ENISCO FORCAM
smart factory experts

FORCE EDGE CONNECT

FORCE EDGE CONNECT

Version 230721

Handbuch

Dokument:	Handbuch - FORCE EDGE CONNECT
Freigabedatum:	21.07.2023
Dokumentversion:	1
Autor:	FORCAM GmbH

Inhaltsverzeichnis

1 Über dieses Dokument

Dieses Dokument beschreibt die Verwendung von FORCAM FORCE EDGE CONNECT.
Das Handbuch erläutert die unterschiedlichen Komponenten und Funktionen zur Anbindung von Assets sowie die Möglichkeiten zur Signalinterpretation und -weiterleitung.

ⓘ 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.

Zielgruppe

Dieses Handbuch setzt unter anderem Grundkenntnisse in der Signalverarbeitung / elektronischen Datenverarbeitung voraus sowie Hardware-Kenntnisse bezüglich der verwendeten Steuerungen. Sollten Sie dazu keine oder wenige Kenntnisse haben, nehmen Sie sich die Zeit, sich mit den Grundlagen vertraut zu machen.

ⓘ Wir empfehlen Ihnen die Nutzung unserer Academy: **https://forcam.com/academie/**
Die FORCAM Academy bietet das Wissen zum effektiven Einsatz der Methoden für die digitale Transformation und der Technologien für die Smart Factory.
Unser Institutsteam begleitet Sie auf Basis von Lean Manufacturing und TPM-Methoden und unterstützt Sie dabei, Veränderungen im Unternehmen einzuleiten und die Technologien richtig einzusetzen.

ⓣ In unserem **Kundenportal** finden Sie alle Handbücher, Produktbeschreibungen und weitere Informationen zu Ihrem Release. Informationen zur grafischen Signalinterpretation finden Sie dort im **Handbuch - Grafische Komposition**.

2 Konzept

FORCE EDGE CONNECT (im Folgenden nur noch EDGE CONNECT genannt) bietet produzierenden Unternehmen eine Lösung nahezu alle Assets zu 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 Fertigung im Green- und 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. Die Kluft zwischen IT (Informationstechnik) und OT (operative Technologie) wird geschlossen.

EDGE CONNECT bietet viele Möglichkeiten zur Anbindung von Assets und liefert deren Signal als standardisierte Events an übergeordnete Systeme. Diese können unter anderem ME- (Manufacturing Execution) oder MOM- (Manufacturing Operation Management) Systeme wie beispielsweise SAP DM/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. 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 von Assets zur Verfügung.

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 Greenfield- und Brownfield-Umfeld ist es wichtig, nicht nur die Signale abzugreifen und weiterzureichen, sondern diese auch für die Weiterverarbeitung zu interpretieren. Diese Aufgabe übernimmt die Komponente Signal Composition. 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.

Das EDGE CONNECT Machine Repository als optionale Erweiterung ermöglicht die einfache Erstellung und Verwendung von Templates. Templates sind Vorlagen für die Anbindung von Assets. Sie enthalten bereits alle wichtigen, allgemeinen Informationen. Nur individuelle Informationen, wie IP-Adresse oder Seriennummer, müssen ergänzt werden. Das MR erlaubt es, für Asset-Anbindungen Templates zu definieren oder aus bestehenden Anbindungen abzuleiten und für die Anbindung von gleichen Assettypen zu verwenden. Die Template-Struktur sorgt für eine standardisierte Anbindung von gleichen Assets und ermöglicht dadurch die Vergleichbarkeit von Assets des gleichen Typs. Dadurch wird der individuelle Aufwand für die Anbindung eines Assets noch einmal deutlich reduziert, wodurch die zeit- und ressourcenschonende Umsetzung von Digitalisierungsprojekten ermöglicht wird.
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.

3 Systemkomponenten

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

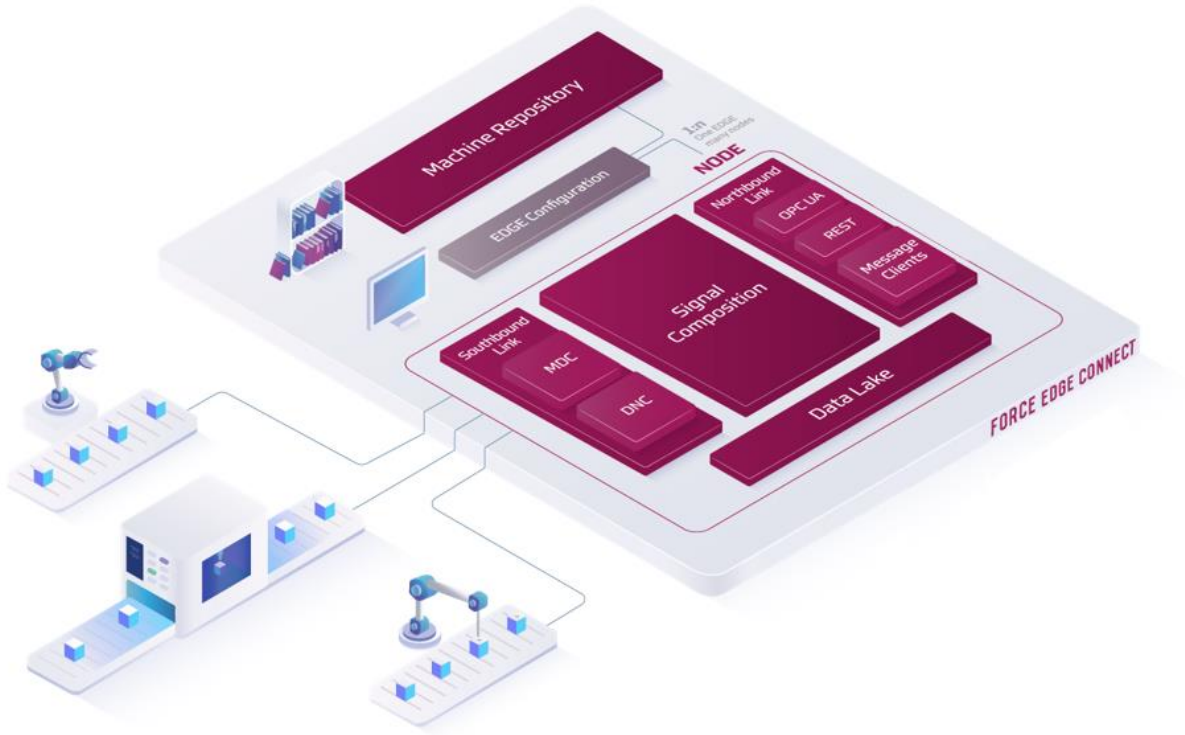


Bild 1: Schematischer Aufbau von EDGE CONNECT

3.1 EDGE Node

Der Node ist das wesentliche Element der EDGE CONNECT zur Asset-Anbindung. Er besteht aus den folgenden zentralen Komponenten:

3.1.1 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“.

Plug-ins

Plug-ins im Umfeld von EDGE CONNECT implementieren die Kommunikationsverbindung mit spezifischen Asset-Steuerungen. Außerdem sorgen sie für eine erste 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.

Die Plug-ins gliedern sich in Plug-ins zur Maschinendatenerfassung (MDC) und für Distributed Numerical Control (DNC):

- MDC-Plug-ins zur Maschinendatenerfassung

  Dazu gehören sowohl Plug-ins zum unidirektionalen Auslesen von Asset-Signalen als auch solche für eine bidirektionale Signalübertragung, also dem Auslesen und Zurückschreiben von Signalen.
- DNC-Plug-ins zum Übertragen und Auslesen von NC-Dateien
  Mit Hilfe dieser Plug-ins werden NC-Programme an das Dateisystem des Assets übertragen oder das an dem Asset aktive Programm abgefragt.

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

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

3.1.2 Signal Composition

Diese Komponente ermöglicht das Ableiten von logischen Asset-Zuständen. Dadurch lassen sich aus Signalkombinationen standardisierte Events ableiten. Events sind Nachrichten, die an ein Drittsystem gesendet werden.
Die Signalkomposition bietet auch die Möglichkeit, auf Ereignisse zu reagieren und Werte in die Steuerungseinheit des Assets zu schreiben (sofern dies von der Steuerung/dem verwendeten Protokoll unterstützt wird).
Eine solche Signalkomposition kann in der EDGE CONNECT entweder über ein Skript oder eine grafische Lösung umgesetzt werden. Die grafische Komposition ermöglicht einen leichten Einstieg in die Welt der Signalkomposition. (Weitere Informationen zu diesem Editor finden Sie im **Handbuch - Grafische Komposition**.)

3.1.3 Northbound Link

Der Northbound Link stellt die Assetdaten aus der Signal Composition der EDGE CONNECT heraus einem beliebigen Drittsystem zur Verfügung. 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“.

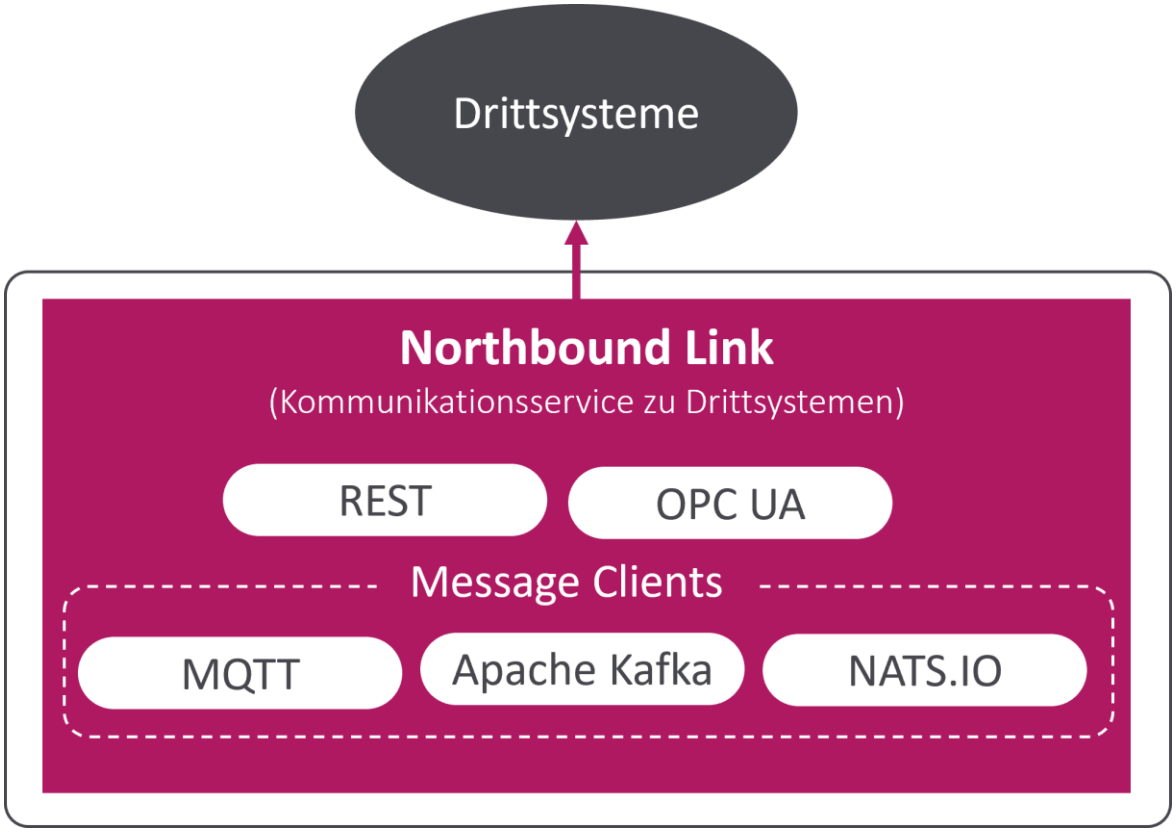


Bild 2: Northbound Link

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 konfiguriert werden. Bei der Verwendung von MQTT, NATS.io und Apache Kafka ist ein Broker als Middleware notwendig.
Der Northbound Link wird mit vorkonfigurierten Standardevents zur Kommunikation mit der MES- oder ERP-Ebene ausgeliefert. Diese lassen sich bei Bedarf weiter individualisieren.

⚠ Die Middleware (Broker) 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 (Signal Composition) 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. Der Data Lake ist als Kurzzeitdatenspeicher konzipiert.

⚠ Der Data Lake muss zusätzlich zu EDGE CONNECT erworben werden.

3.2 EDGE Configuration

EDGE Configuration ist die Benutzeroberfläche 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 in der EDGE Configuration.

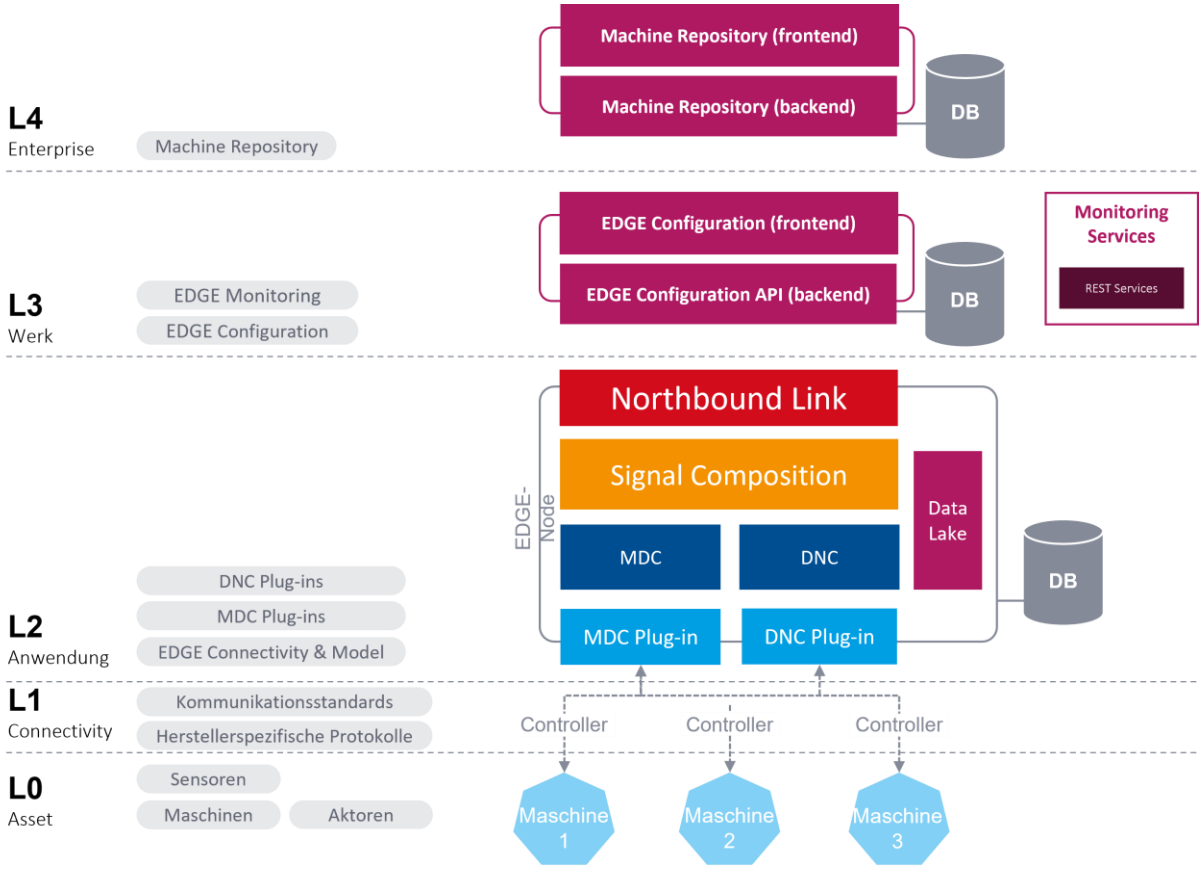
3.3 Machine Repository

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.

