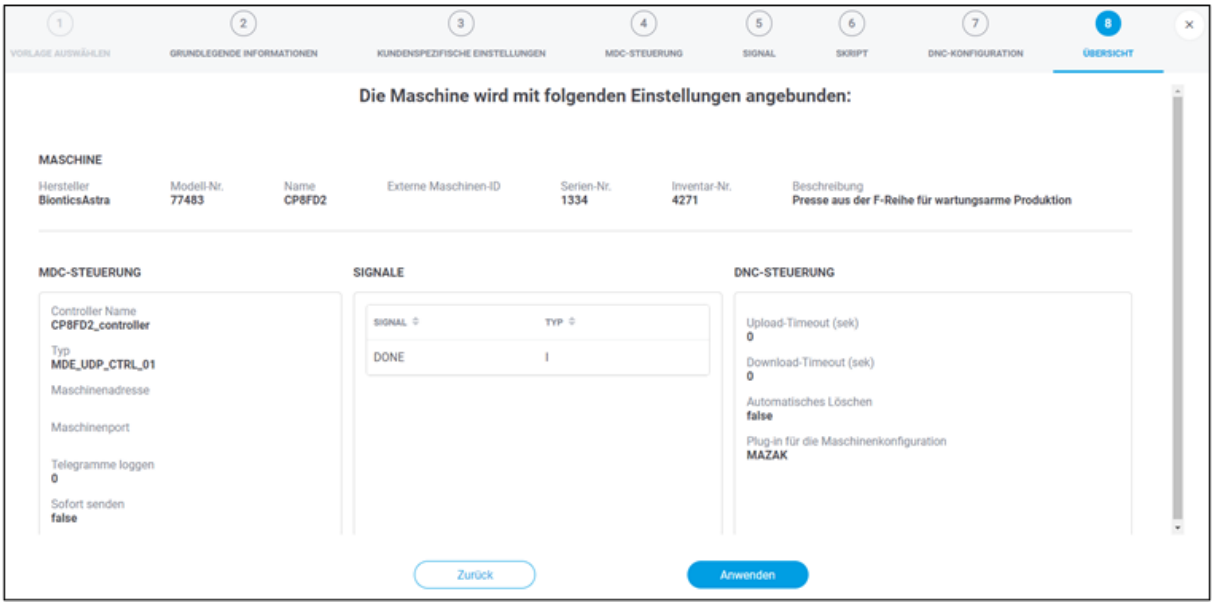


Bild 28: Asset hinzufügen - Übersicht

6.4 Northbound Configuration

In der Northbound-Configuration wird festgelegt, wie die Signale an ein übergeordnetes System geschickt werden. Payload und Endpunkt sind standardmäßig vordefiniert, können aber individualisiert werden.

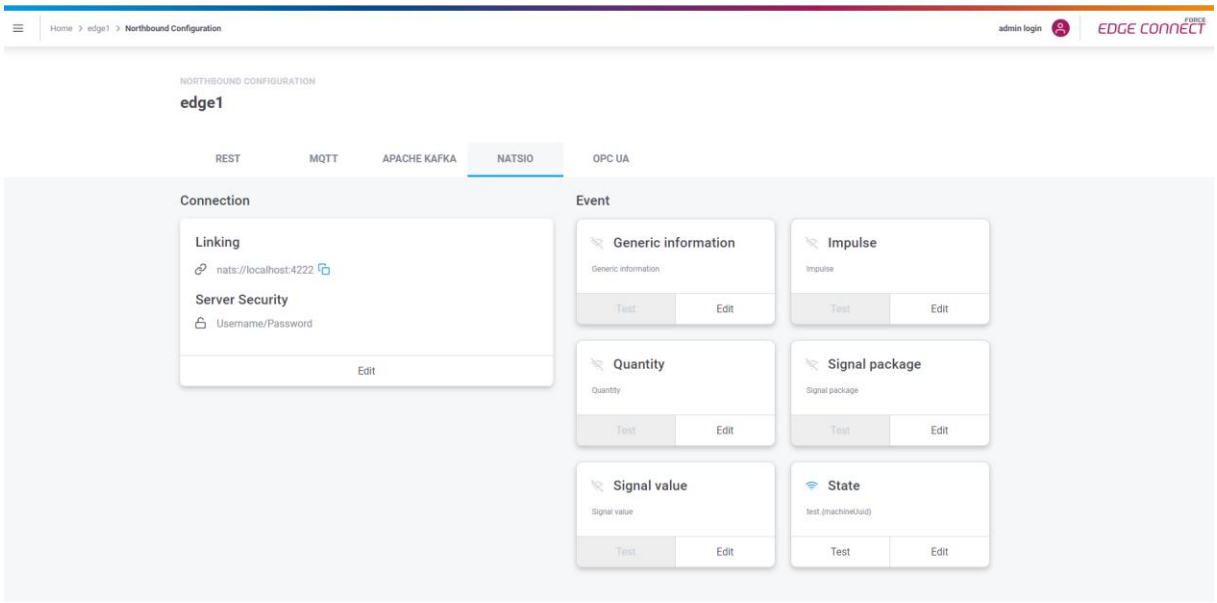


Bild 29: Northbound Configuration in EDGE CONNECT

Events werden in einem Skript verwendet, um daraus ausgehende Ereignisse zu generieren. Hierfür stehen Skriptfunktionen zur Verfügung, die je nach Typ ein entsprechendes Event erzeugen.
Für jeden Event-Typ gibt es ein standardisiertes **Event**. Der Event-Typ **Menge** verschickt beispielsweise die von dem Asset produzierte Menge. Alle verfügbaren Events sind in Kapitel 8.4 aufgelistet.

Im JSON-body unter **PAYLOAD** wird festgelegt, wie die Nachricht an das übergeordnete System aussehen soll. Die Platzhalter (Wildcards) werden schließlich durch die entsprechenden, vorhandenen Signale ersetzt. Beispiel einer Event-Struktur:

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

Skriptfunktionen ermöglichen es Events, **Platzhalter** (Wildcards) zu verwenden, mit deren Hilfe unterschiedliche Informationen weitergegeben werden können. Hierüber kann beispielsweise die Maschinen-ID oder der Zeitstempel in UTC aufgelöst werden. Kapitel 8.6 listet alle verfügbaren Skriptfunktionen auf und erklärt diese.
Ist der Schalter unter **AKTIV** aktiviert, wird das entsprechende Event verschickt. Nicht aktivierte Events werden nicht verschickt.
Ein aktives Event kann durch Klicken auf **TEST** auch verprobt werden. Im Folgedialog können Werte wie **machineId** (Maschinen-ID) oder **value** (Wert) eingetragen werden, um das Signal exemplarisch zu generieren und auszuführen, ohne einen Einfluss auf die tatsächliche Asset-Anbindung zu haben. Auf diese Weise können Events vorab getestet werden, ohne sie in der Live-Umgebung ausführen zu müssen.

