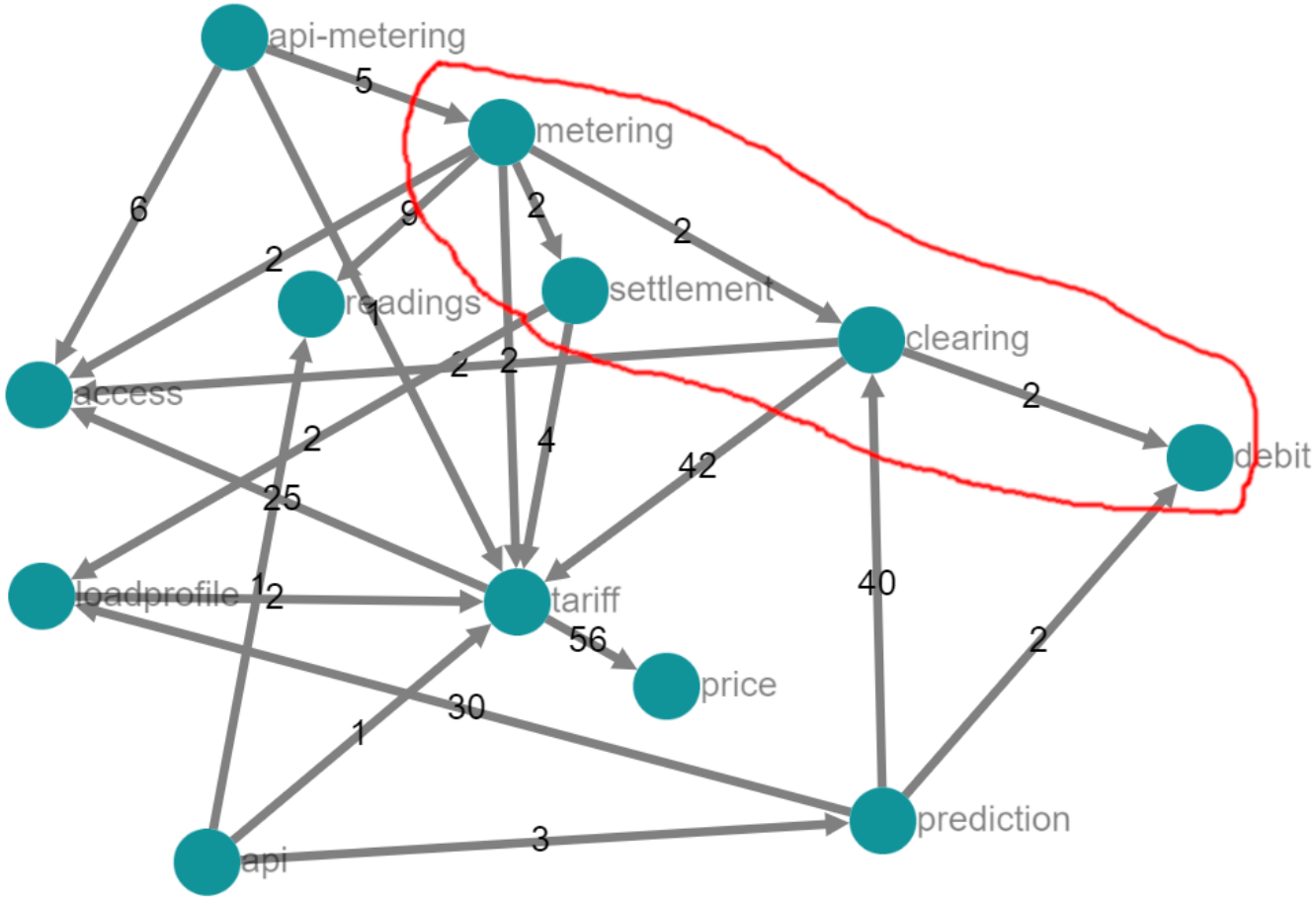
Energy Application Framework [DE]

Das STROMDAO EAF ist eine Open-Source Referenzimplementierung für digitale Energie Infrastruktur. Das Framework kann genutzt werden, um zum Beispiel dynamische Stromtarife als Produkt anzubieten.

- Dynamische Stromtarife - Von der Zählerablesung bis zur Stromrechnung im STROMDAO EAF
- MQTT als Standard für den Austausch von Energiedaten
- Digitale Dienste aus der Nachbarschaft
- EAF - App für dynamische Stromtarife
- Lastprofil für variable Stromtarife
- Einführung - Energieanwendungen
    - Grundlagen
    - Komponenten von Energieanwendungen
    - Integration von Energieanwendungen
    - Beispiele für Energie Anwendungen
- Einführung dynamischer Stromtarife
    - Rahmenbedingungen für dynamische Stromtarife
    - Entwicklung von dynamischen Tarifmodellen
- EDIFACT-Nachrichten in der Energiewirtschaft: Entwickler-Einführung

Dynamische Stromtarife - Von der Zählerablesung bis zur Stromrechnung im STROMDAO EAF



Der Prozess von der präzisen Erfassung des Stromverbrauchs bis hin zur Abrechnung bei dynamischen Stromtarifen ist automatisiert und effektiv durch das STROMDAO Energy Application Framework optimiert. Im Fokus steht dabei eine innovative Technologie, die Leistungsfähigkeit und Benutzerfreundlichkeit vereint.

Zählerstandserfassung und Dateneinreichung im STROMDAO EAF

Die Erfassung des Zählerstands erfolgt automatisiert durch SmartMeterGateways oder Leseköpfe mit hoher Frequenz. Dies kann von täglich bis hin zu jeder Sekunde reichen – ein üblicher Wert liegt bei einer Ablesung alle 15 Minuten. Die STROMDAO EAF Metering API ist standardisiert und kompatibel mit einer Vielzahl von Clients, die eine einfache Anbindung an Heimautomatisierungssysteme oder Energie Management Systeme ermöglichen.

Dank des implementierten `metering` Microservices können Anomalien wie ungenaue Messungen oder fehlerhafte Datenübermittlungen intelligent ausgeglichen werden. Sobald ein Zählerstand erfasst wird, werden die Informationen wie Zählerstand (`reading`) und Zählerkennung (`meterId`) über die `updateReading` Funktion des Microservices nahtlos in das System eingeführt.

Verbrauchsanalyse und Settlement

Die vorliegenden Zählerdaten durchlaufen eine sorgfältige Analyse. Existiert bereits eine vorherige Ablesung, so berechnet der `metering` Service das Verbrauchsdelta. Für die Zuweisung des Verbrauchs zu geeigneten Tarifsegmenten sorgt der `settlement` Service, der dafür die Preisinformationen aus der Zeit vor der Ablesung nutzt. STROMDAO EAF greift dabei automatisch auf die benötigten Preisdaten zu.

Das Settlement ist ebenfalls auf die Verarbeitung digitaler Identitäten ausgelegt, sodass Verbrauchsdaten auf Basis von durch Dritte berechtigte Preisinformationen verrechnet werden können, selbst wenn diese nicht im lokalen Datenbestand von STROMDAO EAF vorhanden sind.

Clearing-Prozess

Dieser Abschnitt umfasst mehrere wichtige Prozessschritte:

1. **Kostenberechnung und Zuordnung zum Kundenkonto:**
   Der `clearing` Service leistet die Umwandlung von Verbrauch und Tarifinformationen in konkrete Kosten.
2. **Konsensfindung und digitale Signatur**:
   Es entsteht ein digital unterzeichneter Konsensus, der als unwiderrufliche Grundlage für die Abrechnung dient.
3. **Buchung und Rechnungsstellung:**
   Die ermittelten Kosten werden durch den `clearing` Service festgehalten und für die Abrechnung vorbereitet. Abschließend erfolgt die Gutschrift auf einem Kundenkonto, welches alle diesbezüglichen Transaktionen kumuliert und für die Rechnungserstellung bereitsteht.

4. **Ausfertigung der Rechnung:**
   Aus den gesammelten Daten generiert das EAF eine vollständige und transparente Stromrechnung für den Kunden.

Jeder dieser Schritte basiert auf präzisen Messdaten und algorithmisch festgelegten Tarifen, was eine korrekte und nachvollziehbare Abrechnung sicherstellt.

Schlussbemerkung

STROMDAO EAF bildet den gesamten Prozess von der Ablesung der Zähler bis hin zur Ausstellung von Stromrechnungen ab und vereinfacht so die Bereitstellung dynamischer Stromtarife für Energieversorger. Nicht zuletzt hebt das Framework die Effizienz und Transparenz durch den Einsatz von Open-Source-Komponenten und den Fokus auf Standards im Rahmen der Digitalisierung des Energiesektors hervor.

MQTT als Standard für den Austausch von Energiedaten

MQTT ist ein weit verbreiteter Standard für den Austausch von Energiedaten. Er wird von vielen Energiemanagement-Systemen und Smart-Home-Lösungen unterstützt. Dies macht es für Stadtwerke einfach, ihre Systeme mit den Systemen ihrer Kunden zu verbinden und Energiedaten in Echtzeit auszutauschen.

Vorteile für den Energieversorger

Ein MQTT-Broker bietet Energieversorgern eine Reihe von Vorteilen, darunter:

- **Echtzeitdaten zum Strombedarf**
  Mit einem MQTT-Broker können Energieversorger den Strombedarf ihrer Kunden in Echtzeit verfolgen. Dies ermöglicht es ihnen, ihre Stromerzeugung und -verteilung besser zu planen und zu optimieren.
- **Verbesserte Kundenbetreuung**
  Mit einem MQTT-Broker können Energieversorger ihren Kunden einen besseren Service bieten. Sie können beispielsweise ihren Kunden helfen, ihren Stromverbrauch zu optimieren und Geld zu sparen.
- **Neue Geschäftsmodelle**
  Mit einem MQTT-Broker können Energieversorger neue Geschäftsmodelle entwickeln, wie z.B. Energy As A Service (EaaS).

Vorteile für die Stromkunden

Stromkunden können von einem MQTT-Broker beim Energieversorger eine Reihe von Vorteilen erwarten, darunter:

- **Zentraler Ansprechpartner für alles rund um Strom**
  Mit einem MQTT-Broker haben Stromkunden einen zentralen Ansprechpartner für alle Fragen rund um ihren Stromverbrauch. Sie müssen sich nicht mehr an verschiedene Anbieter wenden, um Informationen zu ihrem Stromverbrauch, ihren Stromtarifen oder ihren Stromprodukten zu erhalten.
- **Kein großer Cloud-Anbieter**
  Viele Energiemanagement-Systeme und Smart-Home-Lösungen werden von großen Cloud-Anbietern wie Amazon, Google oder Microsoft betrieben. Stromkunden, die diese

Systeme nutzen, müssen ihre Daten an diese Cloud-Anbieter weitergeben. Mit einem MQTT-Broker beim Energieversorger können Stromkunden ihre Daten in einer sicheren Umgebung speichern und verarbeiten.

- **Digitaler Dienst "aus der Nachbarschaft"**
  Ein MQTT-Broker beim Energieversorger ist ein digitaler Dienst, der "aus der Nachbarschaft" kommt. Stromkunden können sich darauf verlassen, dass ihre Daten sicher sind und dass sie einen guten Service erhalten.

Digitale Dienste aus der Nachbarschaft

Die Energiewende und die Digitalisierung des Energiesektors stellen Energieversorger vor große Herausforderungen, aber auch vor neue Chancen. Energieversorger, die sich erfolgreich an die neuen Gegebenheiten anpassen und ihren Kunden innovative und digitale Dienste anbieten, werden in Zukunft erfolgreich sein.

Das STROMDAO Energy Application Framework und ein MQTT-Broker, der beim Energieversorger gehostet wird und an das EAF angebunden ist, bieten Energieversorgern die Möglichkeit, sich an die neuen Gegebenheiten anzupassen und ihren Kunden eine Reihe von innovativen und digitalen Diensten anzubieten. Dies bietet Energieversorgern eine Reihe von Vorteilen, darunter Effizienz, Transparenz, Flexibilität, Innovation und neue Geschäftsmöglichkeiten.

STROMDAO Energy Application Framework

Das STROMDAO EAF (Energy Application Framework) ist eine Open-Source Plattform, die Energieversorgern die Möglichkeit bietet, dynamische Stromtarife zu implementieren und zu verwalten. Das EAF bietet eine Reihe von Vorteilen für Energieversorger, darunter:

- **Effizienz**
  Es automatisiert die Prozesse der Datenerfassung, Tarifberechnung und Abrechnung. Dies spart Energieversorgern Zeit und Geld.
- **Transparenz**
  Das Framework bietet Kunden einen transparenten Einblick in ihre Stromkosten in Echzeit via APP. Dies stärkt das Vertrauen der Kunden in den Energieversorger.
- **Flexibilität**
  Es ermöglicht es Energieversorgern, ihre Stromtarife flexibel an die Bedürfnisse ihrer Kunden und die aktuellen Marktbedingungen anzupassen.
- **Innovation**
  Das EAF bietet Energieversorgern die Möglichkeit, neue innovative Stromprodukte und -dienstleistungen zu entwickeln.