Das Machine Repository muss zusätzlich zu EDGE CONNECT erworben werden.

3.4 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.

Level 3 - Werk

Die Konfigurationskomponente kann mindestens 1 bis maximal 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 Configuration 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 3: Möglichkeiten der Installation von EDGE CONNECT

Für die Installation wird ein Installer bereitgestellt, in dem die EDGE Configuration, der EDGE Node und (optional) das 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 Assets 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.

ⓘ Weitere Informationen in den **Installation Guides** und **System Requirements**.

5 Grundlegende Einstellungen

Für allgemeine Einstellungen zu EDGE CONNECT steht das Menü (Bild 4) zur Verfügung.
Neben einer Einführung in die Tabellenhandhabung geht dieses Kapitel auf die folgenden Punkte ein:

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

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

Bild 4:Aufruf des Menüs in der EDGE CONNECT

5.1 Benutzerverwaltung

In der Benutzerverwaltung werden Benutzer für EDGE CONNECT erstellt. 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.

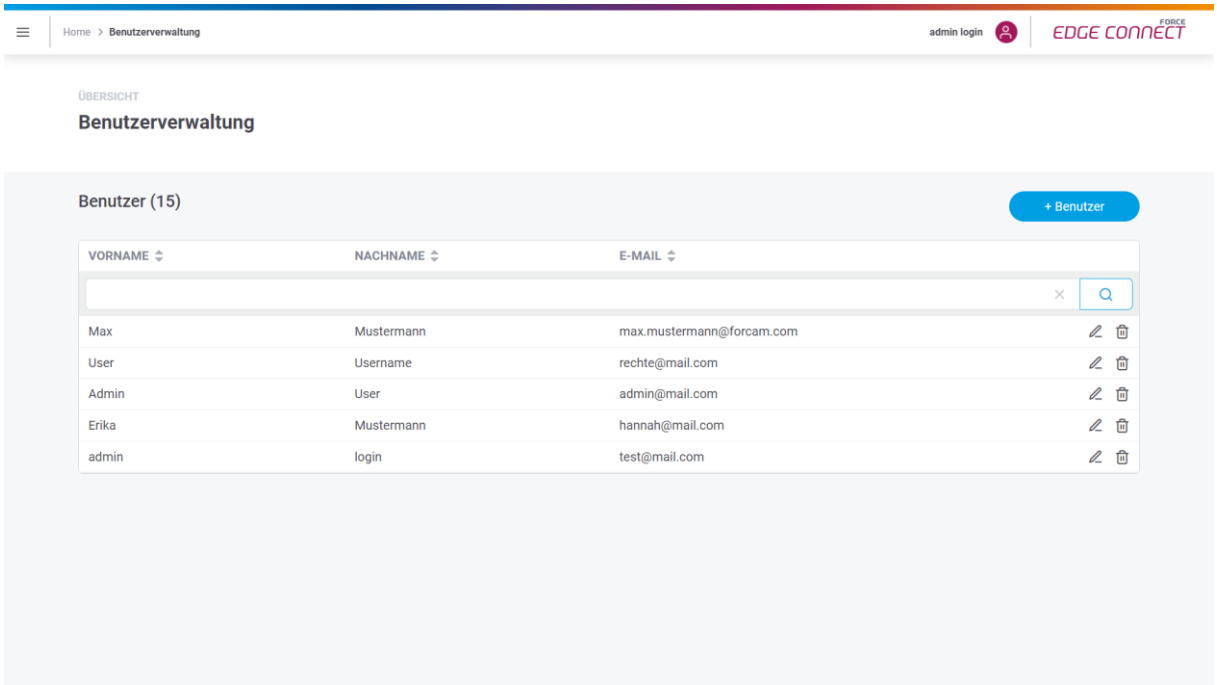


Bild 5: Benutzerverwaltung von EDGE CONNECT mit 5 Benutzern

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 (siehe Tabelle 1).
5. **Speichern**.

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

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 und anbinden.
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.

Tabelle 1: Benutzerrechte in EDGE CONNECT

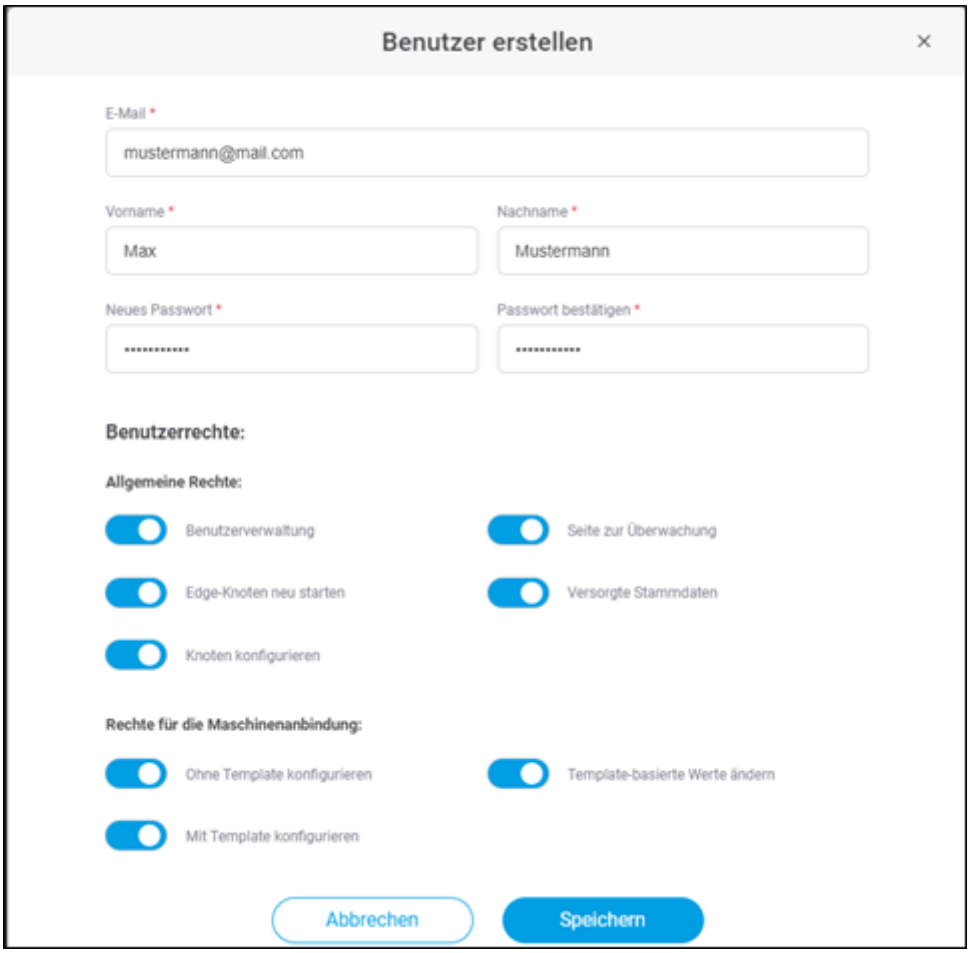


Bild 6: Dialog zu Erstellen eines neuen Benutzers

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 Drittanwendung wie z. B. SAP DM 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 aus Drittsystemen angelegt werden. Anbindungsspezifische Informationen können durch ein Template oder manuell eingetragen werden.
Das Löschen von Assets über die API ist nicht möglich. Sie können aber als gelöscht gekennzeichnet werden.

Werden die Stammdaten in der Drittanwendung verschickt, können die Daten genutzt werden, um in der EDGE Configuration neue Assets anzulegen. 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.

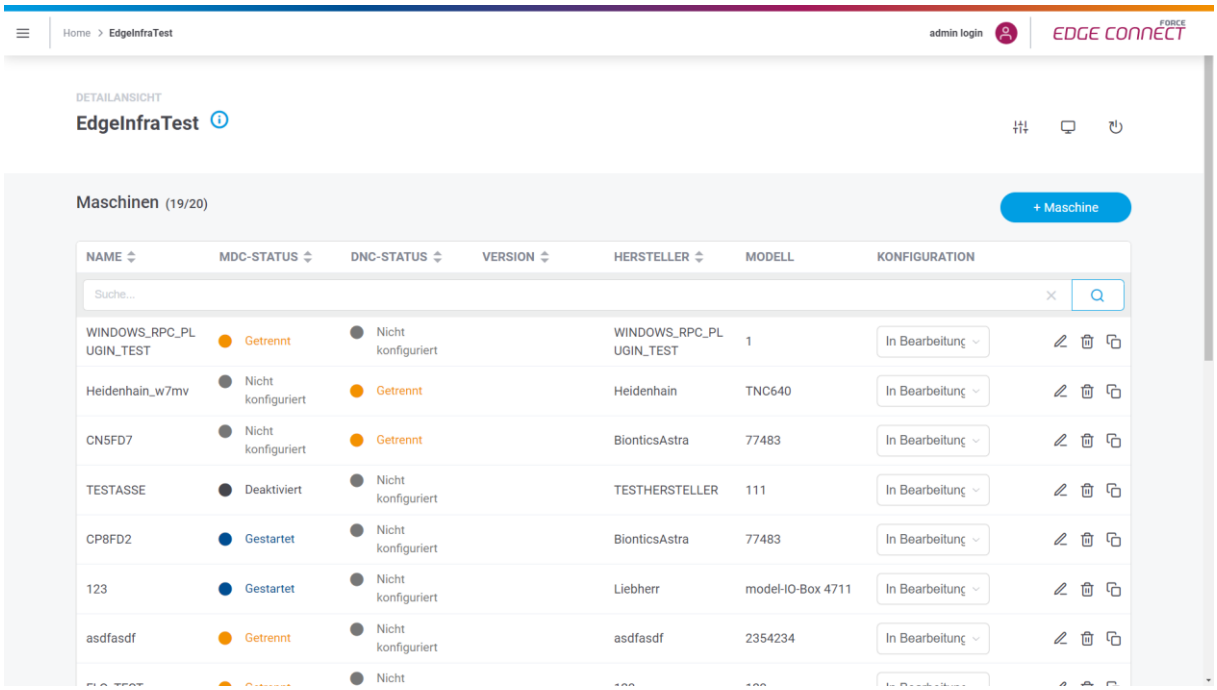


Bild 7: Liste von Assets mit versorgten Stammdaten

Folgende Stammdaten können über die API angelegt werden:

- Assetname
- Assettyp
- Assetklasse
- Hersteller
- Modell

- Seriennummer
- Externe Maschinen ID
- Inventar-Nr.

Asset fertig konfigurieren

Die versorgten Stammdaten können auf einem bestehenden EDGE-Knoten angelegt werden. Dazu öffnet sich der Konfigurationsdialog zum Hinzufügen eines Assets (siehe Abschnitt 6.3). Durch die API eingegangene Stammdaten werden dabei höher priorisiert.
Ein Beispiel: Ein Asset wurde über die API angelegt. Beim Konfigurieren wird später ein Template ausgewählt. Es werden dennoch weiterhin die Stammdaten verwendet, die über die API übergeben wurden. Die Stammdaten des Templates werden verworfen.

Um „Versorgte Stammdaten“ an die EDGE CONNECT zu versorgen:

1. Swagger aufrufen.
   IP-Adresse (der EDGE Configuration) + 60066/api/configuration/supplied-master-data
2. Unter **Supplied Master Data** POST konfigurieren und senden.
   - ➔ Asset ist unter **Versorgte Stammdaten** im Menü der EDGE CONNECT zu finden.

Um neue Stammdaten auf einem Knoten anzulegen:

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

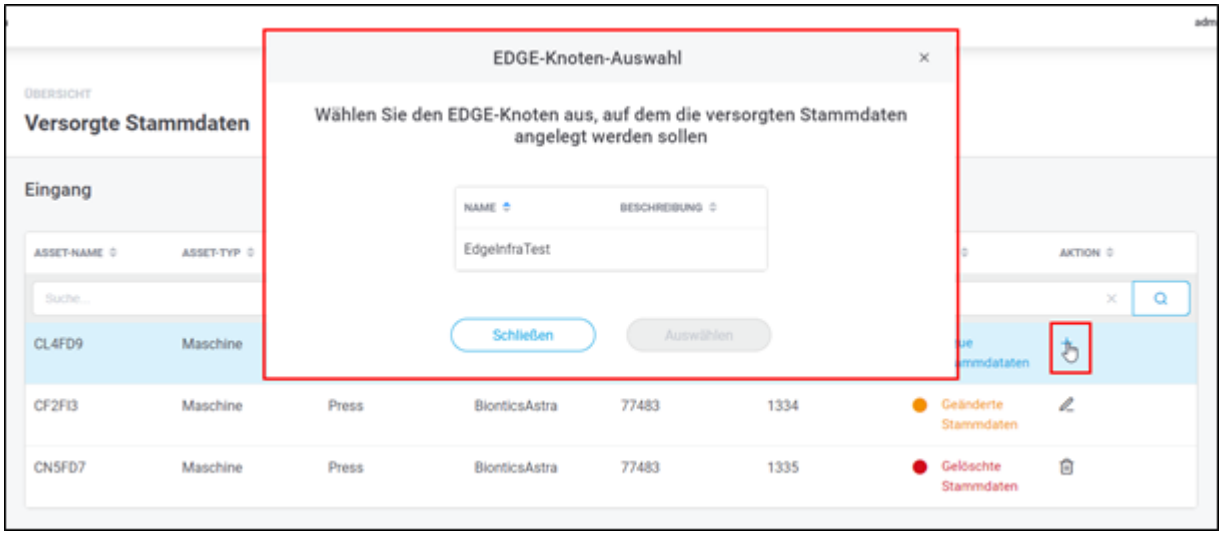


Bild 8: Auswahl eines EDGE-Knotens zum Anlegen von versorgten Stammdaten

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 auf das Stift-Icon (rechte Seite) 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. Optional: Den entsprechenden Knoten neu starten.