6.4.1 Signale & Events von EDGE zum übergeordneten System

Für die Versorgung von Signalen und Events von einem EDGE-Knoten zu einer 3rd-Party-Applikation gibt es fünf technische Möglichkeiten.

ⓘ Die Versorgung kann jeweils im EDGE-Knoten selbst konfiguriert werden.

HTTP/REST

Zur Versorgung des Fremdsystems kann ein beliebiger dort bereitgestellter REST-Endpunkt bedient werden. Dabei werden die HTTP Methoden POST und PUT unterstützt.
Als HTTP-Authentifizierungsmethoden sind folgende Standards implementiert:

- Basic Authentification: Authentifizierung nach RFC 2617 durch Eingabe von Benutzername und Passwort (siehe https://datatracker.ietf.org/doc/html/rfc2617).
- Client credential flow: Authentifizierung nach OAuth 2.0 RFC 6749 über dem System bekannte Client ID und Client Secret. (siehe https://auth0.com/docs/flows/client-credentials-flow). Diese Art der Authentifizierung erfolgt ohne Benutzereingriff, d.h. im Hintergrund.

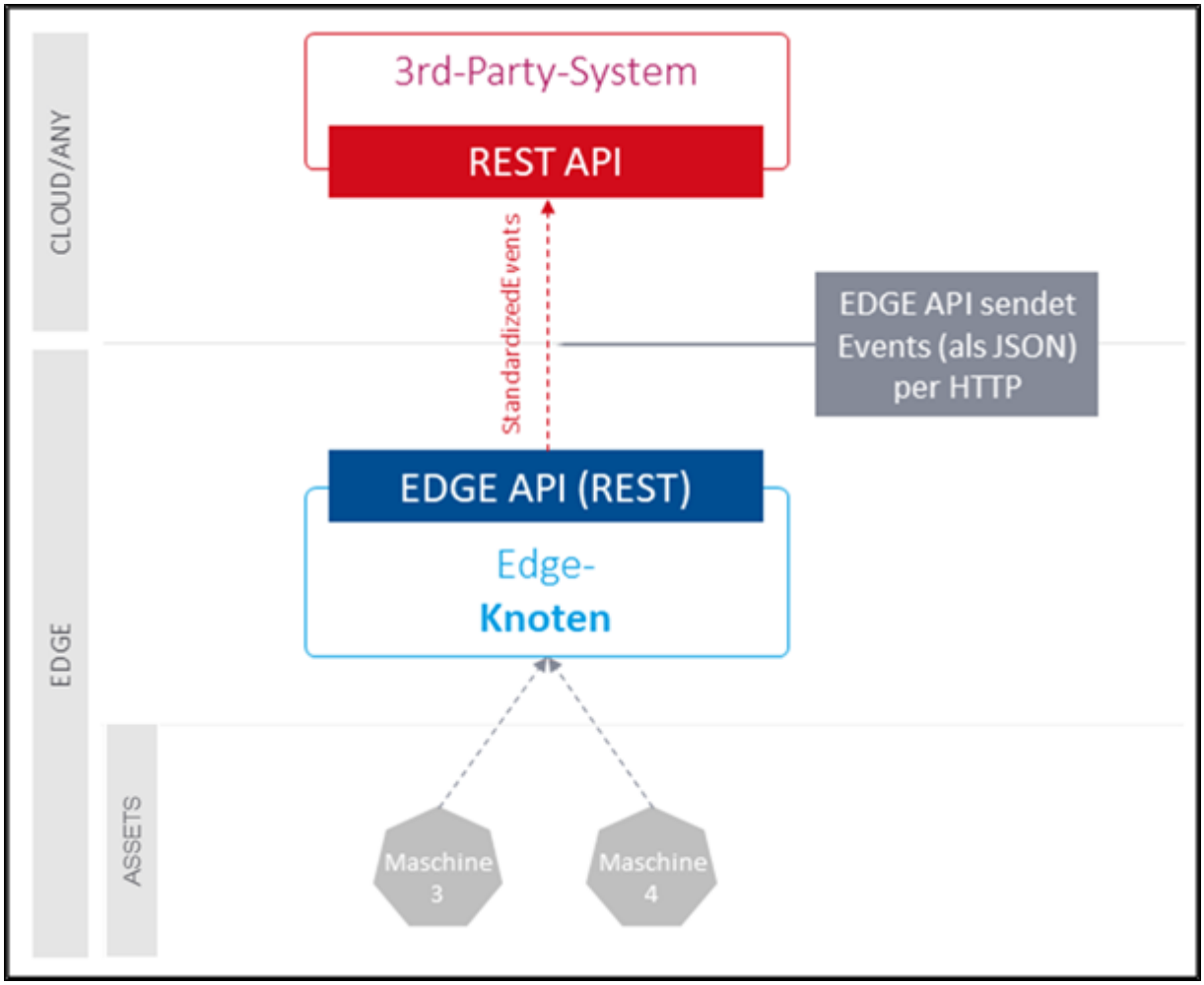


Bild 30: Kommunikation mit übergeordneten Systemen über HTTP/REST

MQTT-Messaging

Zur Versorgung kann auch ein beliebiger MQTT-Broker bedient werden, sofern durch Kunden oder Partner bereitgestellt.