Durch den Einsatz des STROMDAO EAF können Energieversorger ihren Kunden eine Reihe von digitalen Diensten anbieten, die Mehrwerte für beide Seiten bieten.

Mehrwerte für Energieversorger

- **Kundenbindung**
  Durch die Bereitstellung innovativer und digitaler Dienste können Energieversorger ihre Kunden an sich binden und die Kundenabwanderung reduzieren.
- **Umsatzsteigerung**
  Durch die Einführung neuer Stromprodukte und -dienstleistungen können Energieversorger ihren Umsatz steigern.
- **Imageverbesserung**
  Durch die Bereitstellung digitaler Dienste aus der Nachbarschaft können Energieversorger ihr Image als innovatives und kundenorientiertes Unternehmen verbessern.

Chancen für Energieversorger

Das STROMDAO EAF bietet Energieversorgern die Möglichkeit, sich zu digitalen Dienstleistern zu entwickeln. Dies eröffnet ihnen eine Reihe von Chancen, darunter:

- **Neue Geschäftsmodelle**
  Energieversorger können neue Geschäftsmodelle entwickeln, die auf digitalen Diensten basieren.
- **Partnerschaften**
  Energieversorger können Partnerschaften mit anderen Unternehmen eingehen, um neue digitale Dienste zu entwickeln und anzubieten.
- **Wachstum**
  Energieversorger können durch die Bereitstellung digitaler Dienste ihr Wachstum beschleunigen.

MQTT-Broker

Ein MQTT-Broker ist ein Server, der Nachrichten zwischen MQTT-Clients vermittelt. MQTT-Clients sind Geräte, die Daten über MQTT-Themen austauschen. Ein MQTT-Thema ist eine Zeichenfolge, die den Inhalt der Nachrichten beschreibt, die über das Thema ausgetauscht werden.

MQTT ist ein weit verbreiteter Standard für den Austausch von Energiedaten. Er wird von vielen Energiemanagement-Systemen und Smart-Home-Lösungen unterstützt. Dies macht es für Energieversorger einfach, ihre Systeme mit den Systemen ihrer Kunden zu verbinden und Energiedaten in Echtzeit auszutauschen.

Ein MQTT-Broker, der beim Energieversorger gehostet wird und an das STROMDAO EAF angebunden ist, bietet Energieversorgern und ihren Kunden eine Reihe von Vorteilen, darunter:

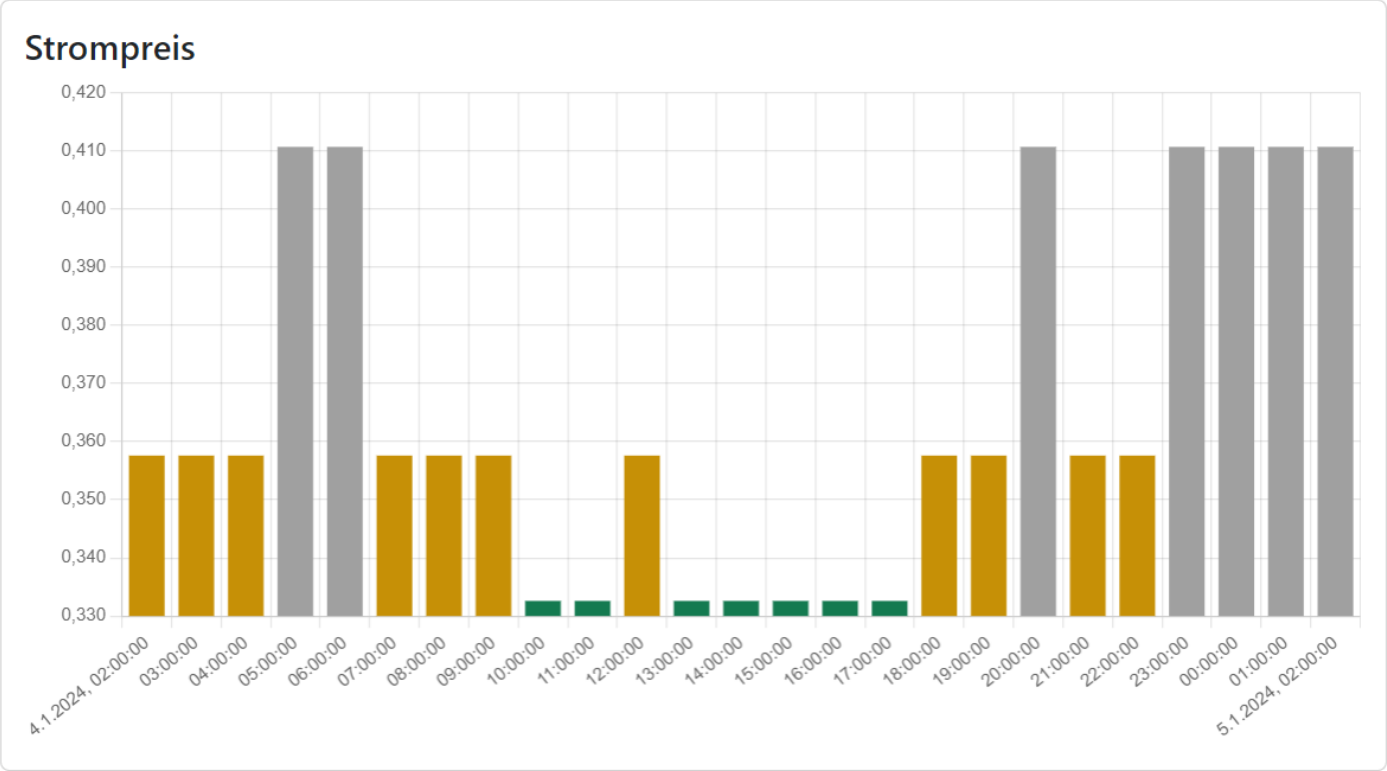
- **Echtzeitdaten zum Stromverbrauch**
  Energieversorger können den Stromverbrauch ihrer Kunden in Echtzeit verfolgen. Dies ermöglicht es ihnen, ihre Stromerzeugung und -verteilung besser zu planen und zu optimieren.
- **Verbesserte Kundenbetreuung**
  Energieversorger können ihren Kunden einen besseren Service bieten. Sie können

beispielsweise ihren Kunden helfen, ihren Stromverbrauch zu optimieren und Geld zu sparen.

- **Neue Geschäftsmodelle**
  Energieversorger können neue Geschäftsmodelle entwickeln, wie z.B. Energy As A Service (EaaS).

EAF - App für dynamische Stromtarife

Strompreis



Der Kunde kann anhand dieser Ansicht seinen Stromverbrauch planen und Geräte, die nicht automatisch gesteuert werden können, trotzdem am Strompreis ausrichten. Wenn der Strompreis hoch ist, kann der Kunde seinen Stromverbrauch reduzieren, indem er z. B. die Waschmaschine oder den Geschirrspüler nicht anstellt. Wenn der Strompreis niedrig ist, kann der Kunde seinen Stromverbrauch erhöhen, indem er z. B. das Elektroauto lädt oder die Heizung einschaltet.

Die großen Verbraucher wie Wärmepumpen und Wallboxen rufen die Daten über eine API ab und steuern ihren Verbrauch automatisch. Die Ansicht dient hier zur Information des Stromkunden. Der Kunde kann sehen, wann die Wärmepumpe oder die Wallbox aktiv ist und wie viel Strom sie verbraucht.

Die Ansicht ist für den Kunden sehr nützlich, da er damit seinen Stromverbrauch optimieren und Geld sparen kann.

Abrechnungsdaten

Abrechnung Haushaltszähler

Niedertarif	Mitteltarif	Hochtarif	Gesamt
1,966kWh / 0,65€	1,381kWh / 0,49€	1,702kWh / 0,70€	5,049kWh / 1,85€

Verbrauch

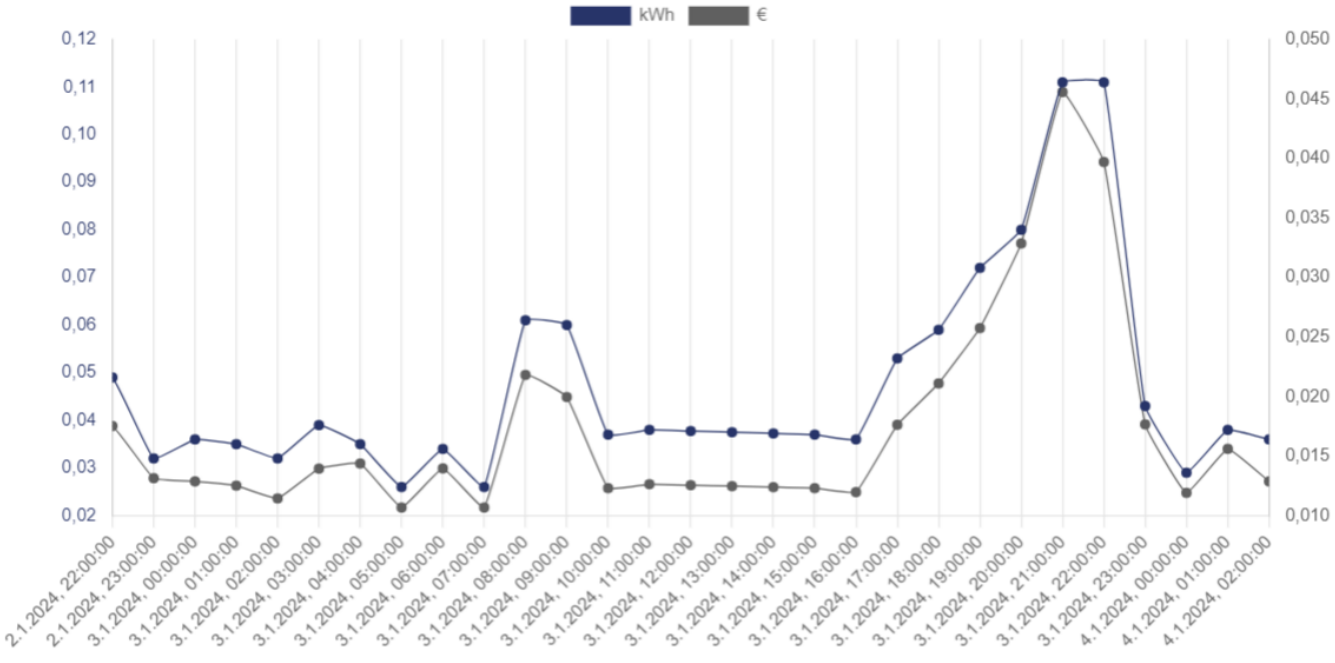


Kosten



⌀ kWh je Euro	3,1	⌀ Preis je kWh	0,320€
⌀ kWh je Tag	8,1	⌀ Kosten je Tag	2,58€
⌀ kWh je Monat	242,6	⌀ Kosten je Monat	77,55€

Verlauf



Die Ansicht der Abrechnungsinformationen bietet dem Kunden einen Überblick über die Verteilung seines Strombezugs zu den Strompreisen, die zu diesem Zeitpunkt gegolten haben. Es werden die Echtzeitdaten seit der letzten Abrechnung angezeigt.

Der Stromkunde kann in der Ansicht sehen, wie viel Strom er zu welchem Preis bezogen hat.

Die Ansicht der Abrechnungsinformationen ist für den Kunden sehr nützlich, da er damit seinen Stromverbrauch und seine Stromkosten besser verstehen kann. Er kann sehen, zu welchen Zeiten er besonders viel Strom verbraucht hat und wie hoch die Strompreise zu diesen Zeiten waren.

Der Kunde kann die Informationen auch nutzen, um seinen Stromverbrauch zu optimieren und Geld zu sparen. Beispielsweise kann er versuchen, seinen Stromverbrauch in Zeiten mit niedrigen Strompreisen zu verlagern.