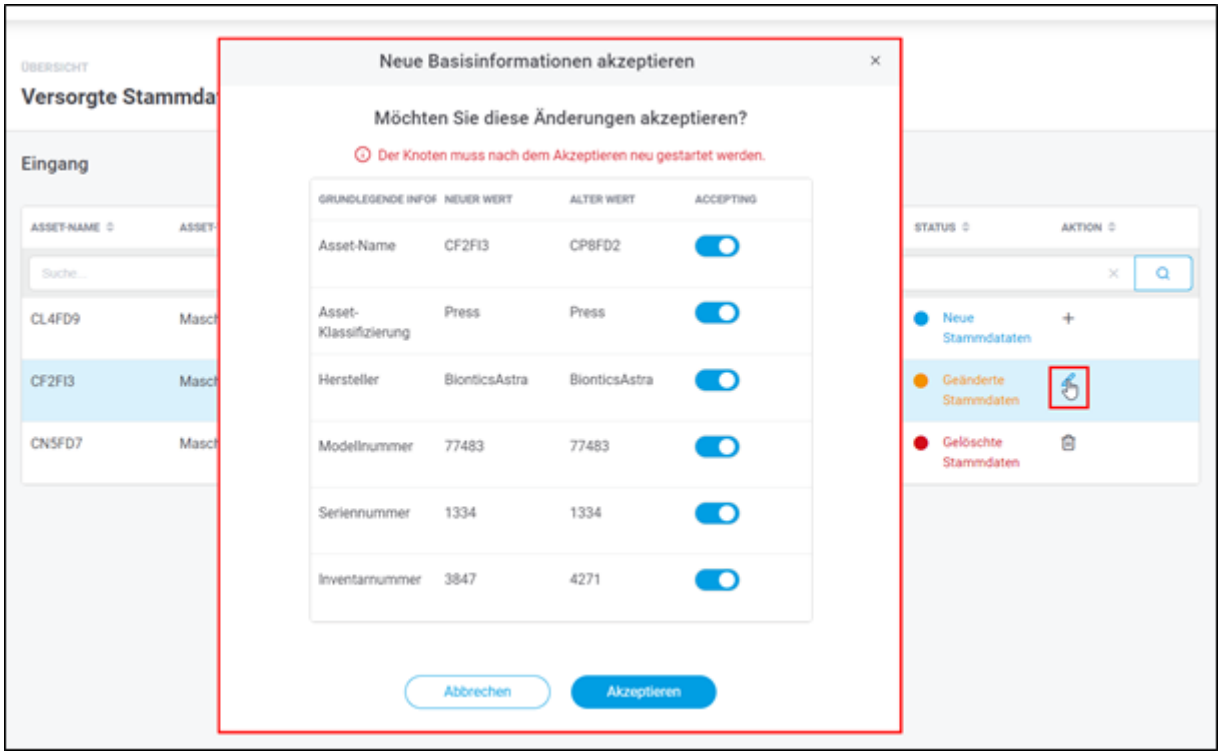


Bild 9: Bestätigung der Änderungen von Asset-Stammdaten

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 sowohl aus dem Knoten als auch aus der Tabelle der Seite entfernt.

5.3 Lizenzierung

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

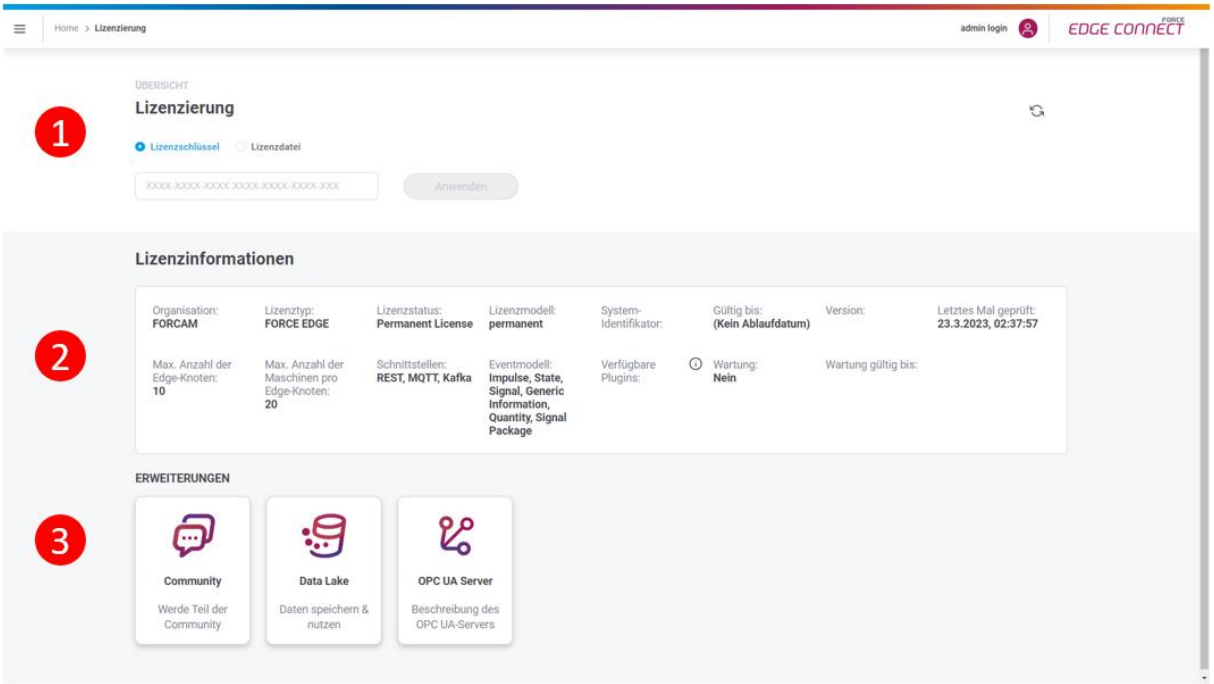


Bild 10: Lizenzierung und Übersicht

(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 PLUGINS** und **DNC PLUGINS** stellt FORCAM zusätzliche Anwendungen bereit. Diese werden benötigt, um über das entsprechende Plug-in mit einem Asset zu kommunizieren. Die Anwendungen ermöglichen eine bidirektionale Kommunikation.

5.5 Monitoring der EDGE Configuration

Das Monitoring im Menü dient zur Überwachung der EDGE Configuration. Die Überwachung der EDGE-Node-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 11: 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. Sie können die Spalten alphabetisch auf- oder absteigend sortieren.

Aufsteigend

Absteigend

Sortierung aus

Bild 12: 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.

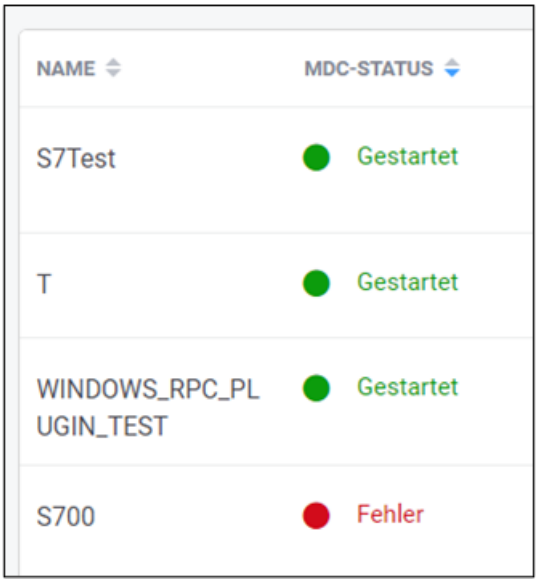


Bild 13: Alphabetische Sortierung und inhaltliche Gruppierung

6 EDGE Configuration

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.

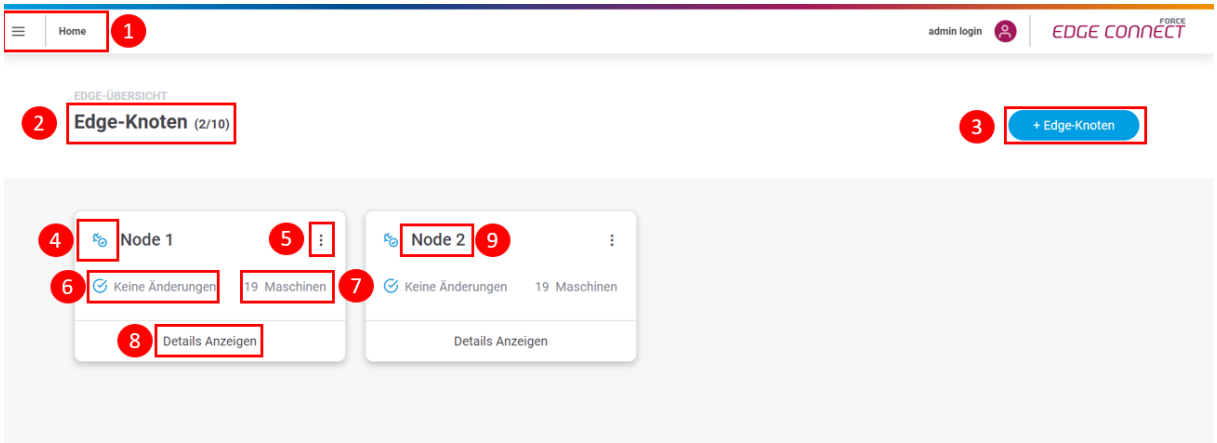


Bild 14: Einstiegs- und Übersichtsseite von EDGE CONNECT

(1) Home von EDGE CONNECT
(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
- Northbound Configuration
- Neustart des Knotens

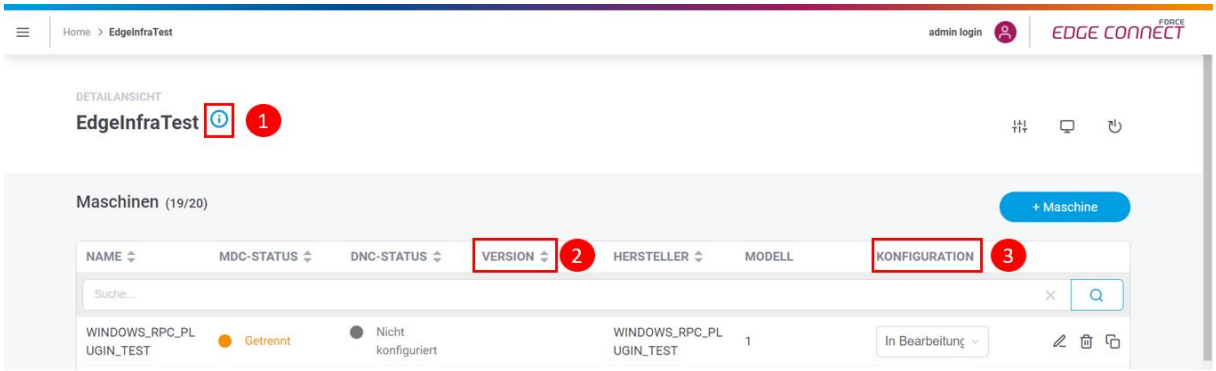


Bild 15: Asset-Übersicht als Folgeseite nach Klicken auf EDGE-Knoten

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 dem Nutzer einen Überblick zu geben:

- **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 kann ein Werk oder eine Linie in einem Werk abbilden. Pro Werk kann es mehrere Knoten geben. Sie werden logisch gebündelt, so dass die Last von Assets sinnvoll aufgeteilt werden kann.

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.

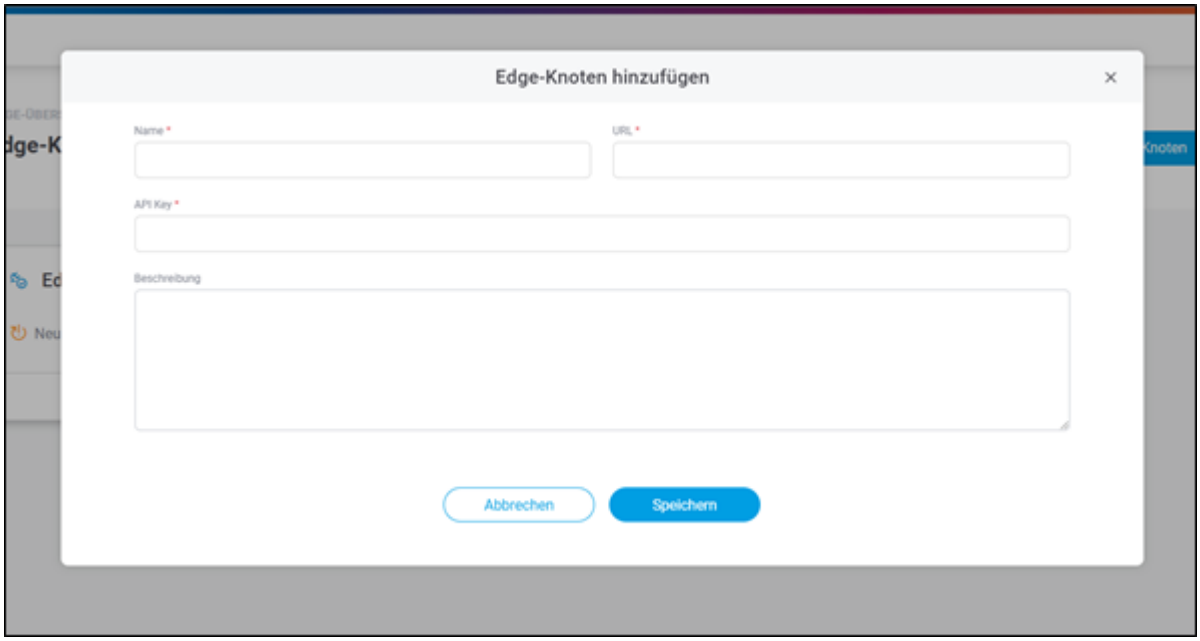


Bild 16: Dialog zum Hinzufügen eines neuen Knotens

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. Auch nicht auf anderen Konfigurationen.
- **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

Es ist 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.

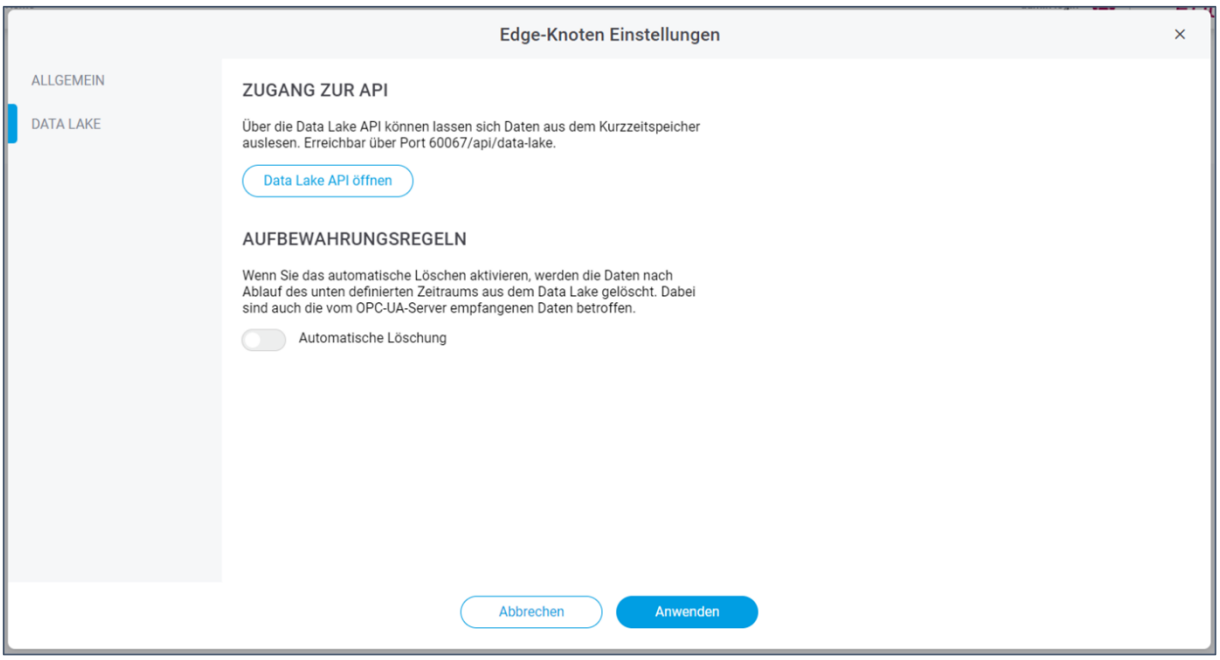


Bild 17: Dialog zum Bearbeiten eines bestehenden Knotens

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. Links zum Reiter **DATA LAKE** navigieren.
5. Werte bearbeiten.
6. **Anwenden** klicken.