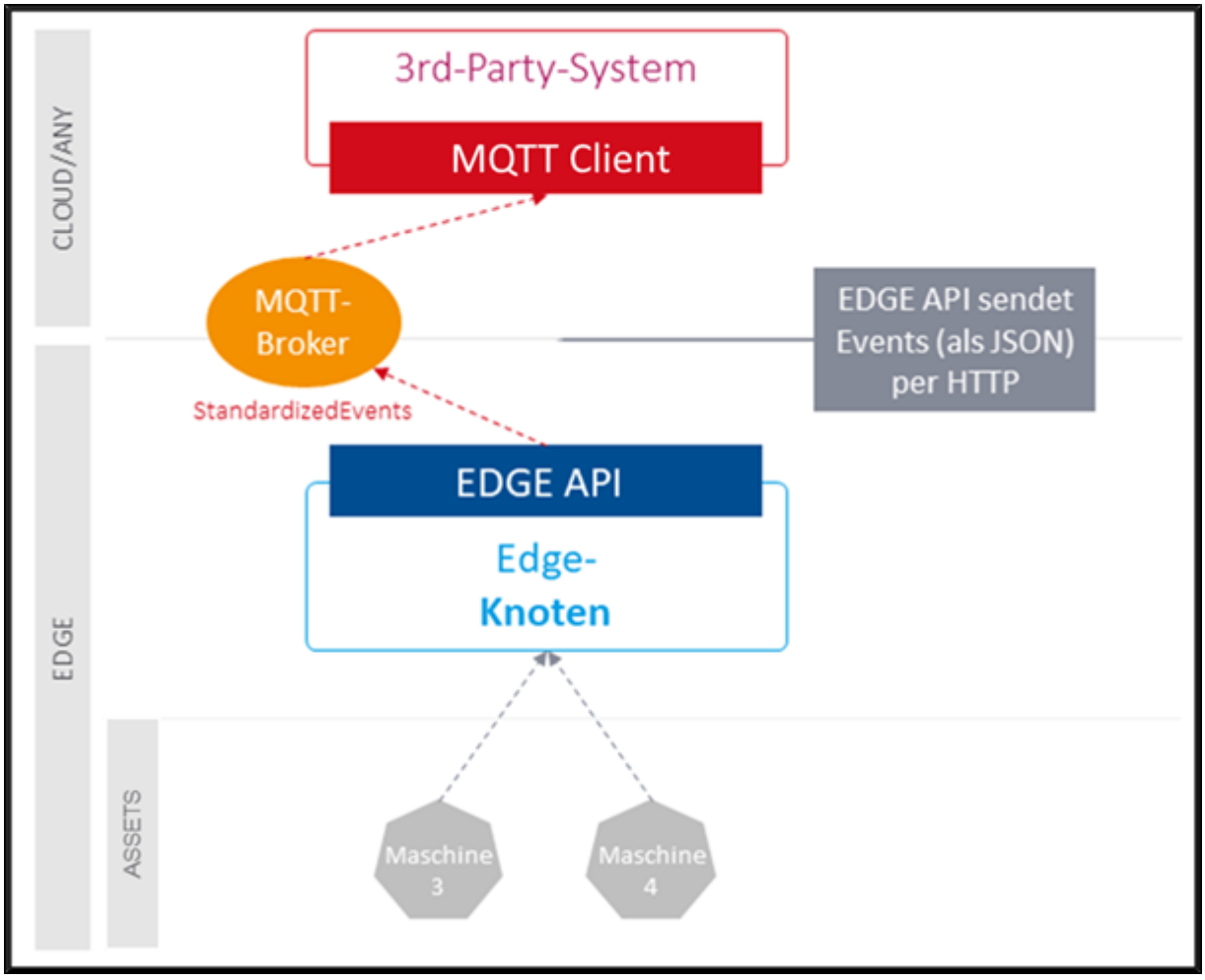


Bild 31: Kommunikation mit übergeordneten Systemen über MQTT-Broker

Apache Kafka

Die Versorgung des Drittsystem kann durch die Unterstützung von Apache Kafka erfolgen, sofern es durch den Kunden oder Partner bereitgestellt wird.