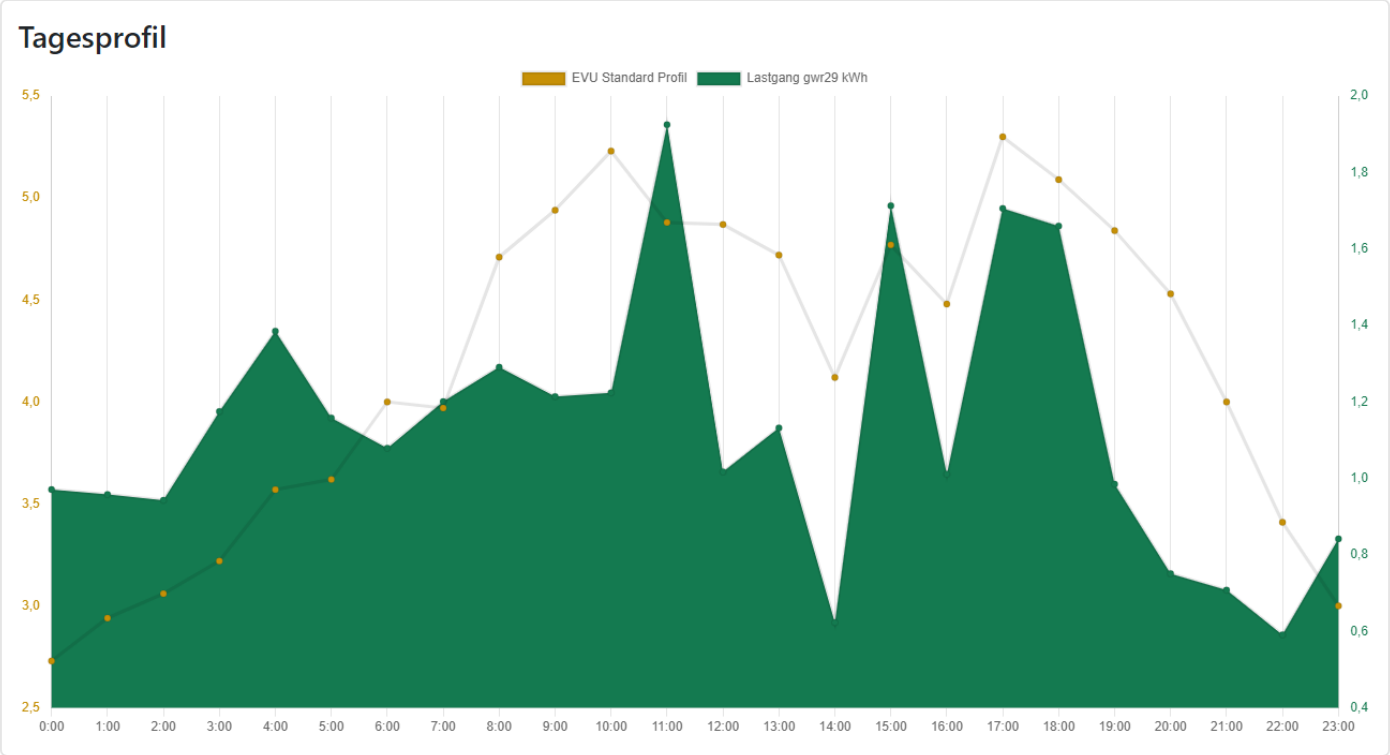
Beispiel:

Der Kunde sieht in der Ansicht der Abrechnungsinformationen, dass er am 15.03.2023 zwischen 18:00 und 19:00 Uhr 1 kWh Strom zu einem Preis von 30 ct/kWh bezogen hat. Am 16.03.2023 zwischen 12:00 und 13:00 Uhr hat er 2 kWh Strom zu einem Preis von 20 ct/kWh bezogen.

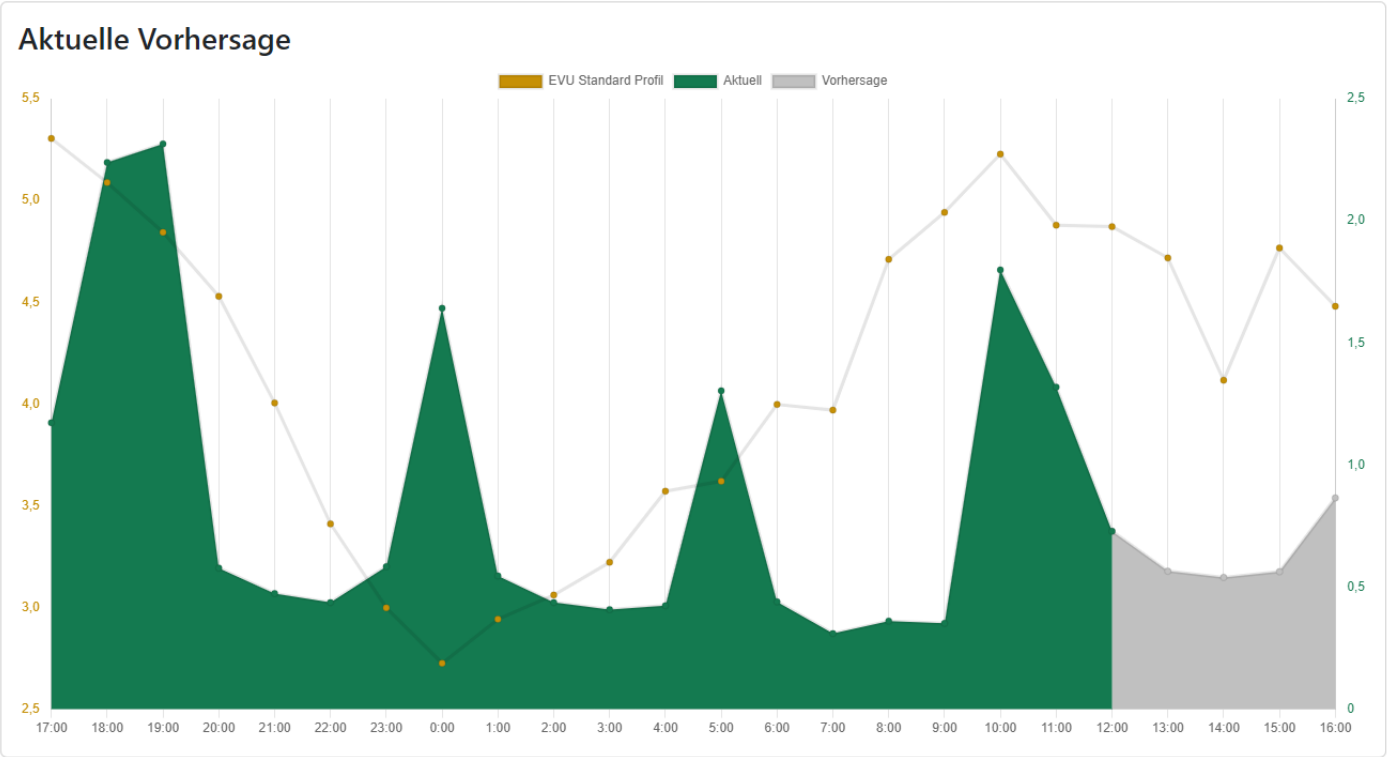
Der Kunde kann aus diesen Informationen schließen, dass der Strompreis am 15.03.2023 zwischen 18:00 und 19:00 Uhr höher war als am 16.03.2023 zwischen 12:00 und 13:00 Uhr. Wenn der Kunde seinen Stromverbrauch optimieren möchte, kann er versuchen, seinen Verbrauch in Zeiten mit niedrigen Strompreisen zu verlagern.

Die Ansicht der Abrechnungsinformationen ist ein wichtiges Werkzeug für den Kunden, um seinen Stromverbrauch und seine Stromkosten zu verstehen und zu optimieren.

Lastprofil für variable Stromtarife



Auf Seite des Energieversorgers bietet das STROMDAO EAF eine Visualisierung in Echtzeit je Messstelle. In dieser Visualisierung wird für den Stromkunden sein Lastprofil basierend auf hochfrequenten Zählerdaten gezeigt. In orange ist das Gesamtprofil des Energieversorgers gezeigt, so dass ein schneller Überblick über die Profiltreue der Messstelle erkannt werden kann.



Die Prognose wird auf Basis der historischen Daten der jeweiligen Messstelle erstellt. Dabei werden verschiedene Faktoren berücksichtigt, wie z.B. die Tageszeit, die Wochentage und die Jahreszeiten. Die Prognose wird kontinuierlich aktualisiert, sodass sie immer auf dem neuesten Stand ist.

Einführung - Energieanwendungen

Energieanwendungen umfassen die Verfahren und Technologien zur Umwandlung und Verteilung von Energie in nutzbare Formen. Sie umfassen Erzeuger, Speicher, Verteiler und Verbraucher sowie Mess-, Steuerungs- und Automatisierungssysteme. Energieanwendungen sind in übergeordnete Netzwerke und Marktstrukturen integriert und stehen in Wechselwirkung mit dem Energiemarkt, Regulierungsbehörden und anderen Teilnehmern des Energiesystems.

Grundlagen

Energieanwendungen bezeichnen das breite Spektrum von Verfahren und Technologien, die darauf abzielen, Energie in nutzbare Formen umzuwandeln und zu verteilen, um spezifische Bedürfnisse und Anforderungen zu erfüllen. Diese reichen von alltäglichen Vorgängen wie der Beleuchtung und Heizung von Gebäuden über die Energieversorgung industrieller Prozesse bis hin zum Betrieb von elektrischen Fahrzeugen.

Die Anwendungen selbst beruhen auf einer Vielzahl von Ausrüstungen und Technologien, wie zum Beispiel (vergl. Komponenten von Energieanwendungen):

1. **Erzeuger**: Dies können Photovoltaikanlagen sein, die Sonnenlicht in elektrische Energie umwandeln, Windturbinen, die kinetische Energie des Windes nutzen, oder konventionelle Generatoren, die fossile Brennstoffe verbrennen.
2. **Speicher**: Batterien und andere Energiespeichersysteme, die Energie für den späteren Gebrauch aufbewahren können.
3. **Verteiler**: Das Stromnetz, das die erzeugte Energie an das Endziel leitet, sowie Ladestationen für Elektrofahrzeuge (Wallboxen), die eine Schnittstelle für das Laden der Fahrzeuge bieten.
4. **Verbraucher**: Endgeräte, die Energie in eine gewünschte Form umsetzen, wie beispielsweise Wärmepumpen für die Heizung, Elektroherde für die Zubereitung von Speisen oder Leuchtdioden (LEDs) für die Beleuchtung.

Zentral für die Funktionsweise von Energieanwendungen sind messbare und steuerbare Prozesse. Messwerte wie erzeugte oder verbrauchte Mengen an Energie sind essenziell, um Energieflüsse zu überwachen, zu regulieren und abzurechnen. Energiezähler und intelligente Messsysteme spielen hierbei eine grundlegende Rolle.

Die aufgeführten Prozesse werden durch Überwachungs-, Steuerungs- und Automatisierungssysteme integriert, die über Schnittstellen nach außen verfügen, um Informationen und physische Ressourcen mit anderen Systemen auszutauschen, sowie um wirtschaftliche Transaktionen abzuwickeln.

Ein wichtiger Aspekt von Energieanwendungen ist deren Integration in übergeordnete Netzwerke und Marktstrukturen. Die Erzeugung und der Verbrauch von Energie sind nicht isoliert zu sehen, sondern stehen in Wechselwirkung mit dem Energiemarkt, Regulierungsbehörden und anderen Teilnehmern des Energiesystems. Die Vernetzung und der Datenaustausch zwischen Energieanwendungen und den jeweiligen Marktakteuren werden zunehmend durch digitale Technologien erleichtert, was unter anderem im Konzept des "Smart Grids" verankert ist.

Zusammenfassend umfassen Energieanwendungen eine Vielzahl komplexer Systeme und Technologien, die darauf ausgerichtet sind, Energie zu generieren, zu speichern, zu verteilen und zu nutzen. Sie umfassen darüber hinaus auch das Management dieser Prozesse, um eine zuverlässige, nachhaltige und effiziente Energieversorgung sicherzustellen, die auf die verschiedenen Bedürfnisse und Anforderungen der verschiedenen Stakeholder abgestimmt ist.

Komponenten von Energieanwendungen

Energieanwendungen umfassen eine Vielzahl komplexer Systeme und Technologien, die darauf ausgerichtet sind, Energie zu generieren, zu speichern, zu verteilen und zu nutzen. Um diese Aufgaben zu erfüllen, bestehen Energieanwendungen aus verschiedenen Komponenten, die ineinandergreifen und miteinander kommunizieren. Diese Komponenten bilden ein komplexes Gefüge, in dem die Anpassung und Abstimmung von erzeugter und verbrauchter Energie, die flexible und intelligente Speicherung sowie die effiziente Verteilung und Nutzung essenziell sind. Übergeordnet wird dies durch fortgeschrittene Mess- und Steuerungstechnologien unterstützt, die die Interaktion zwischen den einzelnen Komponenten und mit dem Energiesystem als Ganzem erleichtern.

1. **Erzeuger**: Zu den Erzeugern gehören vor allem erneuerbare Energiequellen wie Photovoltaikanlagen und Windturbinen, aber auch konventionelle Generatoren. Der Fokus liegt auf der Nutzung und dem Ausgleich der Volatilität, die insbesondere mit erneuerbaren Energiequellen einhergeht, da diese stark vom Wetter abhängig sind.

2. **Speicher**: Batterien und andere Energiespeichersysteme sind kritisch für die zeitliche Entkopplung von Erzeugung und Verbrauch, indem sie Energie aufnehmen, wenn ein Überschuss vorhanden ist, und sie bereitstellen, wenn eine höhere Nachfrage besteht. Dadurch tragen sie zur Wahrung des Gleichgewichts bei und ermöglichen eine Optimierung der Wertschöpfung.