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 Retention 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

Ein Configuration Wizard leitet durch acht Schritte, die für das Anbinden eines Assets 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.

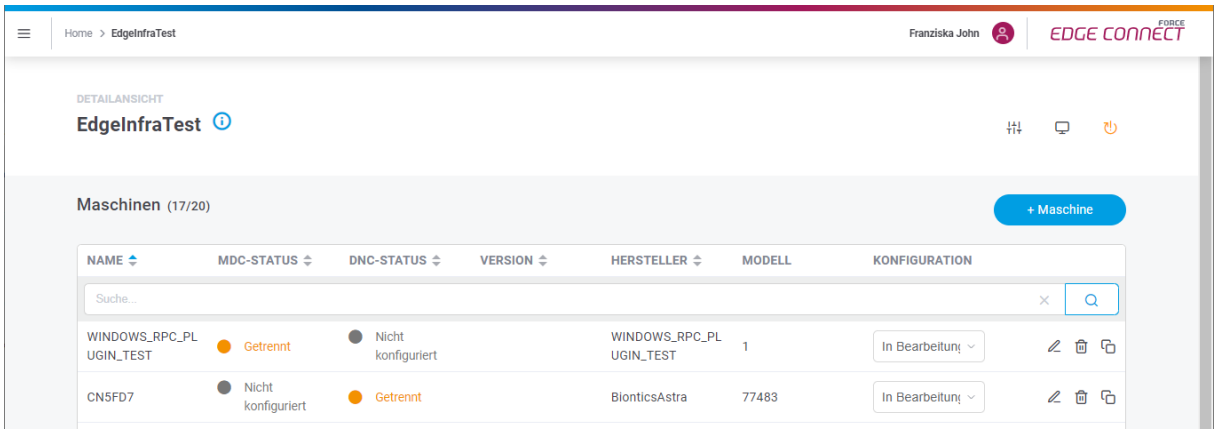


Bild 18: Dialog zur Konfiguration eines Assets in EDGE CONNECT

Um ein Asset hinzuzufügen:

1. In der Node-Detailansicht auf **+ Maschine** klicken.
   ➔ Der Configuration Wizard führt durch die nachfolgenden acht Schritte zur Konfiguration eines Assets.
2. Am Schluss **Anwenden** klicken.

⚠ Alle Einstellungen im Configuration Wizard müssen bei Schritt ⑧ mit **Anwenden** gespeichert werden, sonst geht die gesamte Konfiguration verloren.
Auch das Öffnen eines anderen Menüs führt zum Verwerfen der Konfiguration.

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 Informationen (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 welchem Überarbeitungsstand sich das Template befindet. Wird ein Template überarbeitet, wird die Versionsnummer automatisch um 1 hochgezählt 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 der Configuration Wizard mit Schritt ②.

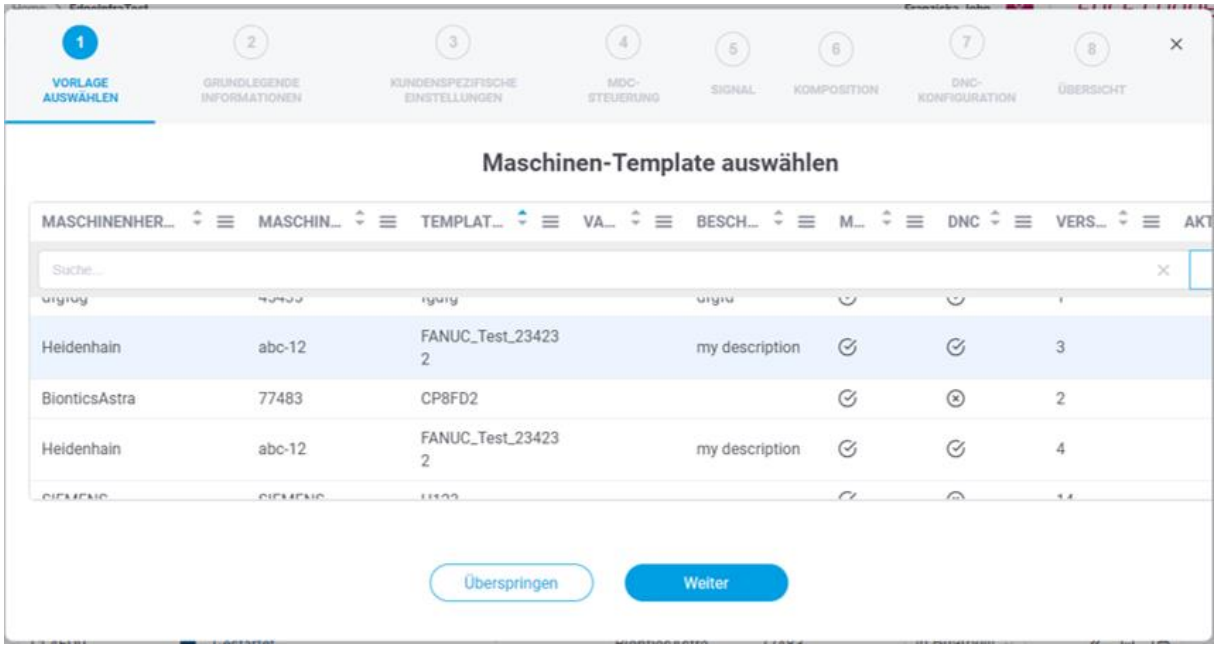


Bild 19: Configuration Wizard – Vorlage auswählen

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

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

6.3.2 ② Grundlegende Informationen

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

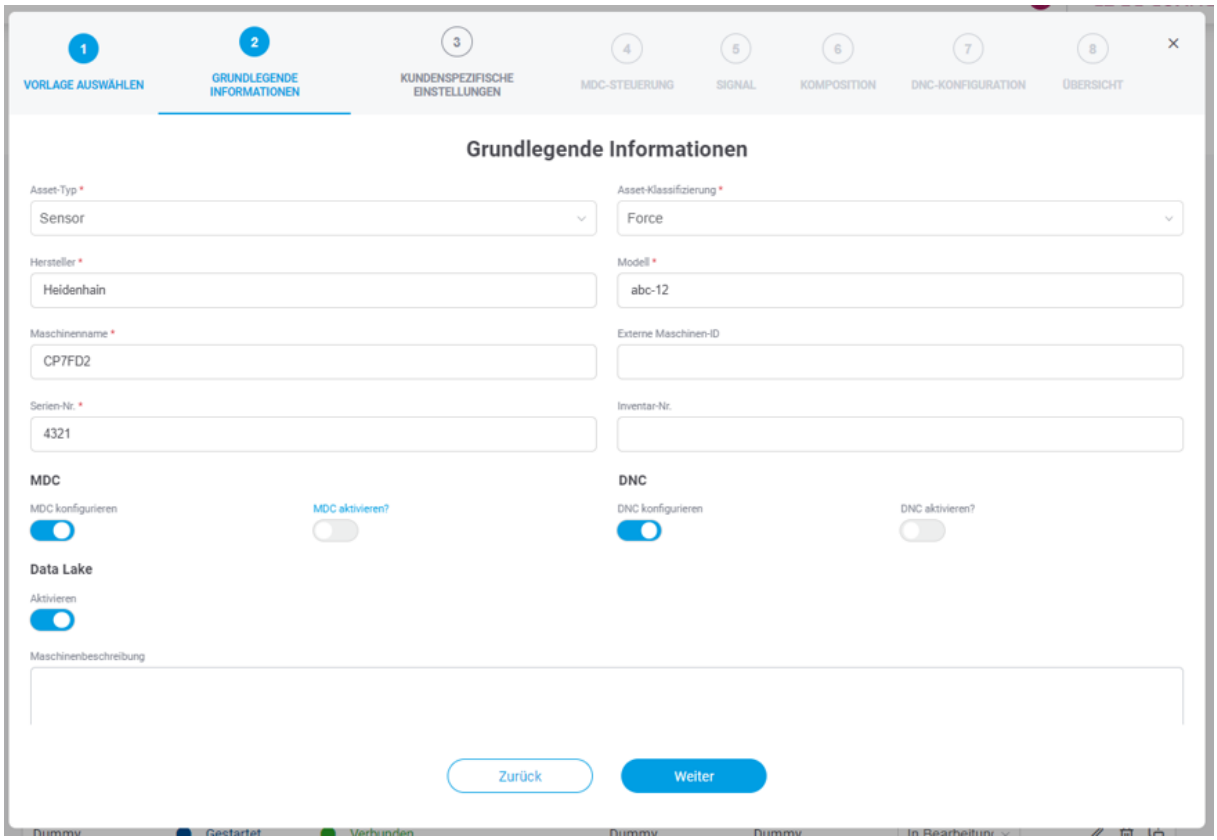


Bild 20: Configuration Wizard – Grundlegende Informationen

Eingabefeld	Beschreibung
Asset-Typ	Auswahl: – Maschine – Sensor – IT System
Asset-Klassifizierung	Auswahl variiert nach Eingabe im Feld Asset-Typ
Hersteller	Freies Eingabefeld
Modell	Freies Eingabefeld
Maschinenname	Freies Eingabefeld
Externe Maschine-ID	Freies Eingabefeld, optional
Serien-Nr.	Freies Eingabefeld

Inventar-Nr.	Freies Eingabefeld, optional
MDC **MDC konfigurieren** **MDC aktivieren?**	Schalter aktivieren, wenn MDC konfiguriert werden soll Schalter aktivieren, wenn MDC verwendet werden soll
DNC **DNC konfigurieren** **DNC aktivieren**	Schalter aktivieren, wenn DNC konfiguriert werden soll Schalter aktivieren, wenn DNC verwendet werden soll
Data Lake - Aktivieren	Schalter aktivieren, wenn Data Lake aktiviert werden soll (Schalter wird nur eingeblendet, wenn Data Lake verfügbar ist.)
Maschinenbeschreibung	Freies Eingabefeld, optional

6.3.3 ③ Kundenspezifische Einstellungen

Dieser Schritt bietet die 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.

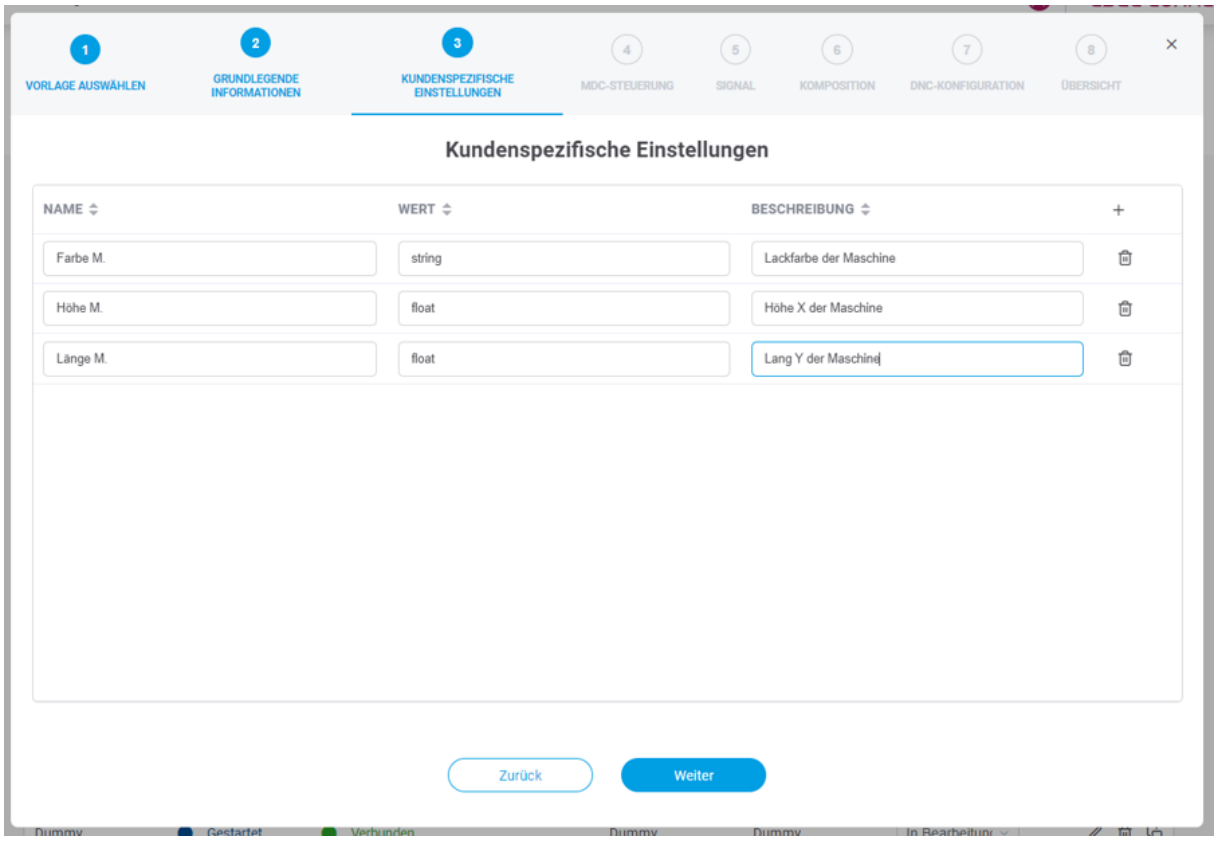


Bild 21: Configuration Wizard – Kundenspezifische Einstellungen

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

6.3.4 ④ MDC-Steuerung

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

Dieser Schritt bietet die Möglichkeit zur Konfiguration einer MDC-Steuerung. Sie bestimmt die Art und Weise, wie EDGE CONNECT mit dem Asset verbunden werden soll.
Eine Übersicht der aktuellen FORCAM Plug-ins ist in Abschnitt 0 aufgelistet.

Hier wählen Sie Typ und Bus-Typ. Je nach Auswahl variieren die weiteren Eingabefelder. Pflichtfelder sind mit einem Stern (*) gekennzeichnet.
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.