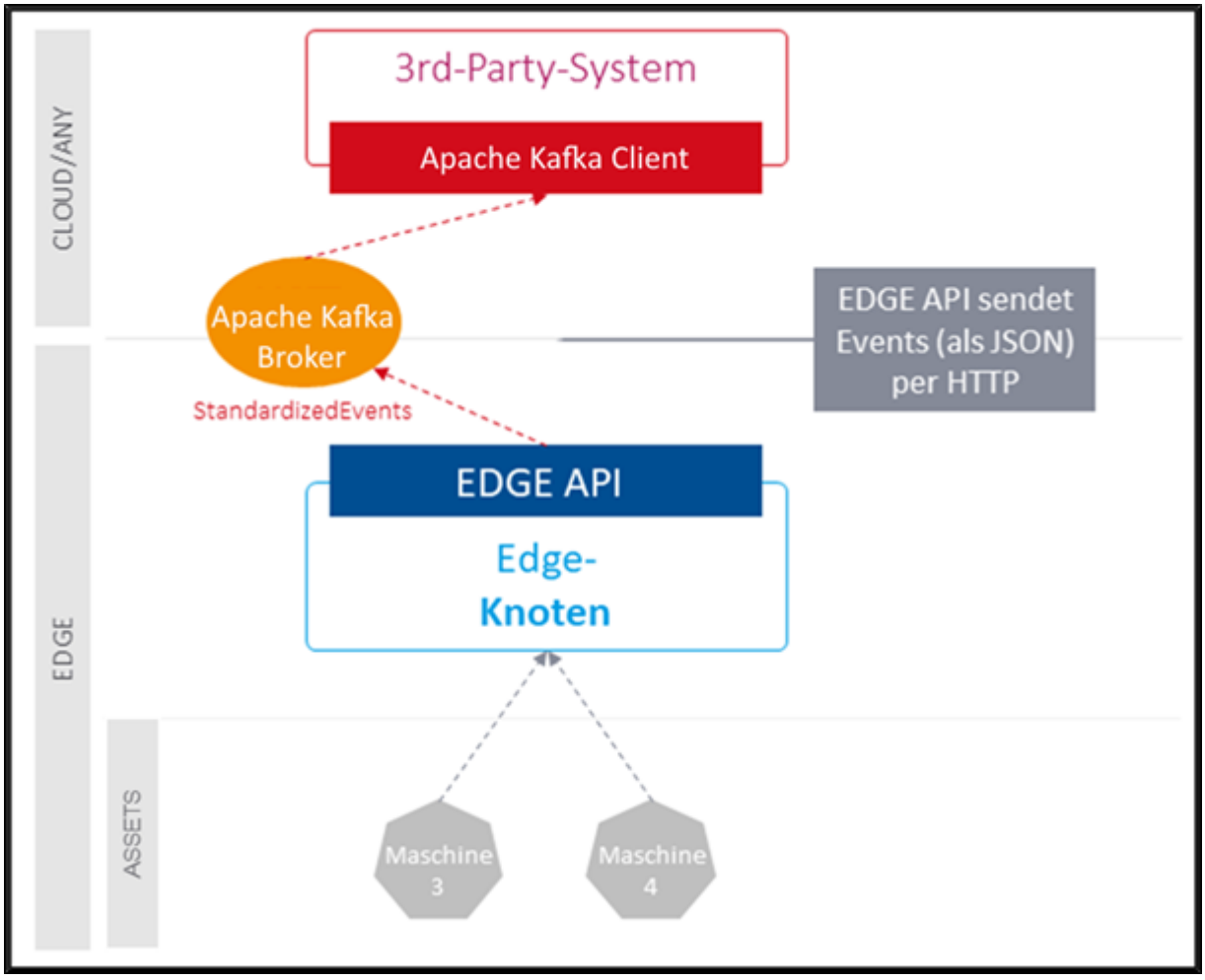


Bild 32: Kommunikation mit übergeordneten Systemen über Apache Kafka

OPC UA

FORCAM stellt einen OPC UA-Server mit der Funktionalität „Data Access“ zur Verfügung. Durch diese Erweiterung werden verschiedene Asset-Daten über die definierte OPC UA-Schnittstelle bereitgestellt. Die Informationsmodelle werden dynamisch auf der Grundlage der vorhandenen, konfigurierten Assets im EDGE-Knoten vorbereitet.
Über die angegebene URL verbindet sich der Benutzer mit dem Server, um die gewünschten Daten abzurufen. Der notwendige Client für den Abruf wird vorausgesetzt.
Es ist nicht nur möglich, die aktuellen Werte eines Events oder Signals abzufragen, sondern ebenfalls den Verlauf. Um diese historischen Datensätze verarbeiten zu können, unterstützt EDGE CONNECT die **Historian**-Funktionalität von OPC UA. Das Historian fungiert als Datenlogger mit SQL-Datenbanken. Es protokolliert Verlaufsdaten und kann zusätzlich als Gateway für den Zugriff auf Echtzeitdaten von allen zugrundeliegenden OPC UA-Servern verwendet werden.
In EDGE CONNECT hat der Benutzer die Möglichkeit, Anmeldeinformationen zu vergeben. Diese Daten müssen dann bei der Verbindung mit dem Server über den Client korrekt eingegeben werden.

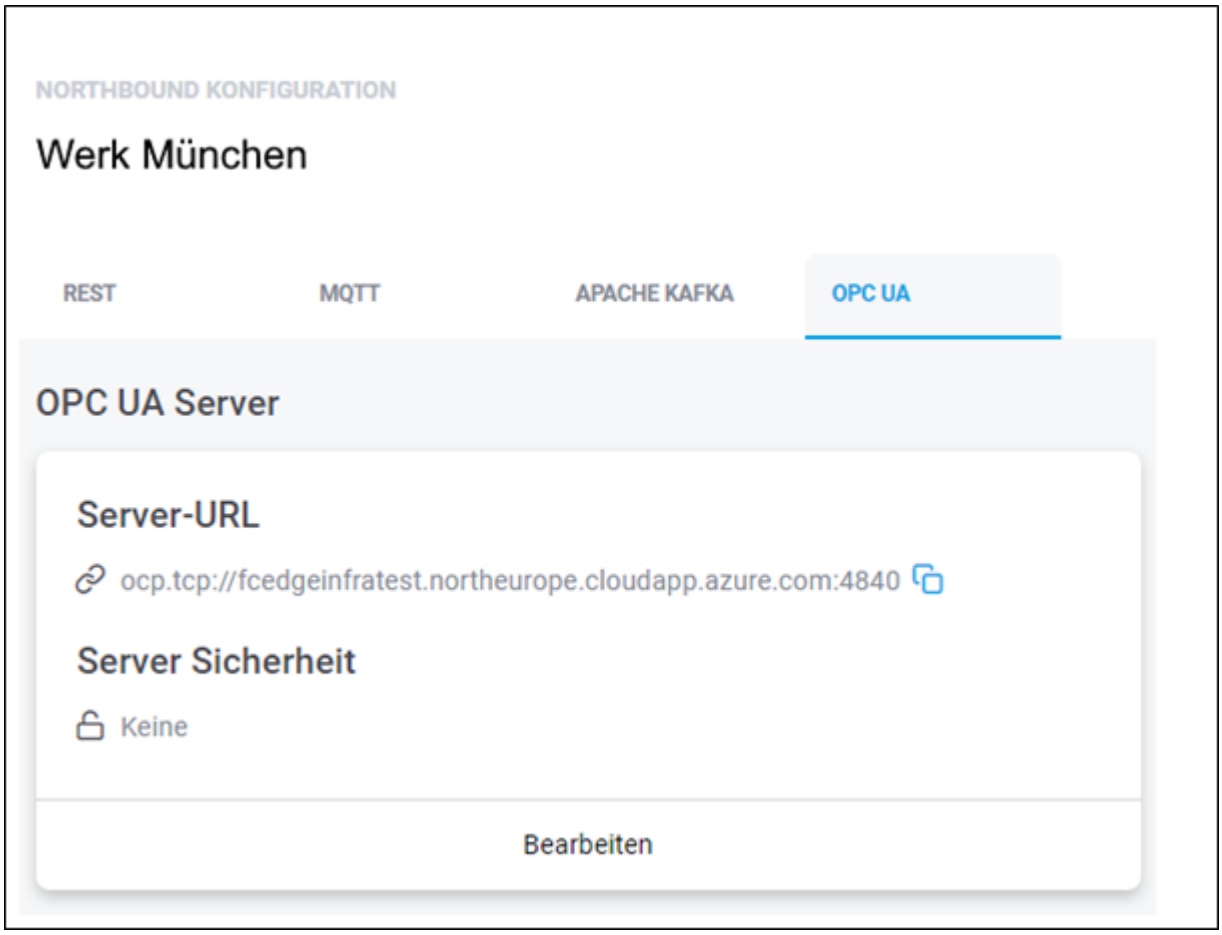


Bild 33: Eventkonfiguration über OPC UA

NATS.io

Die EDGE CONNECT ermöglicht zudem die Anbindung an eine NATS-Infrastruktur für das Versenden von Northbound-seitigen Informationen. Das Empfangen von Informationen (z. B. Business Parameter) wird durch die NATS-Schnittstelle nicht unterstützt.

Für die Übertragung der Events wird sowohl NATS Core, als auch JetStream unterstützt. In beiden Fällen können pro Event unterschiedliche Subjects und auch Streams angegeben werden, um eine Verteilung der Daten zu ermöglichen.
Der Inhalt der zu übertragenden Events kann durch den bestehenden Platzhalter-Mechanismus frei konfiguriert und an das Zielsystem angepasst werden.

Allgemein empfehlen wir die Erstellung von einem Subject je Asset und die Nutzung eines angepassten Skripts zur optimalen Integration der Subjects.