3. **Verteiler**: Das Stromnetz und die Infrastruktur wie Ladestationen für Elektrofahrzeuge ermöglichen die räumliche Distribution der Energie. Während das öffentliche Stromnetz als gemeinnützige Ressource dient, ist für Energieanwendungen vor allem das Management von Engpässen relevant, um eine effiziente Verteilung zu gewährleisten.

4. **Verbraucher**: Wärmepumpen, Elektroherde und weitere Verbraucher stellen die Nachfrageseite dar. Bei Energieanwendungen werden durch Flexibilität und Steuerung die Betriebskosten optimiert, um eine maximale Ressourceneffizienz zu erzielen. Wichtige Aspekte dabei sind sowohl die variablen Energiepreise als auch die Investitionskosten für notwendige Infrastruktur.

5. **Mess-, Steuerungs- und Automatisierungssysteme**: Diese Systeme liefern die erforderlichen Daten und ermöglichen aktives Eingreifen in den Prozess. Das Zusammenspiel mit Energiemanagementsystemen ist hierbei von zentraler Bedeutung,

um Energieflüsse zu überwachen, zu optimieren und zu automatisieren.

Energieanwendungen bilden ein komplexes Gefüge, in dem die Anpassung und Abstimmung von erzeugter und verbrauchter Energie, die flexible und intelligente Speicherung sowie die effiziente Verteilung und Nutzung essenziell sind. Übergeordnet wird dies durch fortgeschrittene Mess- und Steuerungstechnologien unterstützt, die die Interaktion zwischen den einzelnen Komponenten und mit dem Energiesystem als Ganzem erleichtern.

Integration von Energieanwendungen

Die Integration von Energieanwendungen in übergeordnete Netzwerke und Marktstrukturen ist ein wesentlicher Bestandteil der modernen Energielandschaft. Dieser Integrationsprozess ist maßgeblich von technischen Standards, regulatorischen Rahmenbedingungen sowie fortschrittlichen Kommunikationsprotokollen geprägt.

Zentral dabei ist die Marktkommunikation, die durch Standardisierung nach EDI@Energy ein reibungsloses Zusammenspiel verschiedener Akteure im Stromnetz ermöglicht. Energieanwendungen, die Teil dieses Netzwerks sind, müssen demgemäß in der Lage sein, diese standardisierten Nachrichten zu senden und zu empfangen. Solche Kapazitäten sind unabdingbar für den Energiehandel und dessen Funktionsfähigkeit, da die Abwicklung von Marktereignissen auf einem synchronisierten und regelkonformen Informationsaustausch basiert.

Die Wechselbeziehung von Energieanwendungen mit dem Energiemarkt, Regulierungsbehörden und anderen Akteuren manifestiert sich nicht nur in der Notwendigkeit zur Einhaltung von Kommunikationsprotokollen, sondern auch in der Erfüllung weiterer regulatorischer Auflagen. Dazu gehören Berichtspflichten und Protokollierungen, die den transparenten und konformen Betrieb von Energiesystemen sicherstellen sollen.

Übergeordnete Netzwerke und Marktstrukturen

Nach der Liberalisierung der Stromnetze wurde die Marktkommunikation eingeführt, die es den unterschiedlichen Akteuren des Stromnetzes erlaubt, untereinander nach einem festgelegten Standard (EDI@Energy) zu kommunizieren und Marktereignisse entlang der regulatorischen Rahmenbedingungen abzuwickeln. Je nach Aufgabe der Energieanwendung ist es notwendig, dass diese Nachrichten der Marktkommunikation empfangen oder senden kann. Das gesamte Aufgabenfeld des Energiehandels funktioniert nur durch die Integration in die Marktkommunikation.

Wechselwirkung mit dem Energiemarkt, Regulierungsbehörden und anderen Teilnehmern des Energiesystems

Die bereits genannte Marktkommunikation ist hierbei das Protokoll der Kommunikation, aus der sich Wechselwirkungen für die internen Prozesse der Energieanwendung ergeben. Es existieren jedoch auch zusätzliche regulatorische Anforderungen, besonders an die Berichterstellung und Protokollierung, welche unabhängig von der Marktkommunikation sind.

Des Weiteren hat das Konzept der "Smart Grids" eine Fülle von Möglichkeiten für den interaktiven Datenaustausch der verschiedenen Komponenten gebracht, welche am Energiesystem beteiligt sind. Für Energieanwendungen relevant sind hierbei MQTT, SML (Smart Meter Language), SCADA, KNX, MODBUS und viele mehr. Wobei manche ganze Systemtypen bezeichnen, manche Kommunikationsprotokolle sind. Beim Austausch von Energiedaten, welche abseits der Marktkommunikation ist, ist durch fehlende Regulierung und Standardisierung in den letzten Jahrzehnten ein Wildwuchs entstanden. In vielen Fällen nutzen Energieanwendungen für konkrete Protokolle die Dienste eines Energiemanagementsystems, welches die entsprechenden Adapter implementiert.

Herausforderungen

Die Integration von Energieanwendungen in übergeordnete Netzwerke und Marktstrukturen ist mit einer Reihe von Herausforderungen verbunden, darunter:

- Die Komplexität der Marktregeln und -vorschriften
- Die Notwendigkeit, sich an unterschiedliche Kommunikationsstandards anzupassen
- Die Gewährleistung der Datensicherheit und des Datenschutzes
- Die Interoperabilität verschiedener Systeme und Technologien

Lösungen:

Um diese Herausforderungen zu bewältigen, gibt es eine Reihe von Lösungen, darunter:

- Die Entwicklung von standardisierten Schnittstellen und Protokollen
- Die Bereitstellung von Schulungen und Unterstützung für die Akteure des Energiesystems
- Die Förderung von Forschung und Entwicklung im Bereich der Energieintegration
- Die Zusammenarbeit zwischen den verschiedenen Interessengruppen des Energiesystems

Vorteile

Die Integration von Energieanwendungen in übergeordnete Netzwerke und Marktstrukturen bietet eine Reihe von Vorteilen, darunter:

- Eine effizientere und zuverlässigere Energieversorgung
- Eine bessere Ausnutzung erneuerbarer Energien
- Eine Reduzierung der Energiekosten
- Eine geringere Umweltbelastung

Das Konzept der Smart Grids hat die Interaktionsmöglichkeiten zwischen den einzelnen Elementen des Energiesystems weiter erhöht. Smart Grids ermöglichen einen interaktiven Datenaustausch, wobei Protokolle und Systemtypen wie MQTT, SML (Smart Meter Language), SCADA, KNX und MODBUS maßgeblich zum Einsatz kommen. Während einige davon als Kommunikationsprotokolle fungieren, bezeichnen andere komplette Systemtypen.

Aufgrund nur teilweise vorhandener Regulierung und Standardisierung ist in manchen Bereichen des Datenaustausches abseits der Marktkommunikation eine Vielfalt an Lösungen entstanden, die bisweilen als "Wildwuchs" angesehen werden können. Energieanwendungen greifen in solchen Fällen häufig auf die Dienste von Energiemanagementsystemen zurück, welche die erforderlichen Adapter und Schnittstellen zur Verfügung stellen, um eine solide Integration zu ermöglichen. Diese Systeme erleichtern die Handhabung der vielfältigen und oft komplexen Protokolle und ermöglichen somit erst den effizienten Betrieb moderner Energieanwendungen.

Fazit

Die Integration von Energieanwendungen in übergeordnete Netzwerke und Marktstrukturen ist ein wichtiger Schritt hin zu einem nachhaltigen und effizienten Energiesystem. Es gibt zwar noch eine Reihe von Herausforderungen zu bewältigen, aber die Vorteile einer erfolgreichen Integration überwiegen deutlich.

Beispiele für Energie Anwendungen

Die Energiewende stellt die Energieversorgung vor große Herausforderungen. Es gilt, die Abhängigkeit von fossilen Brennstoffen zu reduzieren, die Energieeffizienz zu erhöhen und die Nutzung erneuerbarer Energien auszubauen. Gleichzeitig müssen die Energiepreise bezahlbar bleiben und die Versorgungssicherheit gewährleistet sein.

Energieanwendungen spielen eine Schlüsselrolle bei der Bewältigung dieser Herausforderungen. Sie ermöglichen es, Energie effizienter zu nutzen, die Integration erneuerbarer Energien zu erleichtern und die Flexibilität des Energiesystems zu erhöhen.

Die folgende Liste enthält eine Reihe von Energieanwendungen, die dazu beitragen können, die Energiewende voranzutreiben:

- **Ladetarife für die E-Mobilität**: Tarifmodelle, die auf die spezifischen Bedürfnisse von Elektrofahrzeugen zugeschnitten sind und das Laden zu kosteneffizienten oder ökologisch günstigen Zeiten fördern.

- **Dynamische Stromtarife**: Tarifsysteme, die den Strompreis nach Angebot und Nachfrage bzw. Tageszeit variieren, um die Energieeffizienz zu erhöhen und das Lastmanagement zu optimieren.

- **CO2-Reduktion der Stromversorgung**: Anwendungen und Maßnahmen zur Verringerung des CO2-Fußabdrucks durch die Nutzung regenerativer Energien und die Erhöhung der Energieeffizienz.

- **Energy As A Service**: Dienstleistungskonzepte, die Kunden eine auf ihre Bedürfnisse zugeschnittene Energieversorgung bieten, ohne dass diese in Infrastruktur investieren müssen.

- **Belieferung von Dritten (Abrechnung)**: Systeme zur Messung, Abrechnung und zum Management der Energieversorgung von Dritten, beispielsweise in Mieterstrommodellen.

- **Bewirtschaftung unterbrechbarer oder steuerbarer Lasten**: Anwendungen, die große Verbraucher wie Industrieanlagen während Spitzenlastzeiten steuern, um Netzstabilität zu garantieren und Kosten zu senken.

- **Produktionsplanung auf Basis der Stromkosten**: Strategien zur Anpassung von Produktionsprozessen an die aktuellen Energiepreise, um die Betriebskosten zu minimieren.

- **Optimierte Beschaffung von Strom**: Einsatz von Algorithmen und Marktwissen zur Beschaffung von Strom zu optimalen Konditionen.

- **Energiegemeinschaft / Stromkollektiv**: Zusammenschlüsse von Verbrauchern und Erzeugern, um Energie effizient zu allozieren und kosten- bzw. umweltbewusst zu nutzen.

- **Hybridstrommarkt**: Kombination aus reguliertem Strommarkt und dezentralen, teilweise autonomen Energiesystemen wie Microgrids, die flexibel auf Markt- und Systemanforderungen reagieren.

Einführung dynamischer Stromtarife

Die Einführung dynamischer Stromtarife stellt Energieversorger vor neue Herausforderungen.Beleuchtet werden die Anforderungen an die Versorger, darunter: Kundenschnittstellen: Entwicklung von Apps, APIs und Websites für die Interaktion mit Kunden IT-Integration: Anbindung an Messstellenbetrieb, Messdatenverarbeitung und Rechnungsstellung Energieversorgung: Abkehr vom Standardlastprofil und Auswirkungen auf die Bewirtschaftung Tarifvarianten: Erläuterung verschiedener dynamischer Stromtarifmodelle

Rahmenbedingungen für dynamische Stromtarife

Gesetzlicher und markttechnischer Rahmen

Die Novellierung des § 14a des Energiewirtschaftsgesetzes (EnWG) schafft die rechtliche Grundlage für die Einführung dynamischer Stromtarife. Diese Tarife ermöglichen es, den Strompreis im Tagesverlauf an die tatsächlichen Erzeugungs- und Verbrauchskosten anzupassen. Dadurch wird ein Anreiz für Verbraucher geschaffen, ihren Stromverbrauch in Zeiten hoher Erzeugung und niedriger Nachfrage zu verlagern.

Die Umsetzung dynamischer Stromtarife erfordert eine entsprechende Marktstruktur. Das Energy Application Framework (EAF) bietet eine standardisierte Schnittstelle zwischen Netzbetreibern und Aggregatoren, die den Datenaustausch und die Steuerung von Demand Response ermöglicht.

Demand Response und Energy as a Service

Demand Response ermöglicht es Verbrauchern, ihren Stromverbrauch auf Signale des Netzbetreibers hin zu reduzieren oder zu erhöhen. Durch die Integration von Demand Response in dynamische Stromtarife können Verbraucher ihre Stromkosten senken und gleichzeitig zur Stabilisierung des Stromnetzes beitragen.

Energy as a Service (EaaS) ist ein Geschäftsmodell, bei dem Verbraucher nicht für den Stromverbrauch selbst bezahlen, sondern für eine bestimmte Dienstleistung, wie z. B. Beleuchtung oder Klimatisierung. EaaS-Anbieter können dynamische Stromtarife nutzen, um ihre Kosten zu optimieren und den Verbrauchern flexible und kostengünstige Energielösungen anzubieten.