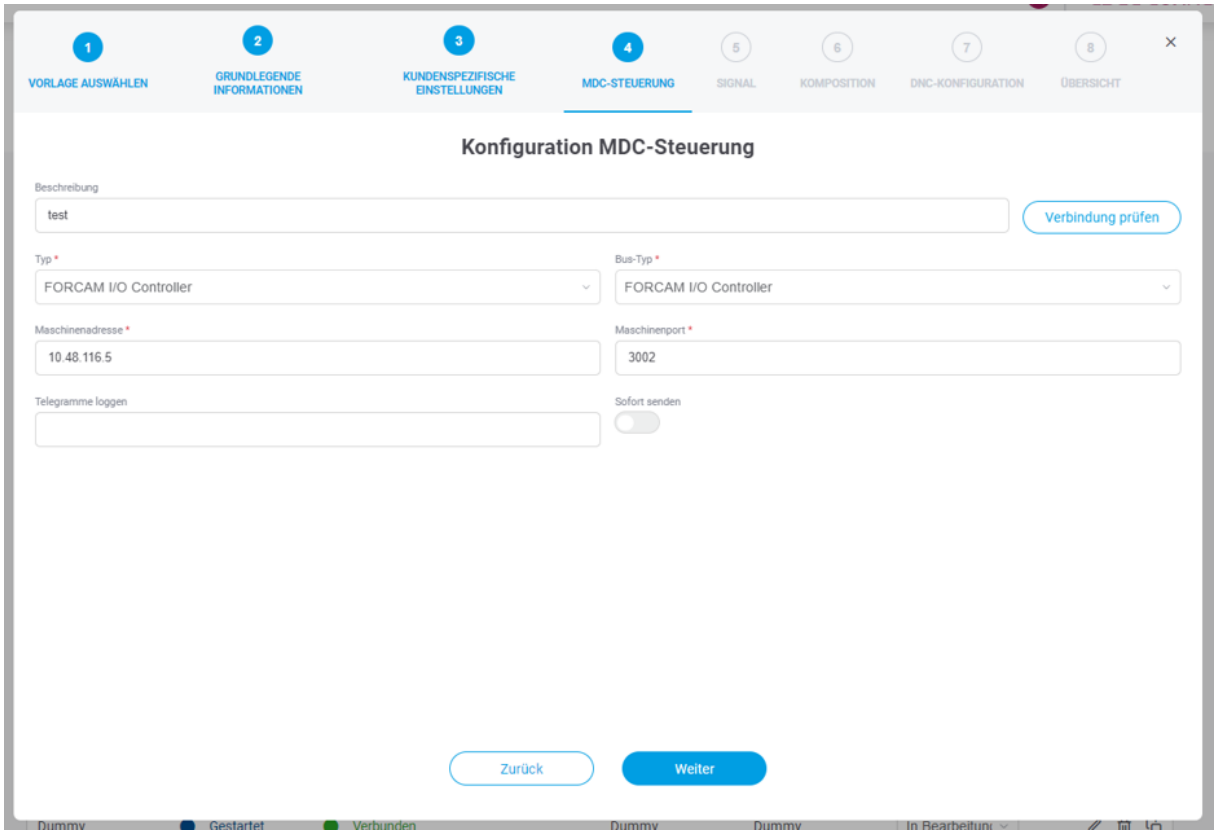


Bild 22: Configuration Wizard – MDC-Steuerung

ⓘ Die Funktion **Telegramme loggen** dient zum Mitloggen der UDP-Telegramme im MDC-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 pro Minute), und zudem können Skalier-Faktoren festgelegt werden. Durch den Skalierungsfaktor ist es beispielsweise möglich, durch den erfassten Widerstand an einem Asset auf die Temperatur zurückzuschließen.

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

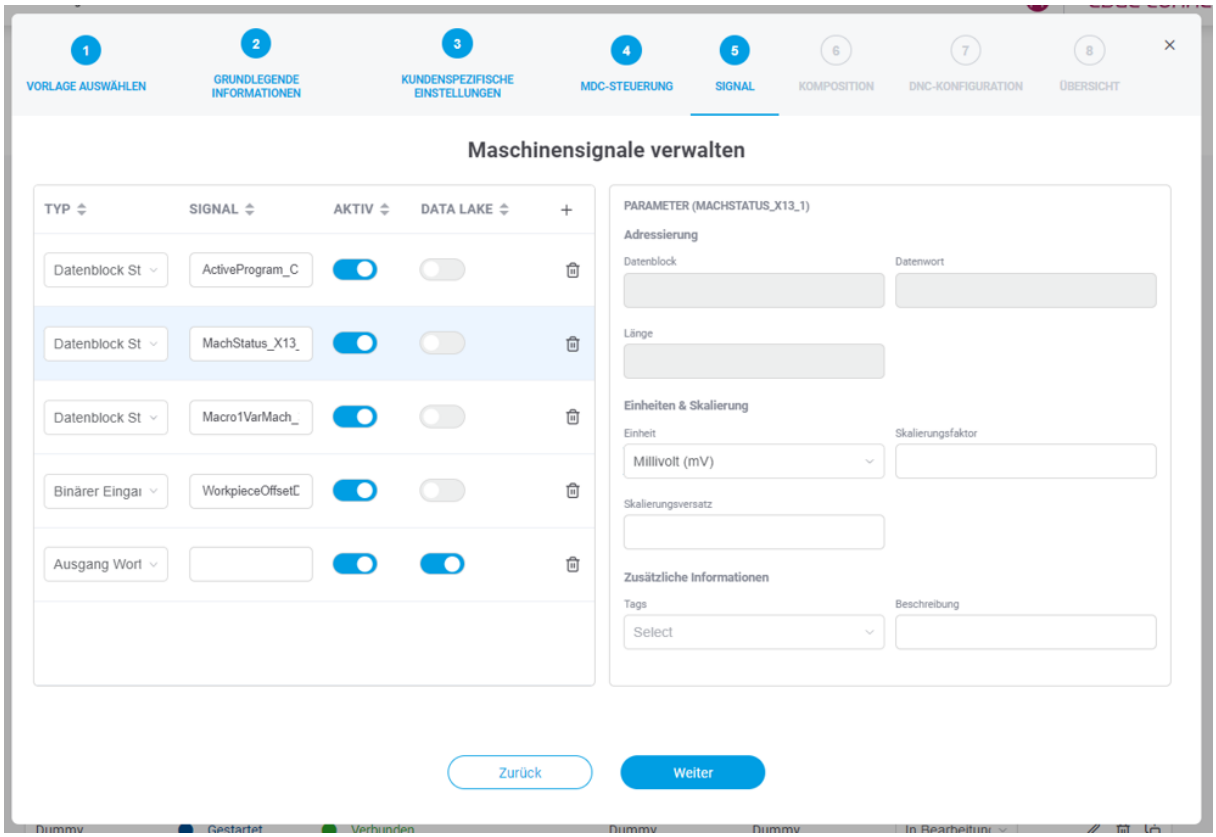


Bild 23: Configuration Wizard – Signal

1. Auf das **+** Icon klicken.
2. **TYP** wählen, Namen für **SIGNAL** eintragen sowie optional den Schalter für **DATA LAKE** aktivieren.
3. Plug-in-spezifische Signal-**Parameter** angeben.
3. Optional**: Einheiten & Skalierung** und **Beschreibung** angeben.
4. Auf **Weiter** klicken.

Eingabefeld	Beschreibung
TYP	Datentyp des Signals, Auswahl unterscheidet sich nach Assetanbindungsvariante
SIGNAL	Name des Signals, freie Eingabe

Eingabefeld	Beschreibung
AKTIV	Auswahl, ob das Signal aktiv erfasst werden soll oder nicht
DATA LAKE	Auswahl, ob Signal im Data Lake gespeichert werden soll oder nicht Option ist nur verfügbar, wenn Data Lake Funktion freigeschaltet ist.
Parameter	**Beschreibung**
Adressierung	Adresse, Eingabeoptionen abhängig vom TYP
Einheit & Skalierung	– Einheit: Auswahl aus Drop-down-Menü – Skalierungsfaktor: Faktor, mit dem ein Wert multipliziert wird, um ihn in seinem Bezugssystem richtig zu interpretieren – Skalierungsversatz: Versatz, der bei der Interpretation von Werten berücksichtigt werden muss
Zusätzliche Informationen	– Tags: Schlagworte, mit denen das Signal angereichert wird – Beschreibung: Freie Eingabe

⚠ Einmal angelegte Signale können nicht mehr verändert werden.

6.3.6 ⑥ Komposition

In diesem Schritt werden die erfassten Signale interpretiert und Bedingungen für die Interpretation definiert. Daraus resultieren beispielsweise Messwerte, Informationen für die Wartung und unterschiedliche Produktionsstatus. Dies ermöglicht logische Schlussfolgerungen zum Verhalten des Assets.

ⓣ Wir empfehlen, unternehmensinterne Richtlinien für die Signalkomposition zu erstellen. Dadurch entsteht ein einheitliches Datenmodell über alle Assets, das die Grundlage für vergleichende Auswertungen bildet.

Die Signalinterpretation kann auf zwei Arten erfolgen: Zum einen wird unter SKRIPT der Text-basierte Code angezeigt und bearbeitet (siehe Bild 25), zum anderen unter **GRAFIK** die grafischen Blöcke (siehe Bild 24).

⚠ Eine gleichzeitige Bearbeitung in **SKRIPT** und **GRAFIK** bzw. ein Wechsel von **SKRIPT** nach **GRAFIK** ist nicht möglich.
Sobald Änderungen im Skript-Editor vorgenommen wurden, muss die weitere Bearbeitung dort erfolgen und der Code wird unter **GRAFIK** nicht mehr angezeigt.

Grafischer Editor

Die Blöcke des grafischen Editors sind Programmier-Bausteine, die ähnlich wie Puzzleteile zusammengesetzt werden können. Vorteil dieses Baukastensystems ist, dass auch Einsteiger ohne Programmiererfahrung Befehle erstellen können.

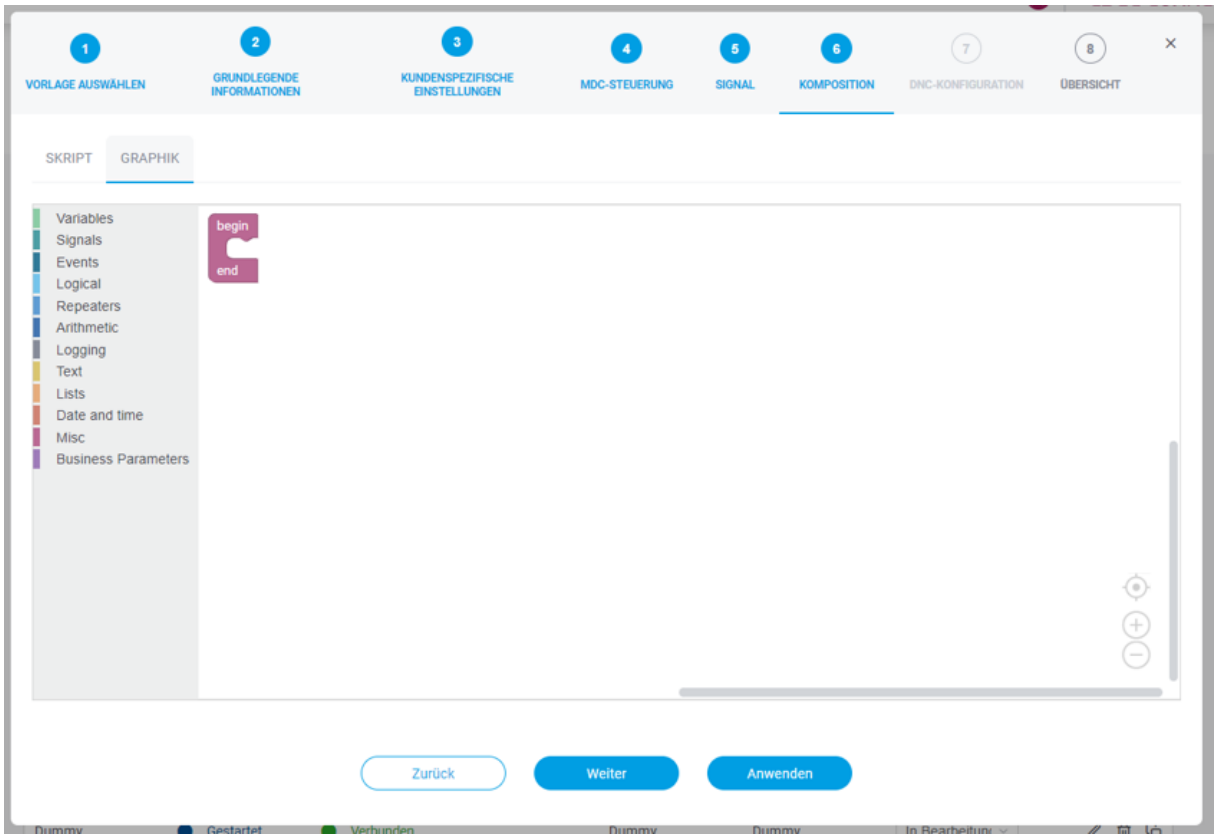


Bild 24: Grafischer Editor

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.

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

Skript-Editor

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

⚠ In **SKRIPT** sollte nur arbeiten, wer Vorkenntnisse im Programmieren hat.

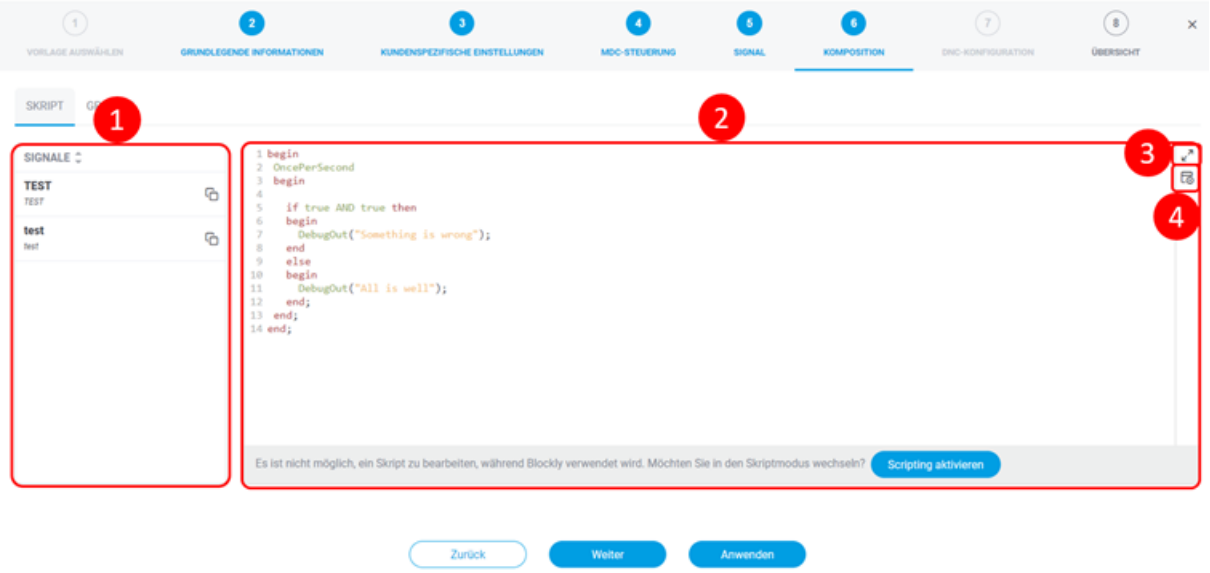


Bild 25: Skript-Editor

(1) Zeigt die im Schritt 5 im Configuration Wizard hinzugefügten Signale an
(2) Bearbeitungsbereich mit aktuellem Skript
(3) Ansicht vergrößern/verkleinern (Vollbild)
(4) Aktuelles Skript auf Gültigkeit prüfen

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

6.3.7 ⑦ DNC-Konfiguration

Dieser Schritt bietet die Möglichkeit zur Konfiguration einer DNC-Steuerung. Sie bestimmt die Art und Weise, wie eine NC-Datei an das Asset übertragen werden soll.
Eine Übersicht der aktuellen FORCAM Plug-ins ist in Abschnitt 0 aufgelistet.