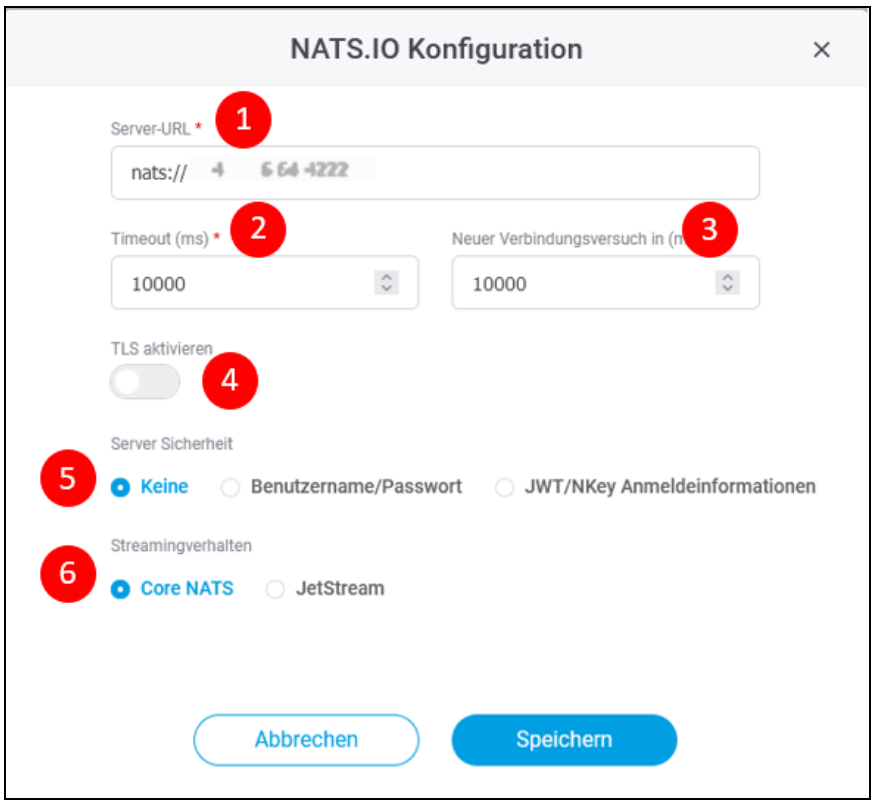


Bild 32: Konfiguration der NATS.io Verbindung

(1) Server-URL: Textfeld für die URL des Servers
(2) Timeout (ms): Begrenzt die Dauer eines Verbindungsversuchs zum Server
(3) Neuer Verbindungsversuch in (ms): Wartezeit für einen erneuten Verbindungsversuch zum Server
(4) TLS aktivieren: Aktiviert die Verschlüsselung über SSL/TLS
(5) Server Sicherheit
    a. Keine: Keine Authentifizierung
    b. Benutzername/Passwort: Authentifizierung über User und Passwort
    c. JWT/NKey Anmeldeinformationen: Authentifizierung mit Zertifikatsdatei
(6) Streamingverhalten:
    d. Core NATS: Gesendete Nachrichten werden nicht gespeichert.
    e. JetStream: Gesendete Nachrichten werden gespeichert. Dabei gilt, dass die Nachricht mindestens einmal übermittelt wird (Quality of Service Level 1).

6.4.2 Daten & Dokumente vom übergeordneten System zu EDGE

Über die EDGE API kann die EDGE CONNECT mit Daten und Dokumenten versorgt werden.
Technisch ist die Übertragung von NC-Dateien über **HTTP/REST** möglich. Das Schreiben von Business Parameter und Signalwerten ist neben **HTTP/REST** auch über **MQTT** ausführbar.
Folgende Schnittstellen werden bereitgestellt:

Tabelle 2: Schnittstellen für die Übertragung von Daten und Dokumenten

Schnittstelle	Beschreibung
Übertragen von Prozess- und Referenzparametern	Diese Business-Parameter können im EDGE Composition Layer verwendet werden, um damit unter anderem standardisierte Events anzureichern (z. B. Auftragsnummer oder Taktzeit).
Übertragung von Signalparametern	Es können Parameterwerte für spezifische Signale übertragen werden. Diese werden direkt auf die Asset-Steuerung geschrieben.
Übertragung von Dokumenten	Es können NC-Programme übertragen werden, welche ebenfalls auf die Asset-Steuerung übertragen werden.

6.4.3 Event konfigurieren

1. In der Detailansicht eines konfigurierten Assets (siehe Bild 16im rechten oberen Bereich auf das Eventkonfiguration-Icon klicken.
2. Im Folgedialog bestimmen, ob **REST**, **MQTT oder KAFKA** verwendet werden soll.
3. In der oberen Leiste **URL** und optional weitere Angaben wie Timeout etc. eintragen. Bestimmt, wohin die Events geschickt werden sollen.
4. SSL-Zertifikatsprüfung bestimmen.
   Ist der Schalter **Überprüfe SSL-Zertifikat** aktiv, kann EDGE CONNECT über REST auch mit einer Anwendung ohne gültiges bzw. unsigniertes Sicherheitszertifikat verbunden werden.
5. Gewünschte Authentifizierung auswählen und Anmeldedaten eintragen.
6. Events wie gewünscht konfigurieren.
7. Speichern.

7 Monitoring

EDGE CONNECT hat die Option integriert, Komponenten und Erweiterungen über die Monitoring-Seite zu überwachen. Die Seite gibt Aufschluss darüber, ob die jeweilige Komponente fehlerfrei läuft, oder ob Störungen vorliegen. Das Monitoring ist über das Icon im rechten oberen Bereich in der Asset-Übersicht aufrufbar (siehe Bild 16).