Die Einführung dynamischer Stromtarife erfordert eine enge Zusammenarbeit zwischen Netzbetreibern, Aggregatoren und Energieversorgern. Folgende Schritte sind dabei zu beachten:

- **Entwicklung von Tarifmodellen**
  Innerhalb der kommenden neun Monaten müssen Stromlieferanten entsprechende Tarifmodelle entwickeln. Diese Modelle sollten die tatsächlichen Erzeugungs- und Verbrauchskosten im Tagesverlauf widerspiegeln.

- **Datenaustausch**
  Netzbetreiber und Aggregatoren müssen die notwendigen Daten über Erzeugung, Verbrauch und Netzlast austauschen. Dies ermöglicht eine genaue Preisgestaltung und eine effektive Steuerung von Demand Response.

- **Steuerung von Demand Response**
  Aggregatoren müssen die Möglichkeit haben, den Stromverbrauch ihrer Kunden auf Signale des Netzbetreibers hin zu steuern. Dies kann durch Smart-Meter-Technologie und entsprechende Kommunikationsinfrastruktur erreicht werden.

- **Abrechnung**
  Energieversorger sind dafür verantwortlich, die verbrauchsabhängigen Kosten an ihre Kunden weiterzugeben. Die Abrechnungssysteme müssen in der Lage sein, die dynamischen Stromtarife zu verarbeiten und die Kosten entsprechend zuzuordnen.

Vision für die Zukunft

Die Einführung dynamischer Stromtarife wird die Energieversorgung in Deutschland flexibler und effizienter gestalten. Verbraucher werden von niedrigeren Stromkosten und einer besseren Kontrolle über ihren Verbrauch profitieren. Netzbetreiber können das Stromnetz stabiler und kostengünstiger betreiben.

In Zukunft könnten dynamische Stromtarife zu einem Standard werden, der den Weg für eine dezentralere und erneuerbare Energieversorgung ebnet. EaaS-Anbieter werden eine wichtige Rolle bei der Bereitstellung innovativer Energielösungen spielen, die den Verbrauchern Komfort und Kosteneinsparungen bieten.

Entwicklung von dynamischen Tarifmodellen

Im Zuge der digitalen Transformation der Energiewirtschaft und angesichts der dynamischen Veränderungen auf den Energiemärkten stehen Stromanbieter vor der Herausforderung, innovative und kundenorientierte Tarifmodelle zu entwickeln, die zugleich betriebswirtschaftlich tragfähig und im Einklang mit regulatorischen Rahmenbedingungen sind. Die Gestaltung solcher Tarifstrukturen erfordert eine subtile Balance zwischen Komplexität und Transparenz, um sowohl die Akzeptanz bei den Kunden zu sichern als auch die operativen Prozesse effizient zu gestalten.

Das Energy Application Framework (EAF), ein fortschrittliches Tool für die Energiebranche, bietet eine Lösung, um Tarifmodelle effektiv zu entwickeln und zu steuern. Mit seiner Hilfe können Ereignisvariablen genutzt werden, die im Kontext des deutschen Energiewirtschaftsgesetzes eine Anwendung finden und damit helfen, Tarife nicht nur flexibler zu gestalten, sondern sie auch mit geringem Aufwand im bestehenden System zu realisieren.

Grundkonzept Preismechanismus

Die zentrale Fähigkeit des EAF ist die Ereignissteuerung, welche es ermöglicht, Energiepreise an spezifische Ereignisse zu koppeln. Dies bedeutet, dass beispielsweise in Zeiten hoher Netzauslastung oder bei Erreichen bestimmter Schwellenwerte automatisch andere Tarifkonditionen geltend gemacht werden können. Die Vorteile dieser Ereignisvariablen Tarifstrukturen liegen auf der Hand: Sie erlauben es Anbietern, flexibel und zeitnah auf sich ändernde Marktbedingungen zu reagieren, ohne dass der Kunde unmittelbar mit der Komplexität der zugrunde liegenden Prozesse konfrontiert wird.

Ergänzend zur Ereignissteuerung ermöglicht die prozessuale Steuerung im EAF die Integration der Tarifmodelle in bestehende ERP-Systeme von Stromanbietern. Auf diese Weise können die Geschäftsprozesse reibungslos weitergeführt werden, während neue Tarifmodelle implementiert werden. Diese Integration sorgt für eine nahtlose Erfahrung sowohl für den Anbieter als auch für den Endverbraucher.

Anforderungen der Stromkunden

Trotz der vielfältigen Möglichkeiten, die das EAF bietet, sollten Anbieter sich bewusst sein, dass für den Kunden eine klare und einfache Kostenstruktur von entscheidender Bedeutung ist.

Komplizierte Tarife, die sich zum Beispiel direkt an Börsenstrompreisen orientieren, sind für den durchschnittlichen Stromkunden schwer nachvollziehbar und können zu Verwirrung und Misstrauen führen. Dies kann gerade im Wettbewerbsumfeld der Energieversorger kontraproduktiv sein, da Kunden eher geneigt sind, einfach verständliche und transparente Angebote zu wählen.

Ein erfolgreiches Tarifmodell balanciert daher die Komplexität der Energiepreisbildung im Hintergrund mit einfachen und verständlichen Preissignalen an den Kunden (Beispiel: GrünstromIndex) . Die Kunst liegt darin, die variablen Kosten, die durch Marktereignisse, Nachfrageveränderungen oder Netzbelastungen entstehen, in Tarifen zu kapseln, die für den Kunden als faire und vorhersehbare Kostenstruktur erscheinen.

Durch die Kombination aus Ereignis- und prozessualer Steuerung bietet das EAF dem Stromanbieter ein mächtiges Instrument, um diese Herausforderung zu meistern. Es ermöglicht die Schaffung von Tarifen, die sich im Takt mit den Anforderungen des Marktes und den Bedürfnissen der Verbraucher bewegen, zugleich aber eingebettet sind in eine Nutzererfahrung, die Einfachheit und Transparenz in den Vordergrund stellt. So kann ein Anbieter einen echten Konkurrenzvorteil erlangen und eine stabile Kundenbeziehung aufbauen, die auf Vertrauen und Klarheit gründet.

EDIFACT-Nachrichten in der Energiewirtschaft: Entwickler-Einführung

Ein praktischer Leitfaden für die deutsche Marktkommunikation

---

□□ Für wen ist diese Einführung?

Du bist Entwickler:in und stehst vor deiner ersten EDIFACT-Nachricht aus der deutschen Energiewirtschaft? Du möchtest verstehen, was all diese kryptischen Zeichen bedeuten und wie du effizient damit arbeiten kannst? Dann bist du hier richtig!

Diese Einführung:

- □ Erklärt EDIFACT von Grund auf – kein Vorwissen nötig
- □ Zeigt praktische Beispiele mit dem `edifact-json-transformer`
- □ Vermittelt regulatorisches Wissen (GPKE, GeLi Gas, WiM, MaBiS)
- □ Gibt Best Practices für die tägliche Arbeit

Powered by: Willi Mako – die intelligente MaKo-Plattform

Open Source Tool: edifact-json-transformer

□□ Inhaltsverzeichnis

1. Was ist EDIFACT und warum brauchen wir das?
2. Die vier Säulen der Marktkommunikation
3. Anatomie einer EDIFACT-Nachricht
4. Schnellstart mit dem Transformer
5. Die wichtigsten Nachrichtentypen

6. Prüfidentifikatoren verstehen
7. Praktische Tipps für den Alltag
8. Häufige Fehler und Lösungen
9. Best Practices
10. Weiterführende Ressourcen

□□ Was ist EDIFACT und warum brauchen wir das?

Die Grundidee

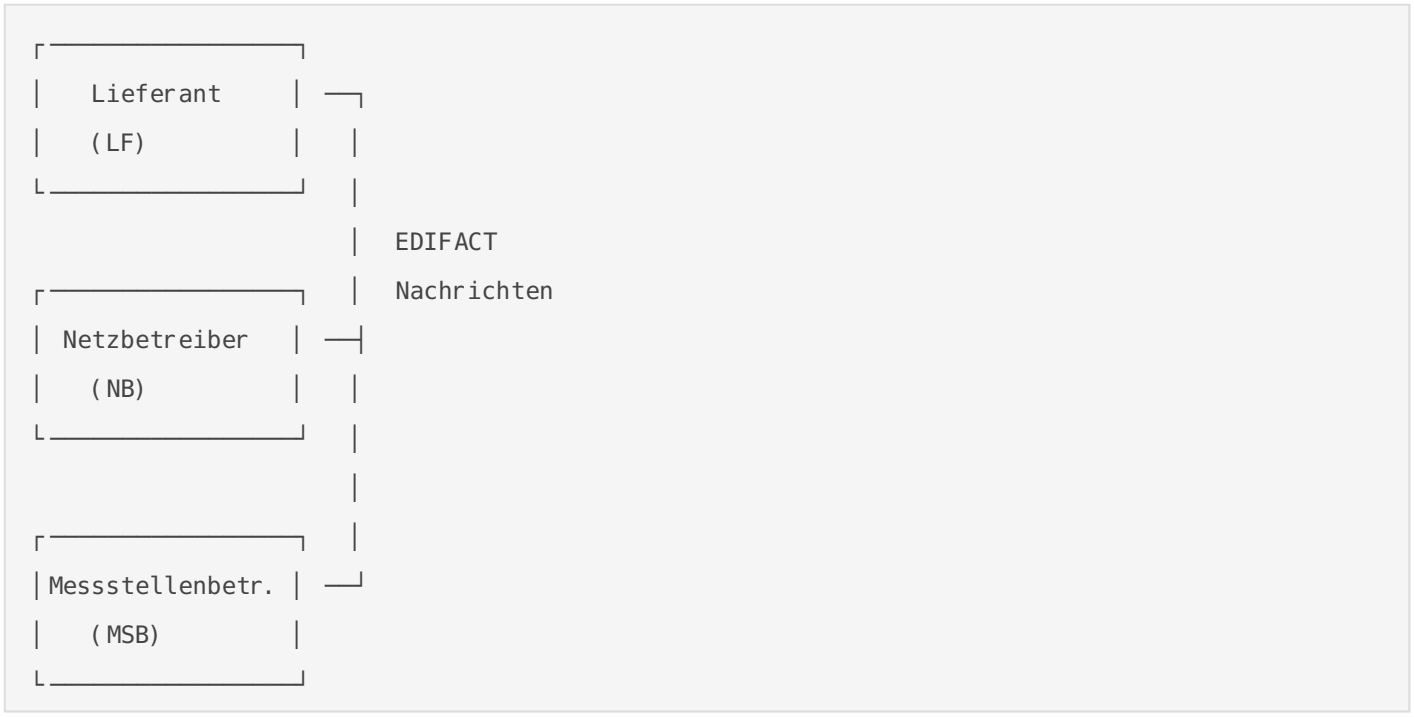
EDIFACT (Electronic Data Interchange For Administration, Commerce and Transport) ist ein UN-Standard für den elektronischen Datenaustausch. In der deutschen Energiewirtschaft ist dieser Standard **gesetzlich vorgeschrieben** und bildet das Rückgrat der Kommunikation zwischen allen Marktteilnehmern.