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

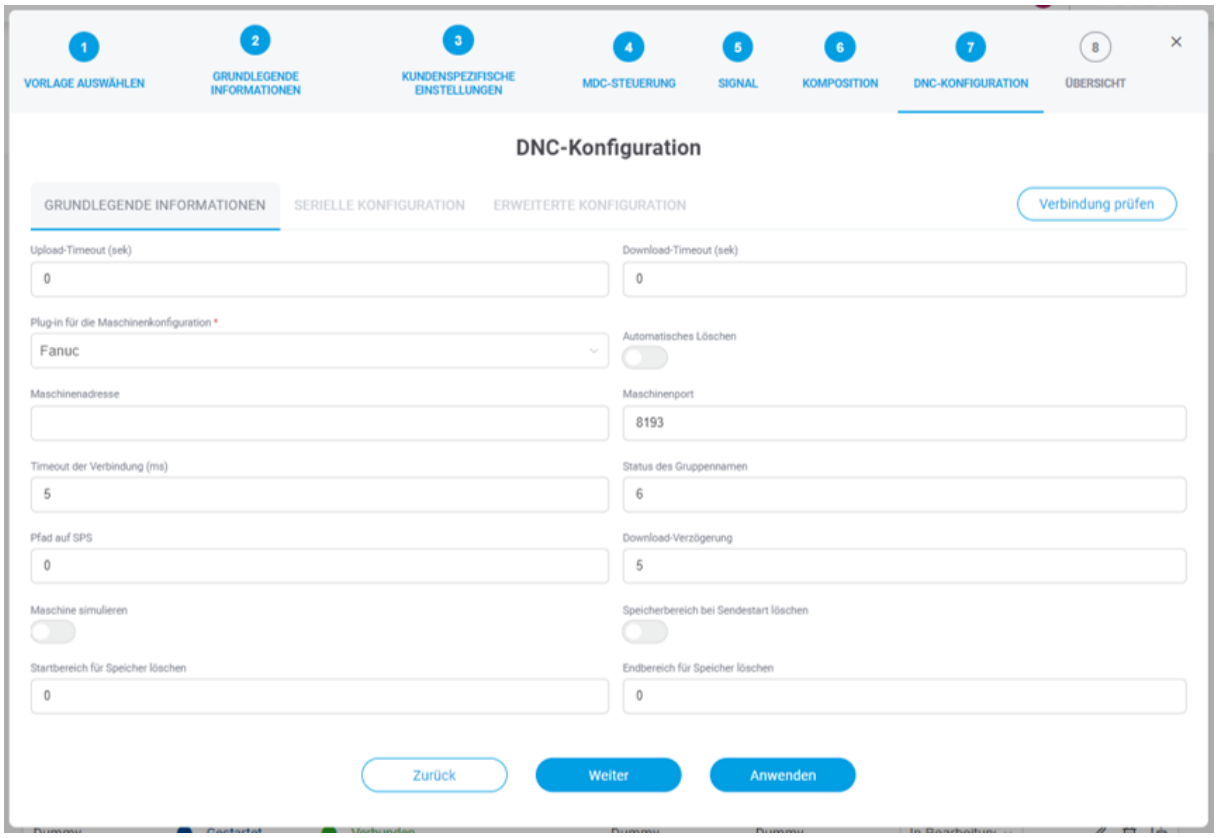


Bild 26: Configuration Wizard – DNC-Konfiguration

Eingabefeld	Beschreibung
Upload-Timeout (sek)	Angabe in Sekunden
Download-Timeout (sek)	Angabe in Sekunden
Plug-in für die Maschinenkonfiguration	Drop-down-Menü aller verfügbaren Plug-ins Je nach Auswahl, werden unten weitere Eingabefelder eingeblendet. Pflichtfelder sind mit einem Stern (*) gekennzeichnet.
Automatisches Löschen	Schalter aktiviert = NC-Datei wird nach dem Lesen automatisch aus dem Asset gelöscht.

ⓘ Nach der DNC-Konfiguration kann die Verbindung überprüft werden.

6.3.8 ⑧ Übersicht

Dieser Schritt 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 15).