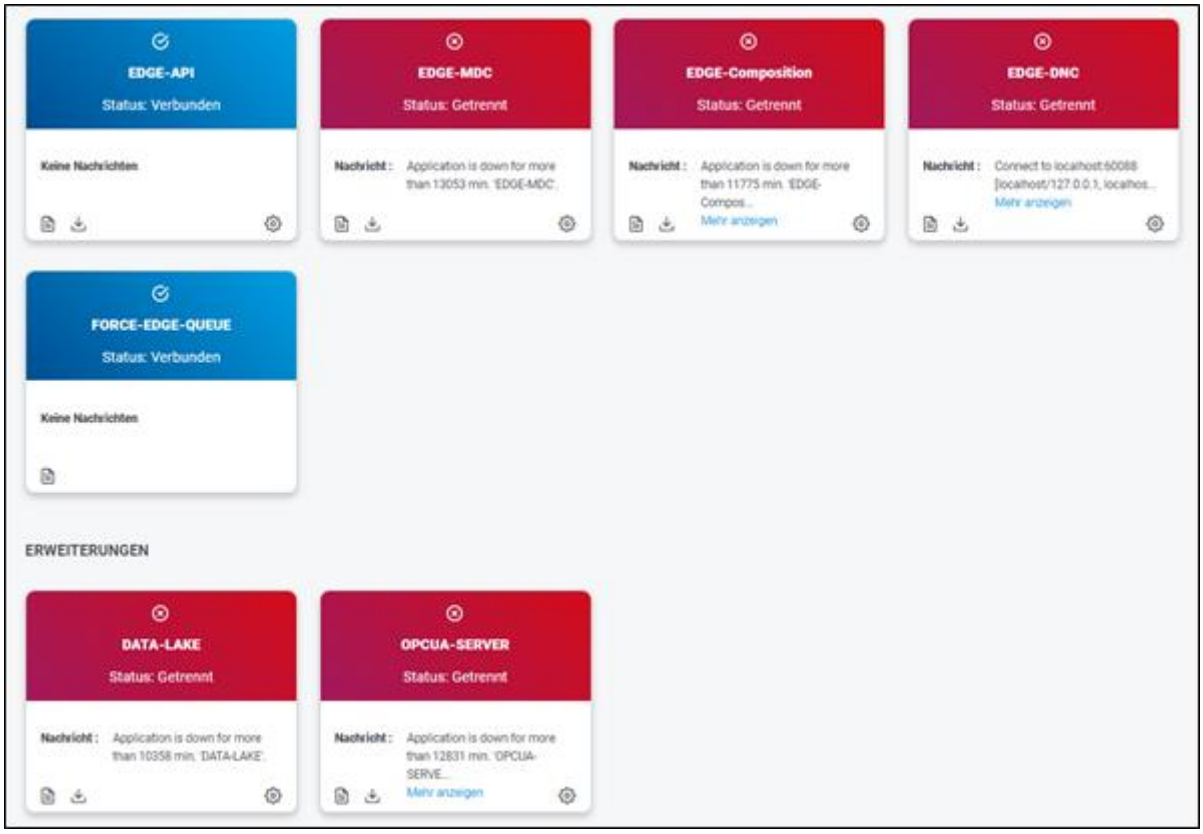


Bild 33: Monitoring in EDGE CONNECT

In jeder Komponente können Fehlermeldungen und Logs gezielt abgerufen werden.

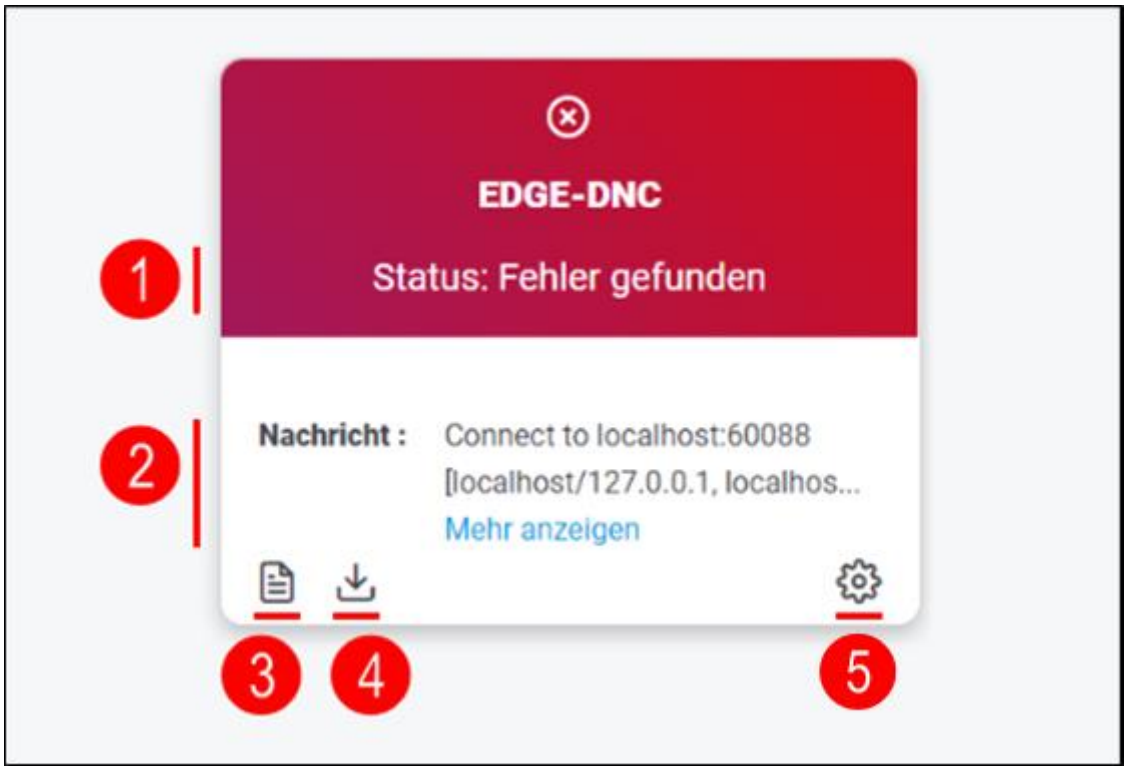


Bild 34: Komponente „EDGE-DNC“ in der Monitoring-Seite

(1) Aktueller Status der Komponente.
(2) Nachricht bei einem Fehler.
Durch Klicken auf **Mehr anzeigen** wird die vollständige Fehlermeldung in einem Pop-up angezeigt.
(3) Zeigt die jeweils letzte Warn- und Fehlermeldung der Komponente an
(4) Erlaubt das Herunterladen einer Logdatei von einem bestimmten Tag
(5) Einstellung des Log-Levels

8 Anhang

8.1 Dokument-Konventionen

Tabelle 3: Verwendete Schriftarten, Formatierungen und Zeichen

Konvention	Beschreibung
Fettschrift	Die Bezeichnungen von Schaltflächen und Optionen sind fettgeschrieben.
Kursivschrift	Hervorgehobene Wörter sind kursivgeschrieben.
Icons	Bei einer Funktion, die über ein Icon dargestellt ist, wird auf das Icon als Objekt referiert.
Handlungsresultat	Handlungsresultate sind durch ➔ gekennzeichnet.
Voraussetzungen	Voraussetzungen sind durch ✓ gekennzeichnet.
Warnungen	Warnungen sind durch ⚠ gekennzeichnet.
Hinweis	Hinweise sind durch ⓘ gekennzeichnet.
Tipps	Tipps sind durch ⓣ gekennzeichnet.