Rechtliche Grundlagen

Die Basis bildet das **Energiewirtschaftsgesetz (EnWG)**. Darauf aufbauend hat die **Bundesnetzagentur (BNetzA)** detaillierte Festlegungen erlassen, die jeden Aspekt der Marktkommunikation regeln:

Festlegung	Rechtsgrundlage	Was wird geregelt?
GPKE	Geschäftsprozesse Kundenbelieferung Elektrizität	Strom-Lieferantenwechsel, An-/Abmeldungen, Stammdaten
GeLi Gas	Geschäftsprozesse Lieferantenwechsel Gas	Gas-Lieferantenwechsel, Gasbelieferung
WiM	Wechselprozesse im Messwesen	Zählerwechsel, Messstellenbetrieb, Messdaten
MaBiS	Marktregeln Bilanzierung Strom	Bilanzkreisabrechnung, Ausgleichsenergie

Die Marktteilnehmer



Rollen:

- **Lieferant (LF)**: Versorgt Endkunden mit Energie, verantwortlich für Abrechnung
- **Netzbetreiber (NB)**: Betreibt das Netz, zentrale Kommunikationsstelle
- **Messstellenbetreiber (MSB)**: Betreibt Zähler, liefert Messdaten

---

□□ Die vier Säulen der Marktkommunikation

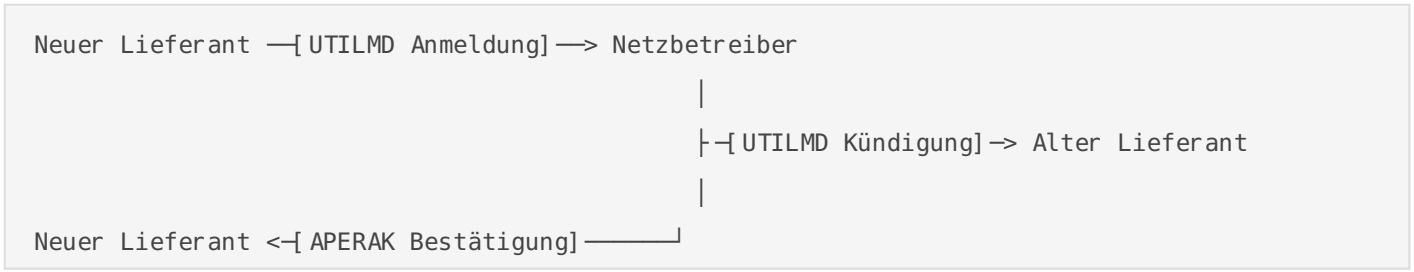
1. GPKE – Strom-Lieferantenwechsel

Kernprozesse:

- Anmeldung zur Netznutzung (Prüf-ID 44001)
- Abmeldung (Prüf-ID 44004)
- Kündigung beim bisherigen Lieferanten (Prüf-ID 44016)
- Stammdatenänderungen (Prüf-ID 44112, 44123)

Typische Nachrichten: UTILMD, MSCONS, INVOIC, APERAK

Beispiel-Workflow:



2. GeLi Gas – Gas-Lieferantenwechsel

Besonderheiten:

- Zusätzliche Brennwert- und Zustandszahl-Informationen
- Spezielle Gasbeschaffenheitsdaten in MSCONS
- ORDERS für Anfragen (z.B. nach Brennwerten)

Typische Nachrichten: UTILMD, MSCONS, ORDERS, ORDRSP, APERAK

3. WiM – Wechselprozesse im Messwesen

Kernprozesse:

- Gerätewechsel (Prüf-ID 17xxx-Serie)
- Messkonzeptwechsel (Prüf-ID 21xxx-Serie)
- Störungsmeldungen (INSRPT)
- Auftragsbestätigungen (IFTSTA)

Typische Nachrichten: ORDERS, ORDRSP, IFTSTA, INSRPT, MSCONS

Beispiel-Workflow:



4. MaBiS – Bilanzkreisabrechnung

Kernprozesse:

- Fahrplanmeldungen
- Bilanzkreiszuordnungen
- Mehr-/Mindermengenabrechnung

- Ausgleichsenergieabrechnung

Typische Nachrichten: MSCONS, PRICAT, PARTIN, COMDIS

□□ Anatomie einer EDIFACT-Nachric

Die Grundstruktur

Eine EDIFACT-Nachricht sieht zunächst kryptisch aus:

```
UNH+1+UTILMD:D:11A:UN:2.6'BGM+E01+12345+9'DTM+137:20241019:102'NAD+MS+9900123456789::293'RFF+Z13:44001'IDE+24+12345678901'UNT+6+1'
```

Zerlegt ergibt sich:

```
UNH+1+UTILMD:D:11A:UN:2.6'     ← Nachrichtenkopf
BGM+E01+12345+9'                ← Dokumentkopf
DTM+137:20241019:102'           ← Datum
NAD+MS+9900123456789::293'      ← Absender
RFF+Z13:44001'                  ← Prüfidentifikator
IDE+24+12345678901'             ← Marktlokation
UNT+6+1'                        ← Nachrichtenende
```

Die Trennzeichen

Zeichen	Bedeutung	Beispiel
`'`	Segment-Ende	`NAD+MS+123'`
`+`	Datenelemente-Trenner	`NAD+MS+123`
`:`	Komponenten-Trenner	`RFF+Z13:44001`
`?`	Escape-Character	`?+` für echtes `+`
`,`	Dezimaltrennzeichen	`1234,56`

Wichtige Segmente im Überblick

Segment	Vollständiger Name	Inhalt	Beispiel
UNH	Message Header	Nachrichtentyp, Version, Referenz	`UNH+1+UTILMD:D:11A:UN:2.6'`
BGM	Beginning of Message	Dokumentart, Nummer, Funktion	`BGM+E01+12345+9'`
DTM	Date/Time/Period	Datum/Zeit mit Qualifier	`DTM+137:20241019:102'`
NAD	Name and Address	Partei-Informationen	`NAD+MS+9900123456789::293'`
RFF	Reference	Referenzen (z.B. Prüf-ID)	`RFF+Z13:44001'`
IDE	Identity	Objekt-Identifikation	`IDE+24+12345678901'`
LOC	Place/Location	Ortsangaben	`LOC+172+DE'`
QTY	Quantity	Mengenangaben	`QTY+220:1234.5:KWH'`
MOA	Monetary Amount	Geldbeträge	`MOA+77:1234.56:EUR'`
CCI	Characteristic	Eigenschaften/Klassen	`CCI+Z19++++Bilanzkreis'`
UNT	Message Trailer	Segmentanzahl, Referenz	`UNT+6+1'`

Kardinalitäten verstehen

In den BDEW-Anwendungshandbüchern (AHB) findest du Kardinalitäten:

Code	Bedeutung	Beschreibung
M	Mandatory	Pflichtfeld, muss immer vorhanden sein
C	Conditional	Bedingt, abhängig vom Kontext
R	Required (BDEW)	BDEW-spezifische Pflichtfelder
D	Dependent	Abhängig von anderen Feldern
N	Not used	Nicht verwendet in diesem Kontext

□□ Schnellstart mit dem Transforme

Installation

```
npm install edifact-json-transformer
```

Deine erste Transformation

```
const { EdifactTransformer } = require('edifact-json-transformer');

// Transformer mit Validierung erstellen
const transformer = new EdifactTransformer({
  enableAHBValidation: true,      // AHB-Regeln prüfen
  parseTimestamps: true,           // Datumswerte formatieren
  generateGraphRelations: true     // Graph-Beziehungen erstellen
});

// EDIFACT-String (z.B. aus Datei gelesen)
const edifactString = `
UNH+1+UTILMD:D:11A:UN:2.6'
BGM+E01+12345+9'
DTM+137:20241019:102'
NAD+MS+9900123456789::293'
NAD+MR+9900987654321::293'
RFF+Z13:44001'
IDE+24+12345678901'
UNT+7+1'
`;

// Transformation durchführen
const json = transformer.transform(edifactString);

// Ergebnis verwenden
console.log('Nachrichtentyp:', json.metadata.message_type);
console.log('Prüf-ID:', json.metadata.pruefidentifikator);
console.log('Marktlokationen:', json.body.stammdaten.marktlokationen);
```

Ausgabe:

```
{
  metadata: {
    message_type: 'UTILMD',
    message_name: 'Stammdaten',
    category: 'master_data',
    applicable_processes: ['GPKE', 'GeLi Gas', 'WiM', 'MaBiS'],
```

```
    version: '2.6',
    pruefidentifikator: {
      id: '44001',
      description: 'Anmeldung NN'
    }
  },
  body: {
    stammdaten: {
      marktlokationen: [
        { id: '12345678901', valid: true }
      ]
    }
  },
  parties: {
    sender: {
      id: '9900123456789',
      role: 'MS',
      valid_mp_id: true
    },
    receiver: {
      id: '9900987654321',
      role: 'MR',
      valid_mp_id: true
    }
  }
}
```

□□ Die wichtigsten Nachrichtentype im Detail

1. UTILMD – Utility Master Data (Stammdaten)

Zweck: Austausch von Stammdaten zu Netzanschlüssen, Messlokationen und Lieferantenzuordnungen

Zentrale Segmente:

- **UNH, BGM**: Nachrichtenkopf
- **DTM**: Zeitangaben (Lieferbeginn/-ende)
- **RFF+Z13**: Prüfidentifikator (Geschäftsvorfall)
- **NAD**: Beteiligte Marktpartner
- **IDE+24**: Marktlokations-ID (11-stellig)
- **IDE+25**: Messlokations-ID (33-stellig)
- **LOC**: Lokationsangaben
- **CCI**: Eigenschaften (z.B. Bilanzkreis)

Code-Beispiel:

```
const json = transformer.transform(utilmdString);

// Stammdaten auslesen
const malo = json.body.stammdaten.marktlokationen[0];
console.log(`MaLo-ID: ${malo.id}, gültig: ${malo.valid}`);

// Prüfidentifikator prüfen
if (json.metadata.pruefidentifikator.id === '44001') {
  console.log('Anmeldung zur Netznutzung');
}

// Bilanzkreis ermitteln
const bk = json.body.stammdaten.bilanzkreise[0];
console.log(`Bilanzkreis: ${bk.id}`);
```

Typische Anwendungsfälle:

- Lieferantenwechsel (GPKE/GeLi Gas)
- Stammdatenpflege
- Messlokations-Anmeldung (WiM)

---

2. MSCONS – Meter Reading Schedule Consumption (Messwerte)

Zweck: Übermittlung von Verbrauchsdaten zwischen Marktpartnern

Zentrale Segmente:

- **UNH, BGM**: Nachrichtenkopf
- **DTM**: Messperiode (Start/Ende)
- **RFF**: Referenzen (Marktlokation, Geschäftsvorfall)
- **NAD**: Beteiligte Partner
- **LOC**: Messlokation
- **QTY**: Messwerte mit Einheit
- **SEQ**: Zeitreihen-Sequenzen
- **MEA**: Detaillierte Messwerte (oft Viertelstundenwerte)

Code-Beispiel:

```
const json = transformer.transform(msconsString);

// Messwerte auslesen
json.body.messwerte.messwerte.forEach(messwert => {
  console.log(`
    Qualifier: ${messwert.qualifier}
    Wert: ${messwert.value} ${messwert.unit}
    Gültig: ${messwert.valid_unit}
  `);
});

// Zeitreihen-Daten
console.log('Anzahl Zeitscheiben:', json.body.messwerte.zeitreihen.length);

// Messperiode
const periode = json.body.messwerte.messperioden;
console.log(`Von ${periode[0].datetime} bis ${periode[1].datetime}`);
```

Wichtige Qualifier:

- `220`: Gelieferte Energie
- `66`: Zählerstand
- `74`: Bezogene Energie

Einheiten:

- `KWH`: Kilowattstunde
- `MWH`: Megawattstunde
- `W`: Watt (Leistung)

Typische Anwendungsfälle:

- Bilanzierungsrelevante Messwerte (MaBiS)
- Abrechnungsdaten für Lieferanten
- Austausch zwischen MSB und NB/LF

---

3. ORDERS – Purchase Order (Bestellung)

Zweck: Elektronische Übermittlung von Bestellungen und Anfragen

Zentrale Segmente:

- **UNH, BGM**: Nachrichtenkopf
- **DTM**: Bestelldatum, Liefertermin
- **RFF**: Bestellnummer, Referenzen
- **NAD**: Besteller und Empfänger
- **LIN**: Bestellpositionen
- **PIA**: Produktidentifikation
- **IMD**: Artikelbeschreibung
- **QTY**: Bestellmengen

Code-Beispiel:

```
const json = transformer.transform(ordersString);


// Bestellpositionen auslesen
json.body.bestellung.order_lines.forEach(line => {
  console.log(`
    Position: ${line.line_number}
    Artikel-ID: ${line.item_id}
    Typ: ${line.item_type}
  `);
});
```

Typische Anwendungsfälle:

- Beauftragung Messstellenbetrieb (WiM)
- Anfrage Gasbeschaffenheit (GeLi Gas)
- Materialbestellungen

---

4. ORDRSP – Purchase Order Response (Bestellantwort)

Zweck: Antwort auf ORDERS-Nachricht

Zentrale Segmente:

- **UNH, BGM**: Nachrichtenkopf
- **DTM**: Antwortdatum
- **RFF**: Referenz auf ORDERS
- **NAD**: Absender/Empfänger
- **STS**: Bestellstatus
- **LIN, PIA, IMD**: Positionsdaten

Statuswerte:

- `7`: Bestätigt
- `27`: Abgelehnt
- `4`: Teilweise bestätigt

---

5. INVOIC – Invoice (Rechnung)

Zweck: Übermittlung von Rechnungen und Gutschriften

Zentrale Segmente:

- **UNH, BGM**: Nachrichtenkopf
- **DTM**: Rechnungs-/Leistungsdatum
- **RFF**: Rechnungs-/Bestellnummer
- **NAD**: Rechnungssteller/-empfänger
- **LIN, PIA, IMD**: Rechnungspositionen
- **MOA**: Geldbeträge
- **TAX**: Steuerdetails

Code-Beispiel:

```
const json = transformer.transform(invoicString);


// Rechnungssumme
const summe = json.body.rechnung.totals['77'];
console.log(`Gesamt: ${summe.amount} ${summe.currency}`);
```

```
// Steuern
json.body.rechnung.tax_amounts.forEach(tax => {
  console.log(`Steuer: ${tax.type}, Rate: ${tax.rate}%`);
});
```

MOA-Qualifier:

- `77`: Rechnungssumme
- `79`: Gesamtbetrag inkl. MwSt
- `125`: Steuerbetrag

---

6. APERAK – Application Error and Acknowledgement

Zweck: Bestätigung oder Fehlermeldung für empfangene Nachrichten

Zentrale Segmente:

- **UNH, BGM**: Nachrichtenkopf
- **DTM**: Erstellungsdatum
- **RFF**: Referenz auf Original-Nachricht
- **ERC**: Fehlerinformationen
- **FTX**: Freitext-Erklärungen

Code-Beispiel:

```
const json = transformer.transform(aperakString);

// Status prüfen
if (json.body.quittierung.status === 'positive') {
  console.log('Nachricht erfolgreich verarbeitet');
} else {
  // Fehler ausgeben
  json.body.quittierung.errors.forEach(error => {
    console.error(`Fehler ${error.code}: ${error.description}`);
  });
}
```

Häufige Fehlercodes:

- `1`: Syntax-Fehler
- `2`: Semantischer Fehler
- `3`: Geschäftsprozess-Fehler
- `7`: Nachricht akzeptiert

□□ Prüfidentifikatoren verstehen (RFF+Z13)

Was ist ein Prüfidentifikator?