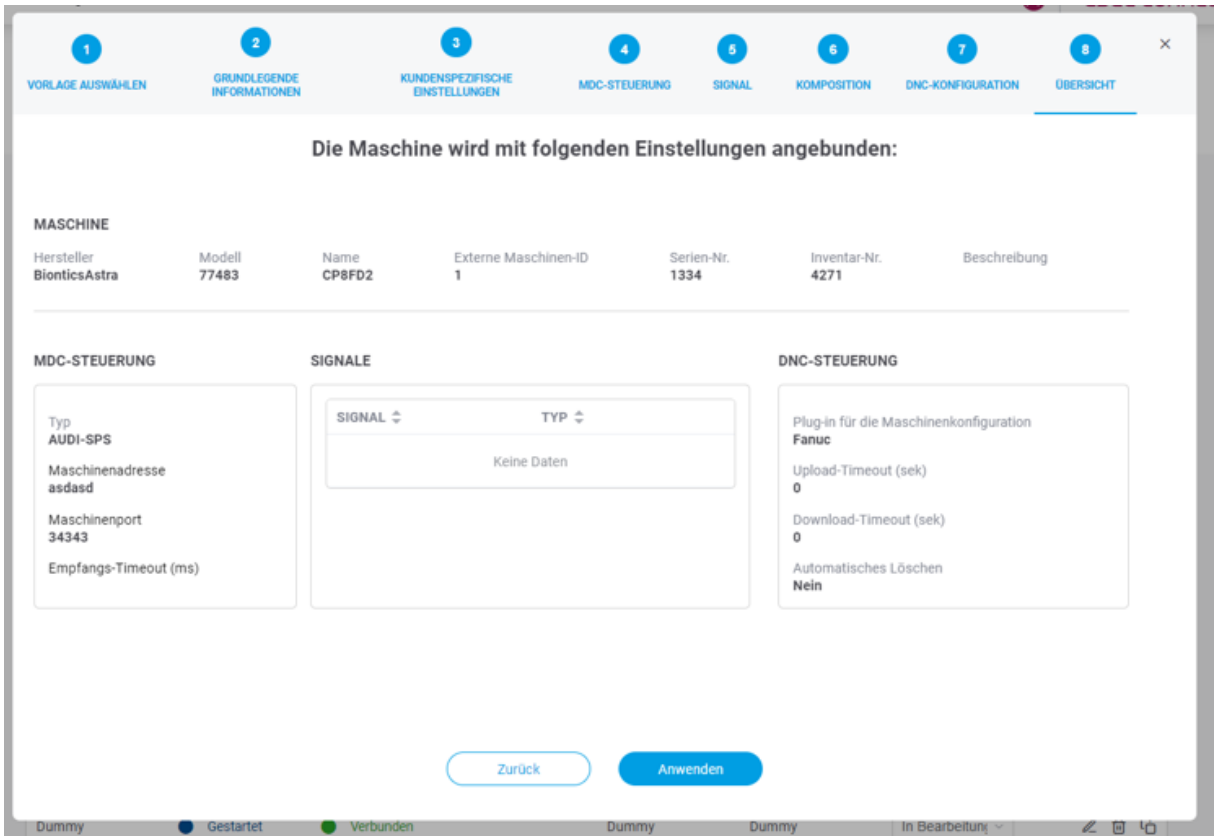


Bild 27: Configuration Wizard – Übersicht

6.4 Northbound Konfiguration

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

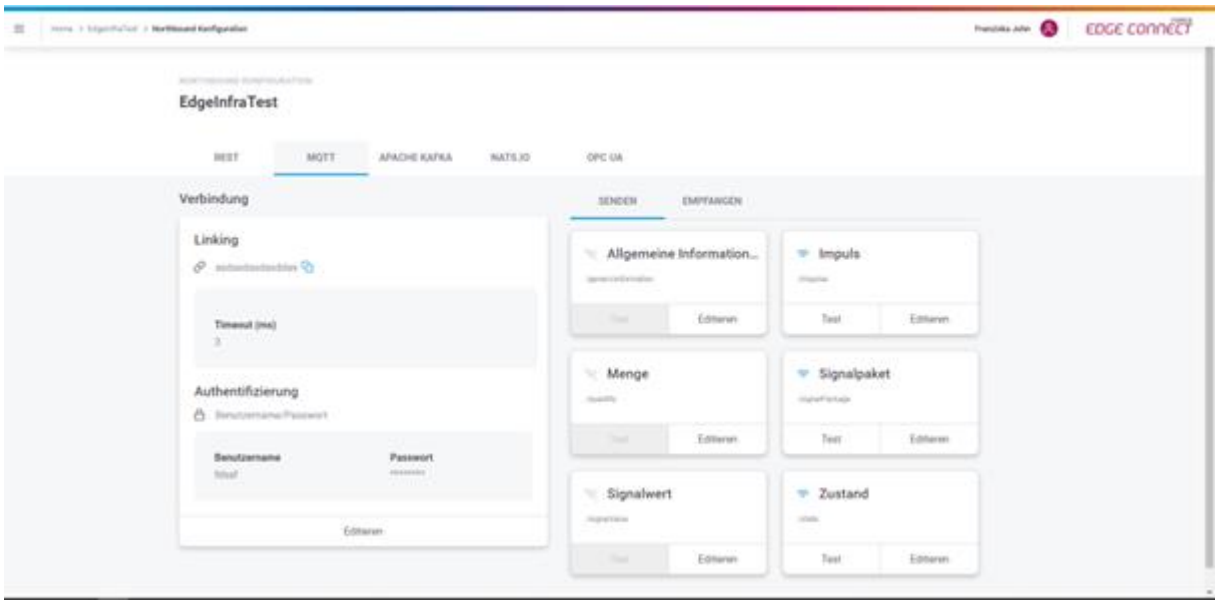


Bild 28: Northbound Konfiguration in EDGE CONNECT

Events sind ausgehende Ereignisse, die über das Skript erzeugt werden. 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: $unit$
}
```

Mithilfe von **Platzhaltern** (Wildcards) können Strukturen von Events bearbeitet werden , mit deren Hilfe unterschiedliche Informationen weitergegeben werden. Hierüber kann beispielsweise die Maschinen-ID oder der Zeitstempel in UTC umgerechnet 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 getestet 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 und 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.

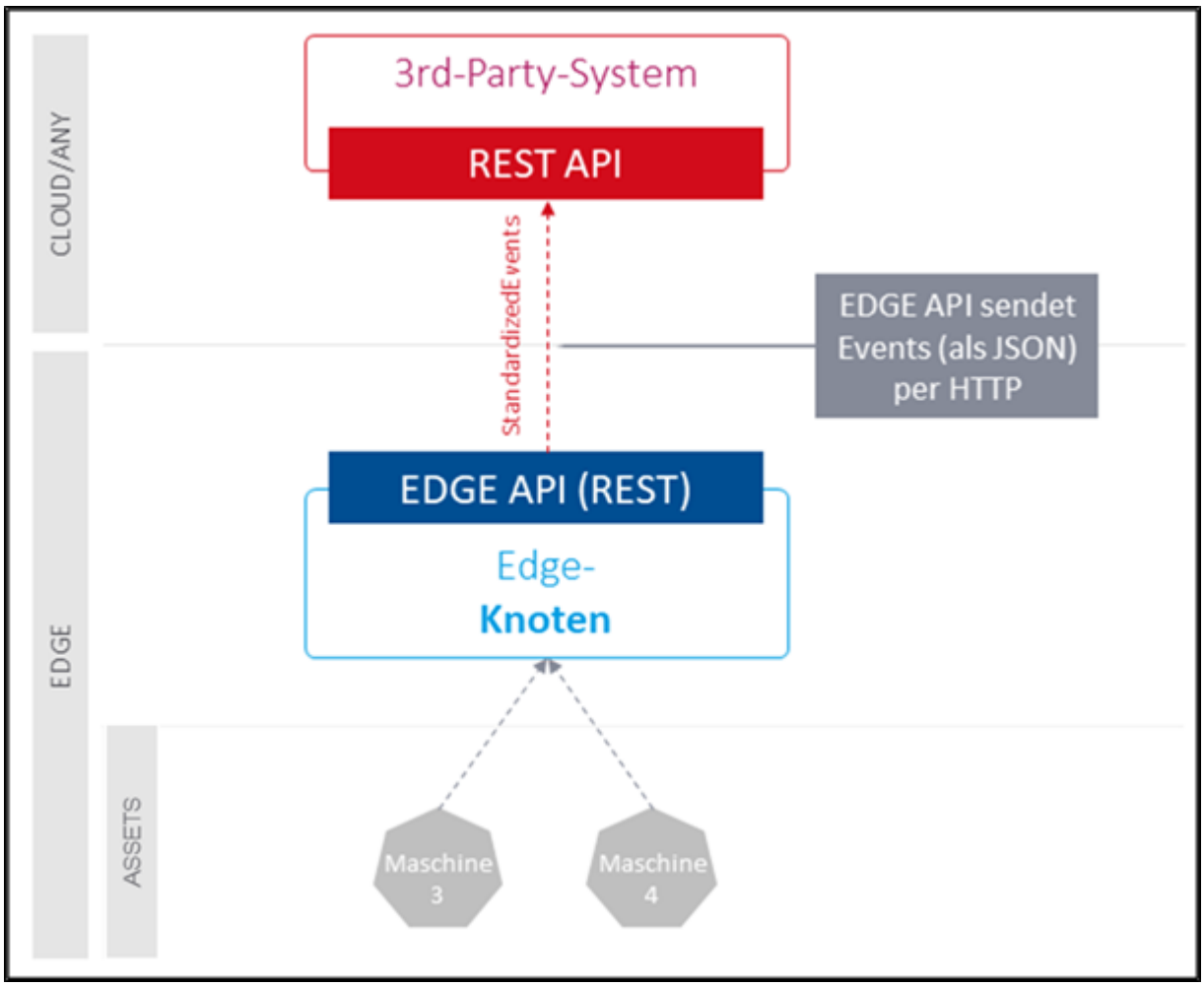


Bild 29: Kommunikation mit übergeordneten Systemen über HTTP/REST

MQTT-Messaging

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

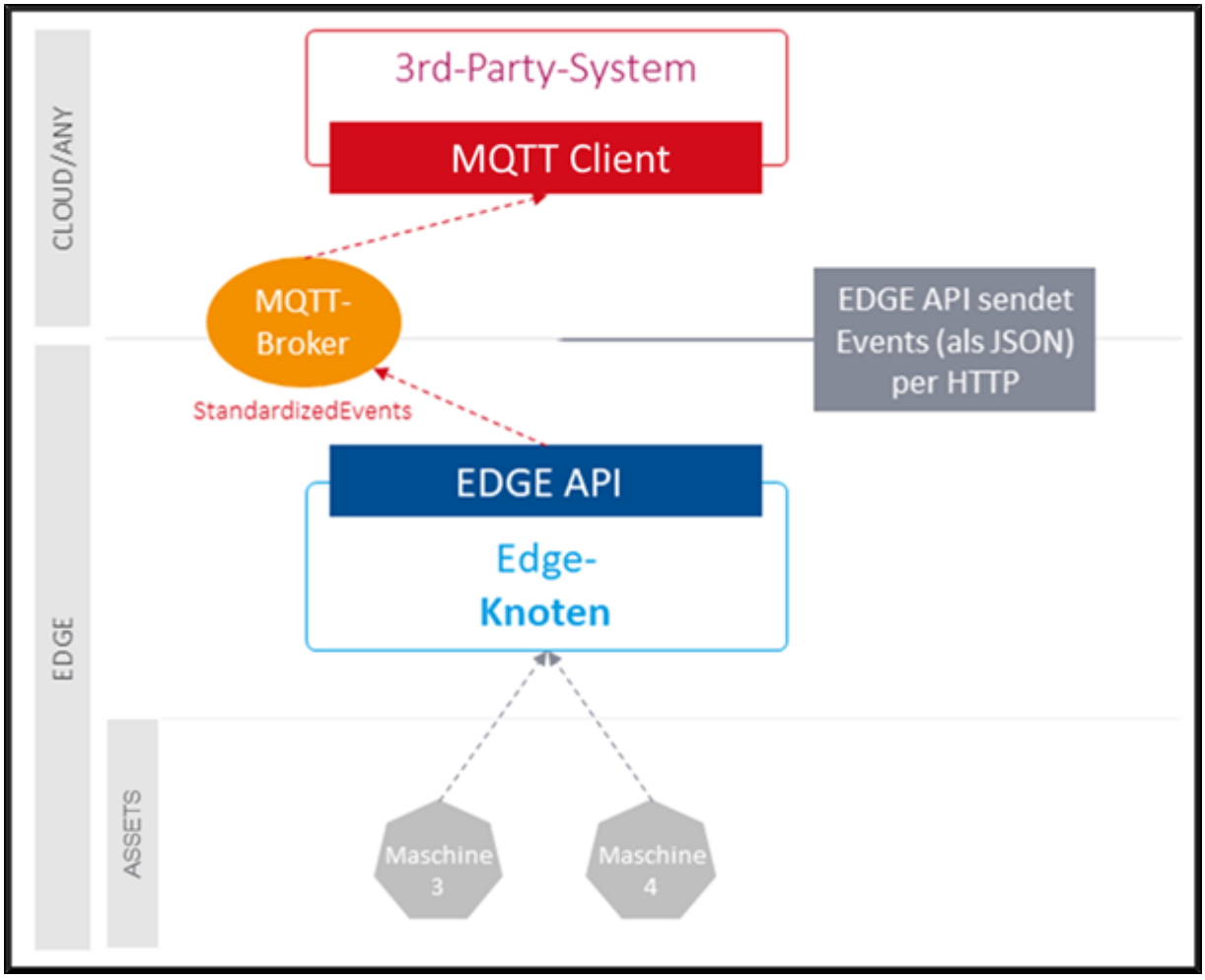


Bild 30: Kommunikation mit übergeordneten Systemen über MQTT-Broker

Apache Kafka

Die Versorgung des Drittsystems kann durch die Unterstützung von Apache Kafka erfolgen, sofern durch Kunden oder Partner bereitgestellt.

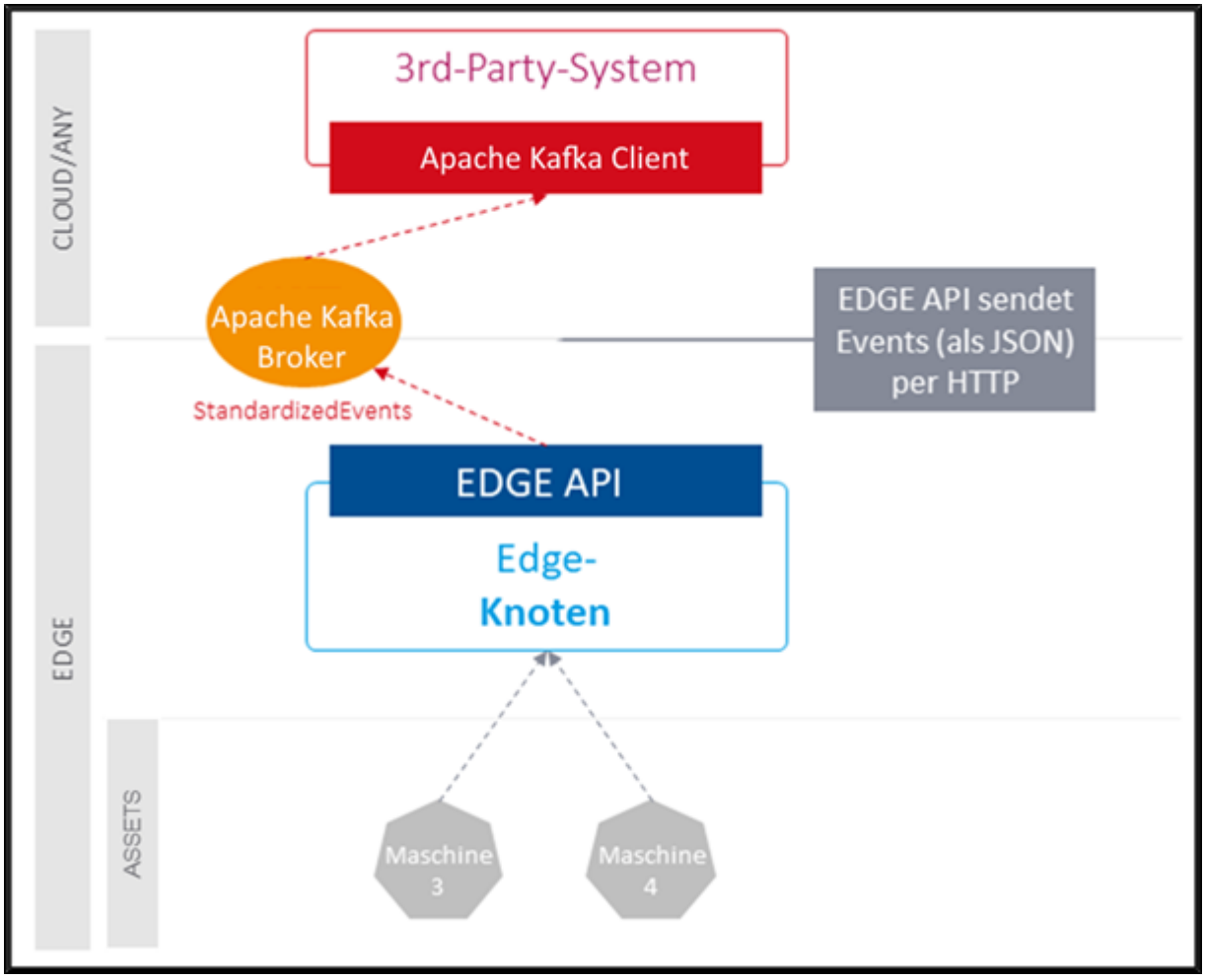


Bild 31: Kommunikation mit übergeordneten Systemen über Apache Kafka

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 kann sich der Benutzer mit dem Server verbinden, 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-Funktion (nur mit Data Lake verfügbar) 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.

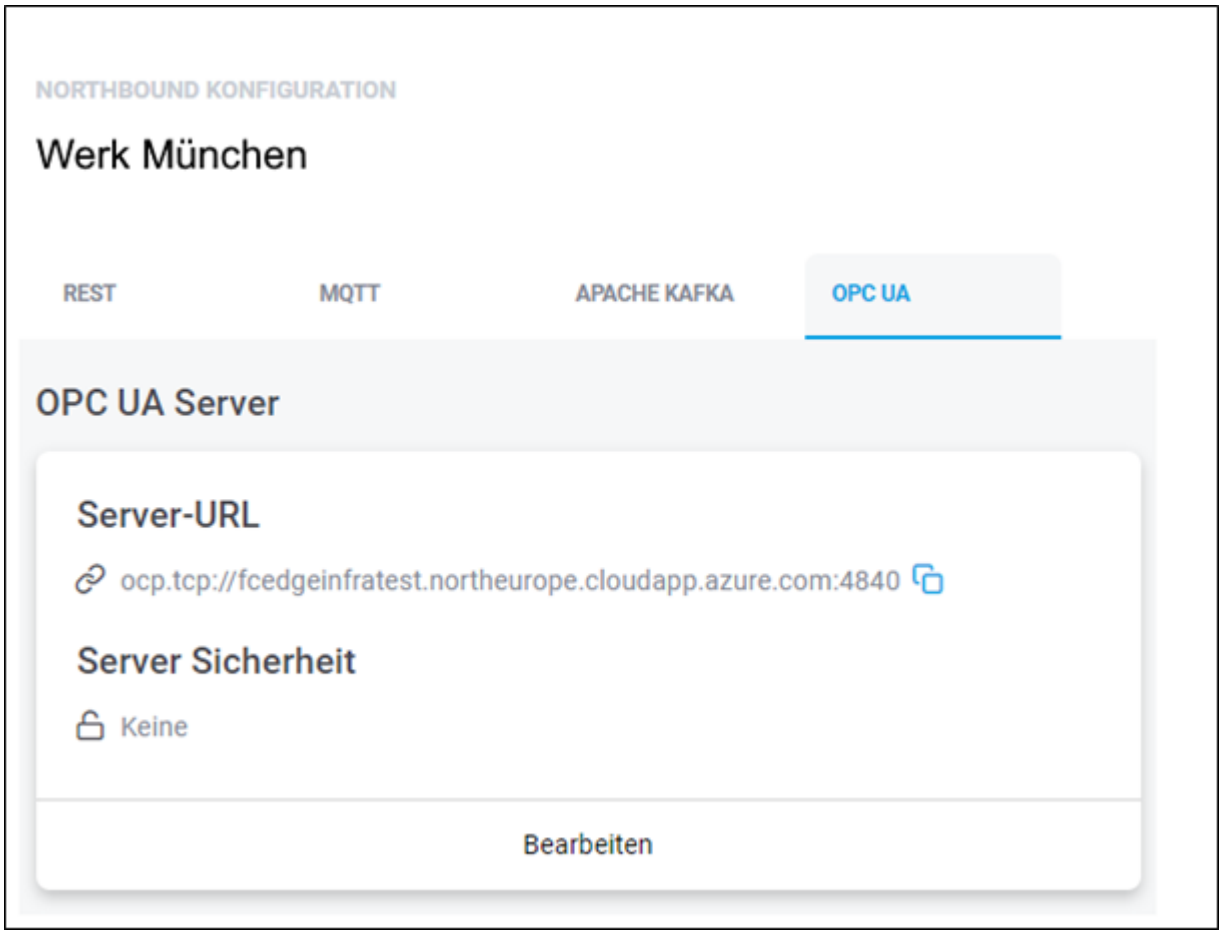


Bild 32: Eventkonfiguration über OPC UA

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 Parametern) wird durch die NATS-Schnittstelle nicht unterstützt.

Für die Übertragung der Events wird sowohl Core NATS, als auch NATS 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.

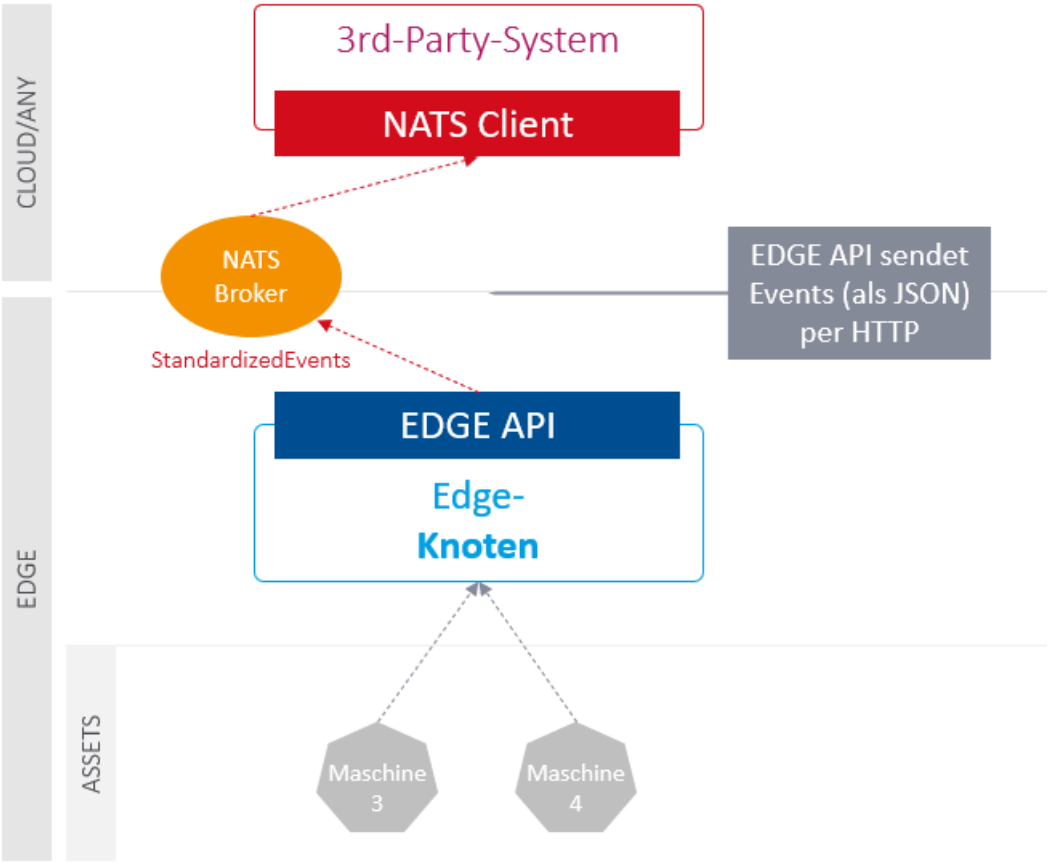


Bild 33: Kommunikation mit übergeordneten Systemen über NATS.io

6.4.2 Daten und Dokumente vom übergeordneten System zu EDGE

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

Funktion	Beschreibung
Übertragen von Prozess- und Referenzparametern	Diese Business Parameter können in der Signal Composition verwendet werden, um damit u.a. 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 auf das Asset übertragen werden.

Tabelle 2: Schnittstellen für die Übertragung von Daten und Dokumenten

6.4.3 Event konfigurieren

Bevor Events erstellt werden, muss zuerst die Verbindung bestimmt werden, über die mit dem Asset kommuniziert wird (Schnittstelle zwischen Asset und Drittsystem).
Anschließend können, je nach Schnittstelle, standardisierte Events erzeugt werden.

Um die Verbindung zu bestimmen:

1. In der Detailansicht eines konfigurierten Assets (siehe Bild 15) im rechten oberen Bereich auf das Northbound Konfiguration-Icon klicken.
2. Im Folgedialog Reiter wählen, welche Schnittstelle verwendet werden soll.
   - REST
   - MQTT
   - Apache Kafka
   - NATS.io
   - OPC UA
3. Unter Verbindung auf **Editieren** klicken.
4. Im Folgedialog Verbindungsinformationen eintragen.
   (Für notwendigen Informationen je Schnittstelle: siehe Tabelle 3)
5. Nur bei REST und Apache Kafka:
   Schalter **Überprüfe SSL-Zertifikat** aktivieren, wenn über ein Standard-Sicherheitszertifikat kommuniziert wird.
   ⓘ Schalter deaktivieren, wenn mit einer Anwendung ohne gültiges oder mit einem unsignierten Sicherheitszertifikat verbunden wird (z.B. bei selbst erstellten Zertifikaten).
6. Gewünschte Authentifizierung auswählen und Anmeldedaten eintragen.
7. **Speichern**.

Konfiguration der Schnittstellen

Konfiguration	Beschreibung
REST	
URL*	Textfeld für die URL des Servers
Inhaltstyp	Datenformat des Nachrichteninhalts
Anzahl der Verbindungen*	Anzahl der aktiven, parallelen Verbindungen zum externen System
Timeout (ms)*	Begrenzt die Dauer der Verbindungsversuche zum Server
Überprüfe SSL Zertifikat	Überprüft das SSL-Zertifikat automatisch Bei selbst erstelltem Zertifikat sollte der Schalter deaktiviert werden.
Authentifizierung	Keine: Keine Authentifizierung HTTP BasicAuth: Benutzer / Passwort Auth 2.0 Client credential flow: Authentifizierung über definierten Standard
MQTT	
URL*	Textfeld für die URL des Servers
Timeout (ms)	Begrenzt die Dauer der Verbindungsversuche zum Server
Authentifizierung	Keine: Keine Authentifizierung Benutzername/Passwort: Authentifizierung über User und Passwort
Benutzer*	Textfeld für den Benutzer
Passwort	Textfeld für das Passwort
Apache Kafka	
Bootstrap Servers*	Textfeld für die URL des Servers
Client ID	
Überprüfe SSL Zertifikat	Überprüft das SSL Zertifikat automatisch Bei selbst erstelltem Zertifikat sollte der Schalter deaktiviert werden.
Security Protocol	Plain Text SASL Plain Text SASL SSL
NATS.io	
Server URL*	Textfeld für die URL des Servers
Timeout (ms)*	Begrenzt die Dauer der Verbindungsversuche zum Server
Neuer Verbindungsversuch in (ms)	Wartezeit für einen erneuten Verbindungsversuch zum Server

Konfiguration	Beschreibung
TLS aktivieren	Aktiviert die Verschlüsselung über SSL/TLS
Server Sicherheit	Keine: Keine Authentifizierung Benutzername/Passwort: Authentifizierung über Benutzername und Passwort JWT/NKey Anmeldeinformationen: Authentifizierung mit Zertifikatsdatei
Streamingverhalten	Core NATS: Gesendete Nachrichten werden nicht gespeichert. JetStream: Gesendete Nachrichten werden gespeichert. Dabei gilt, dass die Nachricht mindestens einmal übermittelt wird (Quality of Service Level 1).
OPC UA	
Serversicherheit	Keine: Kein Benutzerpasswort nötig Anmeldeinformationen: Benutzername / Passwort nötig

