Hintergrundwissen:

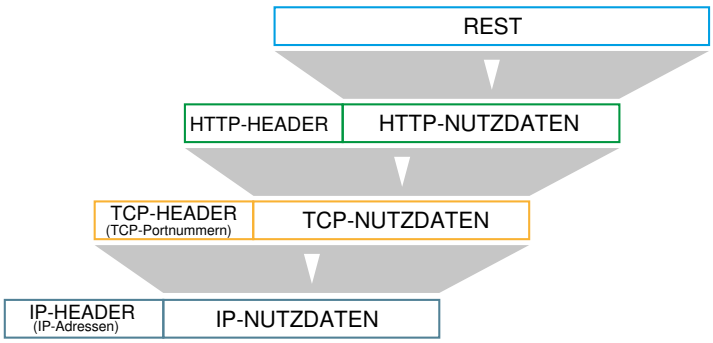
# REST - REpresentational State Transfer

**Industrielle Kommunikation auf Basis von standardisierten HTTP-Requests**

REST beschreibt keinen konkreten Protokollaufbau. Stattdessen gibt REST Eigenschaften vor, die für einen Datenaustausch im industriellen Umfeld erfüllt sein sollten.

## REST - Übertragung auf Netzwerkebene

Vorab sei gesagt, dass REST als übergeordnetes Protokoll HTTP bzw. HTTPS nutzt.




Der Vorteil von HTTP(S) als Kommunikationsgrundlage liegt darin, dass die meisten Netzwerke, auch wenn sie durch Firewalls usw. geschützt sind, für HTTP(s) durchgängig nutzbar sind. Außerdem wird HTTP(S) den für REST geforderten Eigenschaften gerecht.

Genutzt werden, wie bei HTTP bzw. HTTPS üblich, die TCP-Ports 80 bzw. 443.

## REST - Eigenschaften und Grundelemente

**Client/Server-Modell**
REST arbeitet nach dem Client/Server-Verfahren, wobei die Aktion immer vom Client angestoßen wird. Der Server stellt Ressourcen (Daten, Inhalte, Funktionen) zur Verfügung. Der Client sendet einen Request, um auf ausgewählte Ressourcen zuzugreifen. Der Server liefert mit einem Reply die angeforderten Daten oder bestätigt die gewünschte Aktion.

**Zustandslosigkeit**
Bei vielen Client/Server-Anwendungen wird beim Verbindungsaufbau serverseitig ein bestimmter Status (Berechtigung, spezielle Aufgabe, spezifischer Zweck, ...) zugewiesen, der für die Dauer der Verbindung erhalten bleibt. REST behandelt alle Sendungen zum Server zunächst gleich und erst der Inhalt der Datensendung bestimmt den weiteren serverseitigen Umgang bzw. die Einordnung des Requests. Der Status einer Anwendung liegt also in den Händen des Clients.

**Geschichtetes System**
REST sieht eine klare Trennung von Zuständigkeiten vor. Diese Trennung gilt insbesondere für die Abwicklung der Kommunikation und die Weiterverarbeitung der transportierten Inhalte.

Dadurch ist es möglich, Proxies, Gateways oder ähnliche Zwischenstationen auf dem Übertragungsweg zu nutzen.

Außerdem kann je nach Sicherheitsanspruch HTTP oder die verschlüsselte Übertragung per HTTPS für die Kommunikation gewählt werden. Für die funktionale Abwicklung der REST-Inhalte macht das keinen Unterschied.