Der **Prüfidentifikator** (im Segment `RFF+Z13`) ist die **wichtigste Kennung** in einer MaKo-Nachricht. Er identifiziert eindeutig den Geschäftsvorfall und bestimmt:

- Welcher Prozess läuft (Anmeldung, Abmeldung, etc.)
- Welche Segmente Pflicht sind
- Welche Validierungsregeln gelten
- Welche Antwort-Nachrichten zu erwarten sind

Format:

```
RFF+Z13:44001'
     │   │
     │   └─ Prüfidentifikator (5-stellig)
     └─ Qualifier Z13
```

Die wichtigsten Prüfidentifikatoren

GPKE (Strom)

Prüf-ID	Prozess	Richtung	Typische Antwort
44001	Anmeldung Netznutzung	Lieferant → NB	APERAK (44002/44003)
44002	Bestätigung Anmeldung	NB → Lieferant	-
44003	Ablehnung Anmeldung	NB → Lieferant	-
44004	Abmeldung Netznutzung	Lieferant → NB	APERAK (44005/44006)
44005	Bestätigung Abmeldung	NB → Lieferant	-

Prüf-ID	Prozess	Richtung	Typische Antwort
44006	Ablehnung Abmeldung	NB → Lieferant	-
44016	Kündigung beim alten LF	NB → Lieferant	APERAK (44017/44018)
44017	Bestätigung Kündigung	Lieferant → NB	-
44018	Ablehnung Kündigung	Lieferant → NB	-
44112	Stammdatenänderung	NB → Lieferant	-
44123	Bila.rel. Änderung	NB → Lieferant	-

Messwerte

Prüf-ID	Prozess	Verwendung
13002	Zählerstand	MSCONS
13008	Lastgang	MSCONS
13009	Energiemenge	MSCONS
13007	Gasbeschaffenheit	MSCONS (Gas)
13006	Messwert Storno	MSCONS

Stammdaten-Anfragen

Prüf-ID	Prozess	Verwendung
17101	Anfrage Stammdaten MaLo	ORDERS
17102	Anfrage Werte	ORDERS
19101	Ablehnung Anfrage Stammdaten	ORDRSP
19102	Ablehnung Anfrage Werte	ORDRSP

WiM (Messwesen)

Prüf-ID	Prozess	Verwendung
17009	Gerätewechsel	ORDERS
19015	Bestätigung Gerätewechsel	ORDRSP
21039	Auftragsstatus Sperren	IFTSTA
21040	Info Entsperrauftrag	IFTSTA

Rechnungen

Prüf-ID	Prozess	Verwendung
31001	Abschlagsrechnung	INVOIC
31002	NN-Rechnung	INVOIC
31003	WiM-Rechnung	INVOIC
31004	Stornorechnung	INVOIC
33001	Bestätigung	REMADV
33002	Abweisung	REMADV

Code-Beispiel: Prüfidentifikator auswerten

```
const json = transformer.transform(edifactString);
const pruefId = json.metadata.pruefidentifikator.id;

switch(pruefId) {
  case '44001':
    console.log('Anmeldung zur Netznutzung');
    // Validiere Pflichtfelder für Anmeldung
    validateAnmeldung(json);
    break;

  case '44004':
    console.log('Abmeldung vom Netz');
    validateAbmeldung(json);
    break;

  case '13008':
    console.log('Lastgangdaten');
    processLastgang(json.body.messwerte);
    break;

  default:
    console.warn(`Unbekannte Prüf-ID: ${pruefId}`);
}
```

Helper-Funktion

```
const { isGPKEProcess } = require('edifact-json-transformer');

// Prozess-Zuordnung prüfen
if (isGPKEProcess(edifactString)) {
  console.log('GPKE-Prozess erkannt');
}

// Alle Prüfidentifikatoren verfügbar
const { pruefidentifikatoren } = require('edifact-json-transformer');
console.log(pruefidentifikatoren['44001']);
// → "Anmeldung NN"
```

□□ Praktische Tipps für den Alltag

1. Schnell-Check: Was ist das für eine Nachricht?

```
// Nur Metadaten ohne vollständige Transformation
const { EdifactTransformer } = require('edifact-json-transformer');

function quickInfo(edifactString) {
  const transformer = new EdifactTransformer({
    validateStructure: false,  // Schneller
    parseTimestamps: false
  });

  const json = transformer.transform(edifactString);

  return {
    typ: json.metadata.message_type,
    name: json.metadata.message_name,
    pruefId: json.metadata.pruefidentifikator?.id,
    prozess: json.metadata.applicable_processes,
    absender: json.parties.sender?.id,
    empfaenger: json.parties.receiver?.id
```

```
  };
}

console.log(quickInfo(edifactString));
// { typ: 'UTILMD', name: 'Stammdaten', pruefId: '44001', ... }
```

2. Marktlokationen schnell extrahieren

```
const { extractAllMarktlokationIds } = require('edifact-json-transformer');

// Super-schnelle Extraktion
const maloIds = extractAllMarktlokationIds(edifactString);
console.log(maloIds); // ['12345678901', '98765432109']

// In der Datenbank nachschlagen
maloIds.forEach(id => {
  const kunde = db.findByMaloId(id);
  console.log(`Kunde: ${kunde.name} (${id})`);
});
```

3. Zeitscheiben in MSCONS verarbeiten

```
function processTimeSeries(msconsString) {
  const json = transformer.transform(msconsString);
  const zeitreihen = json.body.messwerte.zeitreihen;

  // Viertelstundenwerte
  const viertelstundenwerte = zeitreihen.map((seq, idx) => {
    const messwert = json.body.messwerte.messwerte[idx];
    return {
      sequenz: seq.sequence_number,
      wert: messwert.value,
      einheit: messwert.unit,
      zeitpunkt: calculateTimeSlot(seq.sequence_number, startDate)
    };
  });
```

```
  return viertelstundenwerte;
}

function calculateTimeSlot(sequenz, startDate) {
  // Viertelstunde = 15 Minuten
  const minutes = (sequenz - 1) * 15;
  return new Date(startDate.getTime() + minutes * 60000);
}
```

4. Validierung mit Fehler-Report

```
const { validateAHB } = require('edifact-json-transformer');

function validateAndReport(edifactString) {
  const report = validateAHB(edifactString);

  if (!report.is_valid) {
    console.error('⚠ Validierungsfehler gefunden:');

    report.errors.forEach(error => {
      console.error(`  [${error.category}] ${error.message}`);
    });

    report.warnings.forEach(warning => {
      console.warn(`  ⚠ [${warning.category}] ${warning.message}`);
    });

    return false;
  }

  console.log('✓ Nachricht ist valide');
  return true;
}
```

5. Datumswerte lesen

```
const json = transformer.transform(edifactString);

// Alle Datumswerte
console.log(json.dates);
// {
//   message_date: '2024-10-19',
//   start_date: '2024-01-01',
//   end_date: '2024-12-31'
// }

// Spezifische Prüfungen
const { start_date, end_date } = json.dates;

if (new Date(start_date) > new Date(end_date)) {
  console.error('Start liegt nach Ende!');
}

// Dauer berechnen
const days = (new Date(end_date) - new Date(start_date)) / (1000 * 60 * 60 * 24);
console.log(`Zeitraum: ${days} Tage`);
```

6. Graph-Beziehungen für Analyse

```
const transformer = new EdifactTransformer({
  generateGraphRelations: true
});

const json = transformer.transform(edifactString);

// Neo4j Cypher generieren
const { convertToNeo4jCypher } = require('edifact-json-transformer');
const statements = convertToNeo4jCypher(edifactString);

statements.forEach(stmt => {
  console.log(stmt.cypher);
  console.log(stmt.parameters);
});
```

```
// Oder direkt aus JSON
json.graph_relations.forEach(rel => {
  console.log(`${rel.from.type}:${rel.from.id} -[${rel.relationship}]->
${rel.to.type}:${rel.to.id}`);
});
```

7. Batch-Verarbeitung

```
const fs = require('fs');
const { EdifactTransformer } = require('edifact-json-transformer');

function processBatch(directory) {
  const transformer = new EdifactTransformer({
    enableAHBValidation: true
  });

  const files = fs.readdirSync(directory);
  const results = {
    success: 0,
    errors: [],
    warnings: []
  };

  files.forEach(file => {
    try {
      const content = fs.readFileSync(`${directory}/${file}`, 'utf8');
      const json = transformer.transform(content);

      if (json.validation?.is_valid !== false) {
        results.success++;
      } else {
        results.errors.push({
          file,
          errors: json.validation.errors
        });
      }

      if (json.validation?.warnings?.length > 0) {
        results.warnings.push({
```

```
          file,
          warnings: json.validation.warnings
        });
      }
    } catch (error) {
      results.errors.push({ file, error: error.message });
    }
  });

  console.log(`
    ✅ Erfolgreich: ${results.success}
    ❌ Fehler: ${results.errors.length}
    ⚠️ Warnungen: ${results.warnings.length}
  `);

  return results;
}
```

🔧 Häufige Fehler und Lösungen

1. Syntaxfehler

Problem: Falsche Trennzeichen

```
// ❌ Falsch
"UNH+1+UTILMD:D:11A:UN:2.6'BGM+E01+12345+9'..."  // Fehlt Zeilenumbruch-Escape

// ✅ Richtig
const edifact = normalizeEdifact(rawString);  // Transformer macht das automatisch
```

Problem: Fehlende Pflichtsegmente

```
const json = transformer.transform(edifactString);

if (json.validation?.errors) {
  json.validation.errors.forEach(error => {
```

```
    if (error.message.includes('Fehlendes UNH')) {
      console.error('Nachrichtenkopf fehlt!');
    }
  });
}
```

Lösung: Prüfe die AHB-Vorgaben für den jeweiligen Nachrichtentyp.