*Pflichtfeld

Tabelle 3: Schnittstellenkonfiguration der Northbound Konfiguration

Um ein Event zu konfigurieren:

ⓘ Je nach gewählter Schnittstelle lassen sich bestimmte Evens konfigurieren. Die Dialogfenster sehen daher etwas anders aus.

1. Unter Event auf **Editieren** klicken.
2. Im Dialogfenster Events wie gewünscht konfigurieren.
   Platzhalter auf der linken Seite kopieren und in den Bereich rechts einfügen.
   Oder
   Direkt in den rechten Bereich schreiben.
   ⓘ Skriptsprache kann zwischen JSON, XML oder Text gewechselt werden (siehe Bild 33)
3. Mit dem Schalter **Aktiv** das Event aktivieren.
   Ein aktives Event ist in der Übersicht der Northbound Configuration durch ein blaues Wellen-Icon gekennzeichnet. Bei inaktiven Events ist das Wellen-Icon grau und durchgestrichen.
4. **Speichern**.

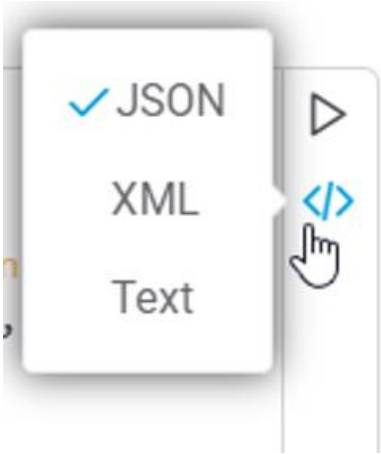


Bild 34: Skriptsprache ändern

6.5 Integration

Folgende API-Schnittstellen stehen für EDGE CONNECT zur Verfügung:

EDGE Node	
EDGE Node API	IP-Adresse + 60067/api/management
Data Lake API	IP-Adresse + 60067/api/data-lake
EDGE Configuration	
EDGE Configuration API	IP-Adresse + 60066/api/configuration
Monitoring API	IP-Adresse + 60066/api/monitoring
Literals	IP-Adresse + 60066/api/literals

Authentifizierung in Swagger

Authentifizierung für EDGE Node API und Data Lake API:

1. Unter **Basic Authentication (http, Basic)** folgende Eingaben machen:
   - **Username**: *usertest@mail.com*
   - **Password**: *secret* (API Key wird während der Installation vergeben)
2. **Authorize** klicken.

Authentifizierung für EDGE Configuration:

3. Unter **Basic Authentication (http, Basic)** den **Username** und das **Password** des jeweiligen Benutzers eingeben.
4. **Authorize** klicken.

7 Monitoring

EDGE CONNECT bietet die Option, die verschiedenen Komponenten der Software 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 Monitoring-Icon im rechten oberen Bereich in der Asset-Übersicht aufrufbar (siehe Bild 15).

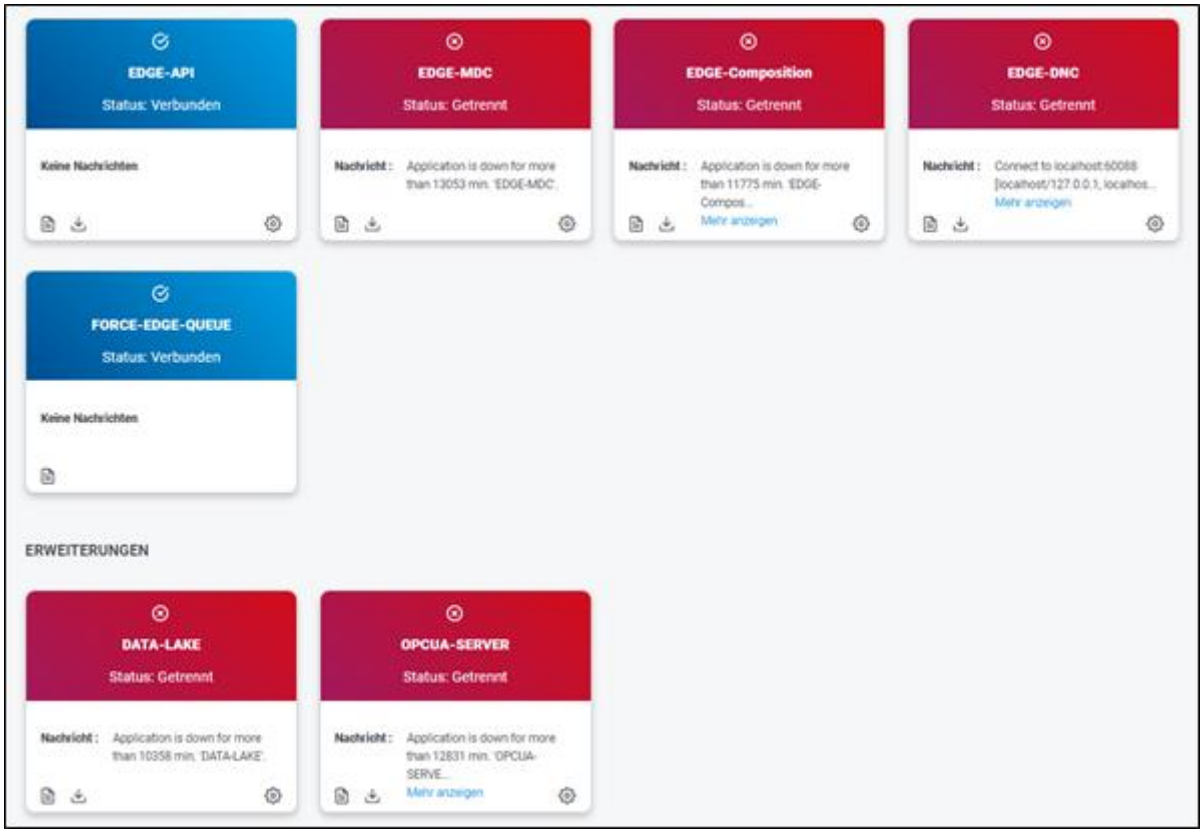


Bild 34: Monitoring in EDGE CONNECT

In jeder Komponente können Fehlermeldungen und Logs gezielt abgerufen werden.

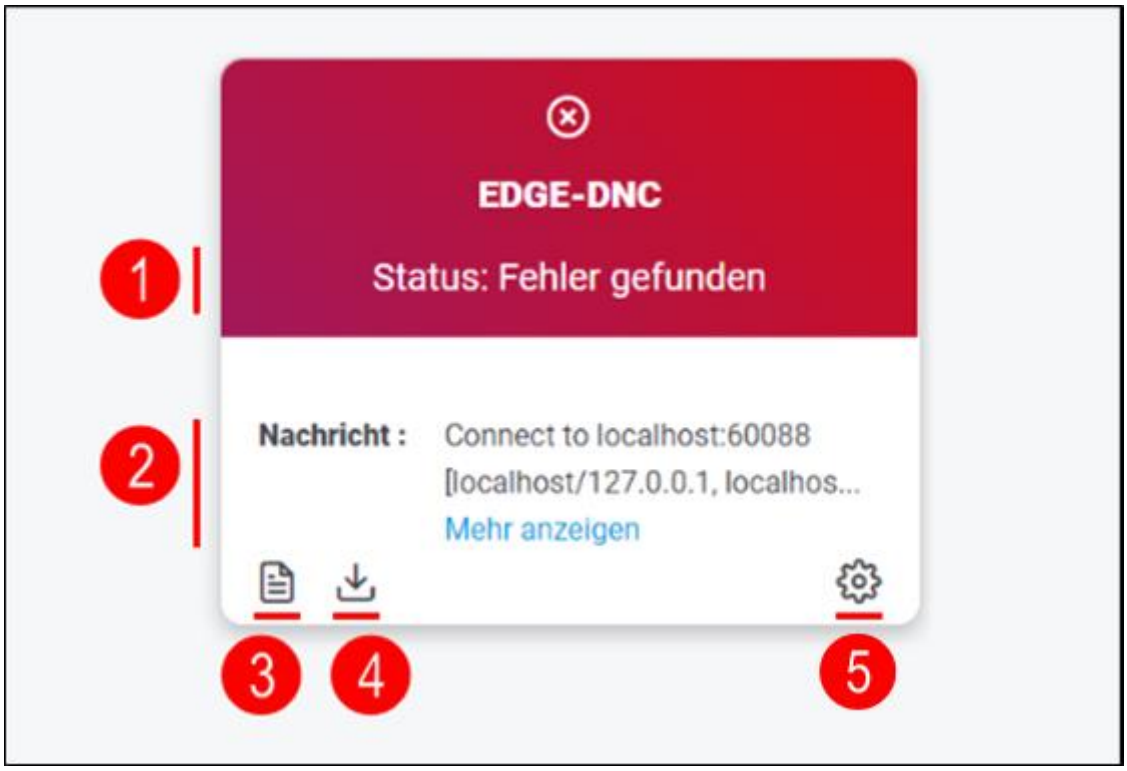


Bild 34: Komponente „EDGE-DNC“ in der Monitoring-Seite

(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 eines bestimmten Tags
(5) Einstellung des Log-Levels

Log-Level

Log-Level	Beschreibung
Trace	Alle Informationen aus dem System werden in der Logdatei gespeichert. ⚠ Erzeugt sehr große Datenmengen.
Debug	Logdatei speichert zusätzlich zu regulären Informationen (siehe Log-Level: Info) weitere Informationen aus dem System
Info	Logdatei beinhaltet Inforationen zum Status des Assets, Verbindungsinformationen usw.
Warnung	Nur Warnungen werden in der Logdatei gespeichert
Fehler	Nur Fehlermeldungen werden in der Logdatei gespeichert

8 Anhang

8.1 Dokument-Konventionen

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.

Tabelle 4: Verwendete Schriftarten, Formatierungen und Zeichen

8.2 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.
MDC	Machine Data Connection (Maschinendatenerfassung)

Abkürzung	Erklärung
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 - bezeichnet Hardware und Software, welche die Leistung physischer Geräte überwacht und steuert.
Plug-in	FORCAM verwendet Plug-ins als vereinfachte Anbindungen an Steuerungen.
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
TLS	Transport Layer Security Verschlüsselungsprotokoll für die Transportschicht des Internets. Die Datenströme zwischen Client und Server werden verschlüsselt. TLS ist das Nachfolgeprogramm von SSL.
UDP	User Datagram Protocol – ist ein minimales, verbindungsloses Netzwerkprotokoll, das zur Transportschicht der Internetprotokollfamilie gehört. UDP ermöglicht Anwendungen den Versand von Datagrammen in IP-basierten Rechnernetzen.
UTC	Coordinated Universal Time (koordinierte Weltzeit)
Wildcard	Platzhalter für andere Zeichen

Tabelle 5: Verwendete Abkürzungen und Begriffe

8.3 Liste unterstützter Plug-ins

MDC-Plug-ins

Name	Lesen	Schreiben	Übertragungsart Polling/Eventbasiert
AUDI SPS	X	X	X/
Controller for FORCAM FORCE DB tables	X		X/
CSV File Exchange	X		X/
Euromap 63	X		X/
Euromap 77 (via OPC UA)	X	X	/X
FANUC	X	X	X/
FORCAM IO Controller	X	X	/X
FORCAM I/O Controller (Hardware)	X		
MAKINO Pro 3/Pro 6	X		X/
MAZAK Mazatol Fusion M640M	X	X	/X
MAZAK Mazatol Fusion M640MTPro	X	X	/X
MAZAK Mazatol Matrix	X	X	/X
MAZAK Mazatol Smart	X	X	/X
MAZAK Mazatol Smooth	X	X	/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		X/
OMRON CS/CJ	X	X	X/
OMRON CV	X	X	X/

Name	Lesen	Schreiben	Übertragungsart Polling/Eventbasiert
OPC Classic	X	X	X/
OPC XML	X		X/
OPC UA	X	X	/X
Rockwell / Allen Bradley	X	X	X/
RPC	X	X	/X
Schneider Electric	X		X/
Siemens LOGO	X		X/
Siemens S5	X		X/
Siemens S7 (200, 300, 400, 1200, 1500)	X	X	X/
SQL Database Exchange	X		X/
Weihenstephan	X		X/
Wiesemann & Theis (WUT)	X		X/
Windows RPC	X	X	/X

Tabelle 6: Liste aller unterstützter Maschinenanbindungsvarianten

DNC-Plug-ins

Name	Lesen	Schreiben
COM	X	X
External program file transfer (Preview version)	X	X
FANUC	X	X
File Handler (File Copy)	X	X
File Handler Server	X	X
FTP Plug-in	X	X
Heidenhain	X	X
Mazak	X	X
MOXA-Box	X	X
RPC (Preview version)	X	X
WUT (Preview version)	X	X

Tabelle 7: Liste aller unterstützter NC-Maschinen-Anbindungsvarianten

8.4 Standardisierte Events

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 – External Machine Id	Zustand der Maschine (läuft, läuft nicht)

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

Tabelle 8: Events und deren Funktion in EDGE CONNECT

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.	-