8.2 Abkürzungen und Begriffe

Tabelle 4: Verwendete Abkürzungen und Begriffe

Abkürzung	Erklärung
Apache Kafka	Apache Kafka ist ein verteiltes System, zur Nachrichtenübertragung das nach dem Publish-Subscribe Verfahren arbeitet.
Asset	Oberbegriff für alle Objekte, die EDGE CONNECT anbinden kann (z. B. Maschinen, Sensoren, Datenbanken, IT-Systeme etc.)
Brownfield	Eine Fabrik oder Fertigungsanlage, die bereits gebaut und schon seit einiger Zeit in Betrieb ist. Der Brownfield-Ansatz in Zusammenhang mit der Industrie 4.0 meint die digitale Transformation einer vorhandenen Fertigungsanlage.
CP	Communication Processor
DB	Datenbank
DNC	Distributed Numerical Control: NC-Anlagen, die mit einem Computer verbunden sind. Die Einzelanlagen können zentral mit NC-Programmen versorgt und koordiniert werden.
IT	Informationstechnik
Maschine	In EDGE CONNECT ist die Maschine eine Teilanlage nach ISA 95. Sind keine weiteren Teilanlagen (d.h. zusätzliche physikalische Steuerungen) vorhanden, sprechen wir von einer Anlage.

Abkürzung	Erklärung
MDC	Machine Data Connection (Maschinendatenerfassung)
MQTT	Message Queuing Telemetry Transport: offenes Netzwerkprotokoll für Machine-to-Machine-Kommunikation (M2M), das die Übertragung von Telemetriedaten in Form von Nachrichten zwischen Geräten ermöglicht, trotz hoher Verzögerungen oder beschränkter Netzwerke
MR	Machine Repository
NATS.io	Ein Konnektivitätssystem welches nach publish-subscribe Prinzip agiert. Das Messaging kann über verschiedene Optionen realisiert werden.
Northbound	Eine northbound-Schnittstelle kommuniziert in einer bestimmten Netzwerk-Komponente mit einem höher eingestuften Element.
OPC UA	Open Platform Communications Unified Architecture (Standard für den Datenaustausch als plattformunabhängige, service-orientierte Architektur)
OT	Operative Technologie
POST	POST ist eine Methode, die von HTTP unterstützt wird und darstellt, dass ein Webserver die im Body der Nachricht enthaltenen Daten annimmt, die angefordert werden.
PUT	Die PUT-Methode wird verwendet, um eine auf dem Server verfügbare Ressource zu aktualisieren. Typischerweise ersetzt sie alles, was an der Ziel-URL existiert, durch etwas anderes.
REST	Representational State Transfer: Programmierparadigma für verteilte Systeme (Zusammenschluss unabhängiger Computer, die sich für den Benutzer als ein einziges System präsentieren)
RESTful API	API für den Datenaustausch auf Basis von HTTP-Anfragen mittels GET, PUT, POST und DELETE, der den Anforderungen bzw. Beschränkungen der REST Architektur unterliegt.
Signal	Aus der Maschinensteuerung ausgelesene Werte wie z. B. Temperatur, Druck oder bestimmte Status.
Southbound	Äquivalent zur Northbound-Schnittstelle kommuniziert eine Southbound-Schnittstelle mit darunterliegenden Komponenten.
SPS	Speicherprogrammierbare Steuerung
UTC	Coordinated Universal Time (koordinierte Weltzeit)
Wildcard	Platzhalter für andere Zeichen

8.3 Liste unterstützter Plug-ins

MDC-Plug-ins

Tabelle 5: Liste aller unterstützter Maschinenanbindungsvarianten

Name	Lesen	Schreiben	Übertragungsart Polling/Eventbasiert
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/
Siemens LOGO	X	X	X/

Name	Lesen	Schreiben	Übertragungsart Polling/Eventbasiert
Siemens S5 mit CP	X		
Siemens S5 ohne CP	X		
Siemens S7 mit CP	X	X	X/
Siemens S7 ohne CP	X	X	X/
SQL Database Exchange	X		X/
Weihenstephan	X		X/
Wiesemann & Theis (WUT)	X		X/

DNC-Plug-ins

Tabelle 6: Liste aller unterstützter NC-Maschinen-Anbindungsvarianten

Name	Lesen	Schreiben
COM	X	X
Heidenhain	X	X
Mazak-DNC	X	X
RPC Plug-in	X	X
FTP Plug-in	X	X
FANUC	X	X
File Handler (File Copy)	X	X
File Handler Server	X	X
MOXA-Box	X	X

8.4 Standardisierte Events

Tabelle 7: Events und deren Funktion in EDGE CONNECT

Event-Typ	Werte	Funktion
Allgemeine Information	– Machine Id – Machine Name – External Machine Id – Reference (any) – Timestamp – Type (any) – Value (any)	Beliebige Information
Impuls	– Machine Id – Machine Name – External Machine Id – Reference (any) – Timestamp – Count	Z. B. Hub, Schuss etc.
Menge	– Machine Id – Machine Name – External Machine Id – Reference (any) – Timestamp – Amount – Unit (optional) – QualityDetail (optional)	Produzierte Menge
Signalpaket	– Machine Id – Machine Name – External Machine Id – Reference (any) – Timestamp – ARRAY [SignalName, Value, TimeStampUTC, Unit (optional)]	Sammlung von Signalen (z. B. Seriennummer, Druck und Temperatur)
Signalwert	– Machine Id – Machine Name – External Machine Id – Reference (any) – Timestamp – SignalName – Value – Unit (optional)	Temperatur, Druck etc.
Zustand	– Machine Id – Machine Name	Zustand der Maschine (läuft, läuft nicht)

Event-Typ	Werte	Funktion
	– External Machine Id – Reference (any) – Timestamp – State (Production or Downtime) – StatusCodes (optional list of statuses)	

8.5 Skript-Beispiele

8.5.1 Asset-Status und Temperatur

Durch das folgende Skript wird der Status der Maschine versendet (Produktion oder Stillstand). Zusätzlich wird die Temperatur ausgegeben. Sobald sich die Temperatur ändert, wird die aktualisierte Temperatur versendet.

```
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;
```

8.5.2 Temperatur und Luftfeuchtigkeit

Durch das folgende Skript werden die aktuelle Temperatur und Luftfeuchtigkeit versendet. Dies passiert im Intervall von 30 Sekunden und sobald eine Änderung dieser Werte stattfindet.

```
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;
```

8.5.3 Kransteuerung

Dieses Skript erfasst eine Kransteuerung mit den Knöpfen Schwarz, Grün und Rot.