Problem: Falsche Datenformate

```
// DTM-Segment mit falschem Format
"DTM+137:19.10.2024:102'"   // ❌ Falsch: TT.MM.JJJJ

"DTM+137:20241019:102'"     // ✅ Richtig: JJJJMMTT
```

Debugging:

```
const json = transformer.transform(edifactString, {
  includeRawSegments: true
});

// Rohe Segmente inspizieren
json.raw_segments.filter(s => s.tag === 'DTM').forEach(dtm => {
  console.log('DTM:', dtm.raw);
});
```

---

2. Semantische Fehler

Problem: Referenznummer-Mismatch

```
// ❌ UNH und UNT stimmen nicht überein
"UNH+1+UTILMD:D:11A:UN:2.6'...UNT+7+2'"

// ✅ Korrekt
"UNH+1+UTILMD:D:11A:UN:2.6'...UNT+7+1'"
```

Automatische Prüfung:

```
const json = transformer.transform(edifactString);

if (json.validation?.errors.some(e => e.message.includes('Referenznummer-Mismatch'))) {
  console.error('UNH/UNT Referenzen stimmen nicht überein!');
}
```

Problem: Ungültige MP-ID

```
const json = transformer.transform(edifactString);

Object.entries(json.parties).forEach(([role, party]) => {
  if (!party.valid_mp_id) {
    console.error(`
      ⚠ Ungültige MP-ID für ${role}
      Wert: ${party.id}
      Erwartet: 13-stellig numerisch
    `);
  }
});
```

Validierung:

```
function isValidMpId(id) {
  return /^\d{13}$/.test(id);
}
```

Problem: Start-/Enddatum vertauscht

```
const json = transformer.transform(edifactString);

if (json.validation?.errors) {
  json.validation.errors.forEach(error => {
    if (error.message.includes('Start-Datum liegt nach End-Datum')) {
      console.error('Zeitlogik ist falsch!');
      console.log('Start:', json.dates.start_date);
      console.log('Ende:', json.dates.end_date);
    }
  });
}
```

3. Geschäftsprozess-Fehler

Problem: Falscher Prüfidentifikator

```
const json = transformer.transform(edifactString);
const pruefId = json.metadata.pruefidentifikator?.id;

// Prüfen ob Prüf-ID zum Nachrichtentyp passt
const validCombinations = {
  'UTILMD': ['44001', '44002', '44003', '44004', '44005', '44006', '44016', '44112', '44123'],
  'MSCONS': ['13002', '13007', '13008', '13009'],
  'ORDERS': ['17009', '17101', '17102'],
  'INVOIC': ['31001', '31002', '31003', '31004']
};

const messageType = json.metadata.message_type;
if (!validCombinations[messageType]?.includes(pruefId)) {
  console.error(`
    ❌ Prüf-ID ${pruefId} passt nicht zu ${messageType}
    Erlaubte IDs: ${validCombinations[messageType].join(', ')}
  `);
}
```

Problem: Fehlende Rollen

```
// Für Anmeldung (44001) sind spezifische Rollen erforderlich
if (json.metadata.pruefidentifikator.id === '44001') {
  const hasLieferant = json.parties.lieferant || json.parties.sender;
  const hasNetzbetreiber = json.parties.netzbetreiber || json.parties.receiver;

  if (!hasLieferant) {
    console.error('❌ Lieferant-Rolle fehlt für Anmeldung');
  }

  if (!hasNetzbetreiber) {
    console.error('❌ Netzbetreiber-Rolle fehlt für Anmeldung');
  }
}
```

Problem: Ungültige Marktlokations-ID

```
const json = transformer.transform(edifactString);

json.body.stammdaten?.marktlokationen?.forEach((malo, idx) => {
  if (!malo.valid) {
    console.error(`
      ⚠️ Marktlokation ${idx + 1} ungültig
      ID: ${malo.id}
      Erwartet: 11-stellig
      Tatsächlich: ${malo.id?.length} Zeichen
    `);
  }
});

json.body.stammdaten?.messlokationen?.forEach((melo, idx) => {
  if (!melo.valid) {
    console.error(`
      ⚠️ Messlokation ${idx + 1} ungültig
      ID: ${melo.id}
      Erwartet: 33-stellig
      Tatsächlich: ${melo.id?.length} Zeichen
    `);
  }
});
```

🎯 Best Practices für Entwickler

1. Validierung auf allen Ebenen

```
class EdifactProcessor {
  constructor() {
    this.transformer = new EdifactTransformer({
      enableAHBValidation: true,
      validateStructure: true,
      validateBusinessRules: true
```

```
  });
}

async process(edifactString) {
  // 1. Syntax-Validierung
  if (!this.validateSyntax(edifactString)) {
    throw new Error('Syntax ungültig');
  }

  // 2. Transformation
  const json = this.transformer.transform(edifactString);

  // 3. Struktur-Validierung
  if (!json.validation?.is_valid) {
    this.logErrors(json.validation.errors);
    throw new Error('Struktur ungültig');
  }

  // 4. Geschäftsprozess-Validierung
  this.validateBusinessLogic(json);

  // 5. Verarbeitung
  return this.processMessage(json);
}

validateSyntax(edifactString) {
  // Grundlegende Checks
  return edifactString.includes("UNH") &&
         edifactString.includes("UNT");
}

validateBusinessLogic(json) {
  const pruefId = json.metadata.pruefidentifikator?.id;

  // Prozessspezifische Validierung
  switch(pruefId) {
    case '44001':
      this.validateAnmeldung(json);
      break;
```

```
      case '13008':
        this.validateLastgang(json);
        break;
    }
  }
}
```

2. Umfassendes Error Handling

```
class MessageValidator {
  validate(json) {
    const errors = [];
    const warnings = [];

    // Pflichtfelder prüfen
    if (!json.metadata.pruefidentifikator) {
      errors.push({
        category: 'AHB',
        message: 'Prüfidentifikator fehlt',
        severity: 'ERROR'
      });
    }

    // Marktlokationen prüfen
    json.body.stammdaten?.marktlokationen?.forEach((malo, idx) => {
      if (!malo.valid) {
        warnings.push({
          category: 'STAMMDATEN',
          message: `MaLo ${idx + 1}: ${malo.id} ist ungültig`,
          severity: 'WARNING'
        });
      }
    });

    return {
      is_valid: errors.length === 0,
      errors,
      warnings
```

```
    };
  }

  generateAPERAK(originalMessage, errors) {
    // APERAK-Nachricht generieren
    return `
      UNH+${generateRef()}+APERAK:D:11A:UN:2.0'
      BGM+${errors.length > 0 ? '27' : '7'}+${generateRef()}+9'
      ${errors.map(e => `ERC+${e.code}'`).join('')}
      UNT+${2 + errors.length}+${generateRef()}'
    `;
  }
}
```

3. Strukturierte Logging-Strategie

```
const winston = require('winston');

const logger = winston.createLogger({
  level: 'info',
  format: winston.format.json(),
  transports: [
    new winston.transports.File({ filename: 'edifact-error.log', level: 'error' }),
    new winston.transports.File({ filename: 'edifact-combined.log' })
  ]
});

function processMessage(edifactString, source) {
  const startTime = Date.now();

  try {
    const json = transformer.transform(edifactString);

    logger.info({
      event: 'message_processed',
      type: json.metadata.message_type,
      pruefId: json.metadata.pruefidentifikator?.id,
      source,
```

```
      duration: Date.now() - startTime,
      validation: json.validation?.is_valid
    });

    return json;

  } catch (error) {
    logger.error({
      event: 'processing_failed',
      source,
      error: error.message,
      stack: error.stack,
      duration: Date.now() - startTime
    });
    throw error;
  }
}
```

4. Modularer Aufbau

```
// parser.js
class EdifactParser {
  parse(edifactString) {
    return transformer.transform(edifactString);
  }
}

// validator.js
class EdifactValidator {
  validateAHB(json) { /* ... */ }
  validateStructure(json) { /* ... */ }
  validateBusinessRules(json) { /* ... */ }
}

// processor.js
class MessageProcessor {
  processUTILMD(json) { /* ... */ }
  processMSCONS(json) { /* ... */ }
```

```
  processORDERS(json) { /* ... */ }
}

// main.js
class EdifactService {
  constructor() {
    this.parser = new EdifactParser();
    this.validator = new EdifactValidator();
    this.processor = new MessageProcessor();
  }

  async handleMessage(edifactString) {
    const json = this.parser.parse(edifactString);

    if (!this.validator.validateAHB(json)) {
      throw new ValidationError('AHB-Validierung fehlgeschlagen');
    }

    const messageType = json.metadata.message_type;
    return this.processor[`process${messageType}`](json);
  }
}
```

5. Testing-Strategie

```
// test/utilmd.test.js
const { EdifactTransformer } = require('edifact-json-transformer');

describe('UTILMD Transformation', () => {
  let transformer;

  beforeEach(() => {
    transformer = new EdifactTransformer({
      enableAHBValidation: true
    });
  });

  test('Anmeldung NN (44001) korrekt verarbeitet', () => {
```

```
    const edifact = loadTestFile('utilmd_anmeldung_44001.txt');
    const json = transformer.transform(edifact);

    expect(json.metadata.message_type).toBe('UTILMD');
    expect(json.metadata.pruefidentifikator.id).toBe('44001');
    expect(json.body.stammdaten.marktlokationen).toHaveLength(1);
    expect(json.validation.is_valid).toBe(true);
  });

  test('Ungültige MaLo-ID wird erkannt', () => {
    const edifact = loadTestFile('utilmd_invalid_malo.txt');
    const json = transformer.transform(edifact);

    expect(json.validation.is_valid).toBe(false);
    expect(json.validation.errors).toContainEqual(
      expect.objectContaining({
        message: expect.stringContaining('11-stellig')
      })
    );
  });

  test('Fehlender Prüfidentifikator wird gemeldet', () => {
    const edifact = loadTestFile('utilmd_no_pruefid.txt');
    const json = transformer.transform(edifact);

    expect(json.validation.errors).toContainEqual(
      expect.objectContaining({
        message: expect.stringContaining('Prüfidentifikator')
      })
    );
  });
});
```

6. Versionierung und Updates

```
class EdifactService {
  constructor() {
    this.transformers = {
```

```
      'v2.6': new EdifactTransformer({ /* config für 2.6 */ }),
      'v2.7': new EdifactTransformer({ /* config für 2.7 */ })
    };
  }

  getTransformer(version) {
    return this.transformers[`v${version}`] || this.transformers['v2.6'];
  }

  process(edifactString) {
    // Version aus UNH extrahieren
    const versionMatch = edifactString.match(/UNH\+\d+\+[^:]+:[^:]+:[^:]+:[^:]+:([^']+)'/);
    const version = versionMatch ? versionMatch[1] : '2.6';

    const transformer = this.getTransformer(version);
    return transformer.transform(edifactString);
  }
}
```

7. Performance-Optimierung

```
// Caching für wiederholte Transformationen
const NodeCache = require('node-cache');
const cache = new NodeCache({ stdTTL: 600 });

function transformWithCache(edifactString) {
  const hash = crypto.createHash('md5').update(edifactString).digest('hex');

  let json = cache.get(hash);
  if (json) {
    console.log('Cache hit');
    return json;
  }

  json = transformer.transform(edifactString);
  cache.set(hash, json);
  return json;
}
```

```
// Bulk-Processing mit Worker Threads
const { Worker } = require('worker_threads');

async function processBulk(messages) {
  const chunkSize = Math.ceil(messages.length / 4);
  const workers = [];

  for (let i = 0; i < messages.length; i += chunkSize) {
    const chunk = messages.slice(i, i + chunkSize);
    workers.push(
      new Promise((resolve) => {
        const worker = new Worker('./edifact-worker.js', {
          workerData: chunk
        });
        worker.on('message', resolve);
      })
    );
  }

  const results = await Promise.all(workers);
  return results.flat();
}
```

□□ Weiterführende Ressourcen

Offizielle Quellen

1. **EDI@Energy Portal** (BDEW)
   - URL: https://www.edi-energy.de/
   - Registrierung erforderlich
   - Enthält: AHB, MIG, Codelisten, Prozessbeschreibungen
2. **Bundesnetzagentur**
   - Festlegungen zu GPKE, GeLi Gas, WiM, MaBiS
   - URL: https://www.bundesnetzagentur.de/
3. **BDEW Codelisten**
   - Aktuelle Qualifier und Codes
   - Updates bei Prozessänderungen

Tools und Plattformen

1. **Willi Mako** – Intelligente MaKo-Plattform
   - Web-App: https://stromhaltig.de/app/
   - Open-Source Client: https://github.com/energychain/willi-mako-client
   - Regulatorisches Wissen, Beispiele, Validierung
2. **edifact-json-transformer**
   - Repository: https://github.com/energychain/edifact-to-json-transformer
   - NPM: `npm install edifact-json-transformer`
   - MIT-Lizenz, Community-Support

Lernressourcen

- **EDIFACT ISO 9735 Standard**: Grundlagen zu Segmenten und Syntax
- **BDEW Schulungen**: Workshops zu Marktkommunikation
- **Online-Communities**: Austausch mit anderen Entwickler:innen

Support

- **Issues/Bugs**: GitHub Issues
- **Pull Requests**: Contributions willkommen!
- **Maintainer**: STROMDAO GmbH (https://stromdao.de/)

□□ Zusammenfassung

Du hast jetzt gelernt:

□**Grundlagen**: Was EDIFACT ist und warum es in der Energiewirtschaft genutzt wird
□**Prozesse**: GPKE, GeLi Gas, WiM und MaBiS im Detail
□**Nachrichtentypen**: UTILMD, MSCONS, ORDERS, INVOIC, APERAK
□**Prüfidentifikatoren**: Die wichtigsten IDs und ihre Bedeutung
□**Praktische Tools**: Wie du mit dem Transformer effizient arbeitest
□**Fehlerbehandlung**: Häufige Probleme erkennen und lösen
□**Best Practices**: Professionelle Entwicklung von MaKo-Software

Nächste Schritte

1. **Installiere** den Transformer: `npm install edifact-json-transformer`
2. **Probiere** die Beispiele aus diesem Guide
3. **Validiere** echte Nachrichten aus deinem Projekt
4. **Erkunde** Willi Mako für tieferes regulatorisches Wissen
5. **Teile** deine Erfahrungen und trage zur Community bei

Denk daran

> "EDIFACT ist komplex, aber mit den richtigen Tools und etwas Übung wirst du zum MaKo-Profi!"

Viel Erfolg bei deinen ersten (oder nächsten) Schritten in der Marktkommunikation!

Lizenz: MIT

Maintainer: STROMDAO GmbH

Powered by: Willi Mako – Intelligente Marktkommunikation

Veröffentlicht auf: Corrently.io

Letzte Aktualisierung: Oktober 2024