**Adressierbarkeit der Ressourcen**
Alle auf einem Server verfügbaren Ressourcen sind über eine eindeutige Adresse (URI = Uniform Resource Identifier) abrufbar. Der URI ist aufgebaut wie die URL, die für die Adressierung im Browser verwendet wird:

`Protokoll://<Host>:<Port>/<Pfad>/<Ressource>?<Parameter>&<Parameter>`

**Anfrage-Methoden**
Für die Requests stehen die standardisierten HTTP-Aufrufe zur Verfügung:

- **GET**
  ruft Ressourcen vom Server ab und wird ausschließlich lesend genutzt
- **POST**
  legt neue Ressourcen an oder ändert bestehende
- **PUT**
  ändert bestehende Ressourcen
- **DELETE**
  löscht bestehende Ressourcen
- **HEAD**
  fordert nur den HTTP-Kopf an, um z. B. die Verfügbarkeit der abzurufenden Ressource zu prüfen

- **OPTIONS**
  ruft Informationen zu den Kommunikationsoptionen ab

Bei den abgerufenen Daten spricht man von Repräsentationen einer oder mehrerer Ressourcen. Daher die Bezeichnung REST für REpresentational State Transfer.

Letztlich sind Repräsentationen nichts anderes als ein Abbild von Teilen der Prozessdaten. Die Übergabe kann in nahezu beliebiger, aber zu vereinbarender Form erfolgen. Üblich sind JSON, XML oder roher Text. Aber auch SVG, MP3 oder andere Formate werden abhängig von der Anwendung genutzt. Die Repräsentation kann auch Hyperlinks auf weitere Ressourcen beinhalten.

Der Client sendet:

`http://192.168.1.19/rest/json/iostate/`

Und das Web-Thermo-Hygrobarometer sendet als Antwort:

```
{
  "iostate": {
    "sensor": [
      {
        "name": "Temperatur",
        "number": 0,
        "unit": "°C",
        "value": 25.9
      },
      {
        "name": "rel. Feuchte",
        "number": 1,
        "unit": "%",
        "value": 43.2
      },
      {
        "name": "Luftdruck",
        "number": 2,
        "unit": "hPa",
        "value": 994.7
      }
    ]
  }
}
```

**Code on Demand**
Bei Webseiten ist es heute üblich, z. B. JavaScript mit- bzw. nachzuladen. Auch REST erlaubt, Quellcode, Programmteile oder Funktionsbausteine nachzuladen. Damit kann zur Laufzeit die Funktion des Clients erweitert oder verändert werden.

**Cache**
REST sieht vor, wiederholende Requests clientseitig aus einem Cache (Zwischenspeicher) zu beantworten, um die Datenlast auf den Übertragungswegen zu reduzieren. Der Server legt über entsprechende Angaben im HTTP-Kopf fest, ob für die angeforderte Repräsentation der Cache genutzt werden darf oder nicht.

**Vorteile von REST**
Einfache Implementierung, da weitestgehend die Mechanismen von HTTP genutzt werden.

**Nachteile von REST**
Durch die Request-/Reply-Technik ist nur Pollen, also das gezielte Abrufen von Daten, aber keine eventgesteuerte Kommunikation möglich. Es müssen die relevanten Daten kontinuierlich gepollt (abgefragt) werden, um Veränderungen zu erkennen.


### Grundlagen zu gängigen Industrieprotokollen

Web-IO - Automatisierung mit Standardprotokollen
IO-Signale mit Modbus-TCP, OPC UA/DA, Rest oder MQTT integrieren

Datenformate und Protokolle
Vom klassischer Datenaustausch bis IoT

Modbus-TCP
Offener Standard für industrielle Kommunikation

MQTT - Message Queue Telemetry Protocol
Datenaustausch ohne direkte Verbindung

OPC UA - OPC Unified Architecture
OPC Unterstützung out of the box

OPC DA - OPC Data Access
Der Prozessdaten-Dolmetscher

**Produkte für industrielle Anwendungen mit Standardprotokollen**

Web-IO 4.0 Digital
2xIn, 2xOut

Bei Bedarf auch über PoE zu versorgen

Web-IO 4.0 Digital
12xIn, 12xOut

12x Ausgänge (6-30V),
12x Eingänge (8-30V)

Weitere Web-IOs

Alle W&T Web-IO Digital 24V

W&T
www.WuT.de

Wir sind gerne persönlich für Sie da:

Wiesemann & Theis
GmbH
Porschestr. 12
42279 Wuppertal
Tel.: 0202/2680-110 (Mo-Fr. 8-17
Uhr)
Fax: 0202/2680-265
info@wut.de



Datenschutz