- Der schwarze Knopf schaltet die Maschine an und aus
- Der rote Knopf löst einen Notfall aus
- Der grüne Knopf sendet einen Impuls für Stückzahlen und zählt diesen anschließend hoch

```
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 > karenzZeit 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;
```

8.5.4 Signalpakete

Das folgende Skript ist ein Beispiel für Signal Packages (Signalpakete):

```
//
// 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;
    H4Old: number;
    H5Old: number;
    H6Old: number;
    H7Old: 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;
```

8.6 Skriptfunktionen

Verwendung	Skriptfunktion Parameter in [...] sind optional	Beschreibung	Output-Event
Standard	SendImpulse(ImpulseCount, [Reference])	Sendet Impulse.	Impulse
Standard	SendQuantity(Quantity, [Unit], [QualityDetail], [Reference])	Sendet Menge.	Quantity
Custom	SendState(State, [StatusCodesListName], [Reference])	Sendet Status.	State
Standard	SendStateProduction([StatusCodesListName], [Reference])	Sendet Produktionsstatus.	State
Standard	SendStateStoppage([StatusCodesListName], [Reference])	Sendet den Zustand Stopp.	State
Standard	SendSignalValue(SignalName, Value, [Unit], [Reference], [CustomerSpecificSetting], [Timestamp])	Sendet den Wert eines Signals. Liste für Timestamp muss mit dem Datentyp "Long" (L) angelegt werden	SignalValue
Standard	SendSignalPackage(SignalNamesListName, ValuesListName, [UnitsListName], [Reference], [CustomerSpecificSetting], [TimestampsListName])	Sendet Signalwerte als Paket. Liste für Timestamp muss mit dem Datentyp "Long" (L) angelegt werden	SignalPackage
Custom	SendGenericInformation(ParamName, ParamValue, [Reference])	Sendet generische Informationen.	GenericInformation
Helfer	ListNew(ListName, DataType)	Erstellt eine neue Liste mit dem Namen ListName und Listenelementen vom Datentyp DataType (S für String, B für Boolean, N für Number).	-
Helfer	ListAdd(ListName, Value)	Fügt der Liste ein Element hinzu.	-
Helfer	ListClear(ListName)	Leert die Liste.	-
Helfer	ListDelete(ListName)	Löscht die Liste.	-

Verwendung	Skriptfunktion Parameter in [...] sind optional	Beschreibung	Output-Event
Helfer	GetMachineStatus()	Gibt den Assetstatus an.	-
Helfer	GetMachineData(ParameterName)	Gibt Assetdaten für den angegebenen Parameter an.	-
Helfer	SetParameter(ParameterName, ParameterValue)	Setzt einen neuen Wert für den angegebenen Parameter.	-
Helfer	GetParameter(ParameterName)	Ruft den Wert für den angegebenen Parameter ab.	-
Helfer	DeleteParameter(ParameterName)	Löscht den Parameter.	-
Helfer	DeleteAllParameters()	Löscht alle Parameter.	-
Helfer	OFFLINE	Merker, ob der Controller offline ist oder nicht.	-
Helfer	IPADDRESS	Die IP-Adresse der Composition.	-
Helfer	HOSTNAME	Hostname des Composition.	-
Helfer	SQRT(args)	Wurzelfunktion MATH.	-
Helfer	SIN(args)	Sinusfunktion MATH.	-
Helfer	COS(args)	Kosinusfunktion MATH.	-
Helfer	TAN(args)	Tangensfunktion MATH	-
Helfer	RISINGEDGE(args)	Zu Beginn ist die Variable FALSCH, die EDGE kontrolliert, ob sich die Werte verändert haben. Ist das der Fall, wird die Variable zu WAHR berichtigt.	-

Verwendung	Skriptfunktion Parameter in [...] sind optional	Beschreibung	Output-Event
Helfer	FALLINGEDGE(args)	Zu Beginn ist die Variable WAHR, die EDGE kontrolliert, ob sich die Werte verändert haben. Ist das der Fall, wird die Variable zu FALSCH berichtigt.	-
Helfer	SUBSTRING(str, startIndex[, endIndex])	Sub-string der angegebenen Zeichenkette.	-
Helfer	TONUMBER(str)	String zu Zahl (doppelt), ersetzt Komma zu Punkt im String.	-
Helfer	TOSTRING(str or number[, formatSpecifier])	Formatangabe der Formularbreite. Für leere Zeichenketten wird die Standardformatierung verwendet. Breite ist die Mindestlänge der Ergebniszeichenfolge. Präzision ist die Anzahl der Dezimalstellen. Wenn nicht angegeben, wird 0 verwendet. Wenn die Formatangabe mit 0 beginnt, werden der Ergebniszeichenfolge aufgefüllte Nullen vorangestellt. Wenn die Formatangabe mit X endet, wird die Zahl in hexadezimal umgewandelt, und zwar mit Groß- oder Kleinbuchstaben mit großem oder kleinem x. In diesem Fall werden die Dezimalstellen immer abgeschnitten.	-
Helfer	LENGTH(obj)	Die Länge eines Objekts als String-Wert.	-

Verwendung	Skriptfunktion Parameter in [...] sind optional	Beschreibung	Output-Event
Helfer	FORMATTIME(timeformatStr, timeOffset, [, timeunit])	Formatiert die aktuelle Zeit mit der Zeiteinheit als eines der folgenden: MILLISEKUNDE SEKUNDE MINUTE STUNDE TAG MONAT JAHR MSABSOLUTE (aktuelle Zeit) "R" bei Format wird als Zahl in Millisekunden angegeben, ansonsten wird das Format verwendet und Offset und Zeiteinheit zum Berechnen der Zeit verwendet.	-
Helfer	STDLOG(ignored, logLevel, suffixNumber, logText)	Der erste Parameter wird ignoriert. Die Log-Ebene sollte W = Warnung, C oder F = Fehler und alles andere für die Debug-Ebene sein. Die Suffix-Nummer, falls nicht 0, wird bei Skript-Logger als "(<SuffixNummer>)" am Ende des Log-Textes angehängt.	-
Helfer	DEBUGOUT(text)	Loggt den Text auf Debug-Log-Ebene mit Parser-Logger.	-
Helfer	COPYFILE(inFile, outFile)	Kopiert Daten von in-file nach out-file. Argumente können Dateipfade sein. Bei Erfolg wird auch die zuletzt geänderte out-file als in-file aktualisiert.	-
Helfer	COPYREPLACE(inFile, outFile, searchStr, replaceStr)	Kopiert von in-file nach out-file wie bei Funktion COPYFILE, und ersetzt dabei alle Vorkommen von search-string durch replace-string.	-
Helfer	ATTIME(seconds, obj)	Berechnet das Objekt jeden Tag zu vorgegebenen Zeiten im Zeitformat (Stunden: Minuten: Sekunden)	-
Helfer	FROMASCII(num)	Sendet eine Zeichenkette zurück, die den numerischen Wert als num hat.	-

Verwendung	Skriptfunktion Parameter in [...] sind optional	Beschreibung	Output-Event
Helfer	SLEEP(ms)	Pausiert den aktuellen Thread für eine bestimmte Zeit in ms.	-

8.7 Abbildungsverzeichnis