MeterData API

Inhalte

Überblick

Die MeterData API stelt eine einfache API dar, mit der Messwerte aus dem eFRIENDS meter, die im Cube gespeichert werden, abgefragt werden können. Grundsätzlich wird zwischen Abfrage (Pull) und Push-Verfahren unterschieden. Für das Pull Verfahren steht eine REST API für die Formate JSON und XML zur Verfügung. Das Push Verfahren wird mittels Websockets (auf Basis socket.io) unterstützt.

REST API

Es ist eine REST API direkt am Cube bereitgestellt, die das Abfragen von Messwerten ermöglicht.

Generell

Für alle Abfragen gilt, dass das Ergebnis in den Formaten JSON und XML geliefert werden kann. Um das Format anzufordern, ist der URL Query-Parameter format mit den möglichen Werte "json" oder "xml" anzugeben. Wird kein "format" angegeben, wird JSON ausgeliefert.

Authentifizierung

Alle API Zugriffe sind geschützt und müssen mit einem HTTP Header **apiKey** bzw. bei WebSocket Connections mit einem token Header authentifiziert werden. Die Tokens können in der App unter "API Manager" verwaltet werden.

Abfrage der aktuellen Messung (getCurrentValue)

Die API ist hier verfügbar:

Protokoll	URL
HTTP	http://<cubeIp>/api/MeterData/getCurrentValue

Parameter

Name	Wert	Beschreibung
format	json | xml	Gewünschtes Ausgabeformat. Default = json

Beispielaufruf

```
# Explizite Angabe des Formats xml

Fritzs-MacBook-Pro-4:~ fritz$ curl --location http://192.168.1.174/api/MeterData/getCurrentValue?format=xml --
header 'apiKey: myCustomAPIKey'
<?xml version="1.0" encoding="utf-8"?>
<energyData>
  <endTime>2018-06-19T22:57:40.000Z</endTime>
  <startTime>2018-06-19T22:57:30.000Z</startTime>
  <energyBalance>-2204.75</energyBalance>
  <details>
    <power1Watt>-325.25</power1Watt>
    <power2Watt>-1795.5</power2Watt>
    <power3Watt>-84</power3Watt>
    <current1Ampere>2.3</current1Ampere>
    <current2Ampere>8.1</current2Ampere>
    <current3Ampere>1.4</current3Ampere>
    <voltage1Volt>229.29999999999998</voltage1Volt>
    <voltage2Volt>229.75</voltage2Volt>
    <voltage3Volt>230.75</voltage3Volt>
  </details>
</energyData>


# Beispiel bei keiner Angabe des Formats (Standard = JSON)

Fritzs-MacBook-Pro-4:~ fritz$ curl --location http://192.168.1.174/api/MeterData/getCurrentValue --header
'apiKey: myCustomAPIKey'
{"endTime":"2018-06-19T22:57:40.000Z","startTime":"2018-06-19T22:57:30.000Z","energyBalance":-2204.75,
"details":{"power1Watt":-325.25,"power2Watt":-1795.5,"power3Watt":-84,
"current1Ampere":2.3,"current2Ampere":8.1,"current3Ampere":1.4,
"voltage1Volt":229.29999999999998,"voltage2Volt":229.75,"voltage3Volt":230.75}}
```

Beschreibung der Felder in Response

Feld	Bedeutung	Beschreibung
endTime	Ende Zeitpunkt der Messperiode (UTC)	
startTime	Start Zeitpunkt der Messperiode (UTC)	
energyBalance	Gemessene Energiebilanz des Haushalts	Hier wird folgende Nomenklatur verwendet: • *negative* Werte entsprechen einem *Verbrauch* • *positive* Werte entsprechen einer *Einspeisung*
details	Detailwerte für Strom, Spannung und Leistung pro Phase	
details /powerXWatt		Durchschnittliche Leistung im gegebenen Zeitbereich für Phase X (X=1,2,3) in der physikalischen Einheit Watt
details/currentXAmpere		Durchschnittliche Stromstärke im gegebenen Zeitbereich für Phase X (X=1,2,3) in der physikalischen Einheit Ampère
details/voltageXVolt		Durchschnittliche Spannung im gegebenen Zeitbereich für Phase X (X=1,2,3) in der physikalischen Einheit Volt

Abfrage historischer Werte (getHistoricalData)

Die API ist hier verfügbar:

Protokoll	URL
HTTP	http://<cubeIp>/api/MeterData/getHistoricalData

Parameter

Name	Wert	Beschreibung
format	json | xml	Gewünschtes Ausgabeformat. Default = json
startTime	2018-06-19T22:00:00.000Z	Startzeitpunkt (inklusive). Format: ISO 8601
endTime	2018-06-19T23:00:00.000Z	Endzeitpunkt (exklusive). Format: ISO 8601
granularity	10s | 1m | 15m | 1h	Gewünschte Auflösung der Daten • 10s: 10-Sekunden Aggregate • 1m: 1-Minuten Aggregate • 15m: 15-Minuten Aggregate • 1h: 1-Stunden Aggregate

Beispielaufruf

```
# Explizite Angabe des Formats xml

curl  --location "http://192.168.1.174/api/MeterData/getHistoricalData?format=xml&startTime=2018-06-19T22:00:
00.000Z&endTime=2018-06-19T23:00:00.000Z&granularity=15m" --header 'apiKey: myCustomAPIKey'
<?xml version="1.0" encoding="utf-8"?>
<energyData>
  <entry>
    <energyBalance>401</energyBalance>
    <endTime>2018-06-19T22:00:00.000Z</endTime>
    <startTime>2018-06-19T21:45:00.000Z</startTime>
    <correctedUsageWh>406</correctedUsageWh>
    <numIntervals>445</numIntervals>
    <details>
      <usagePhase1Wh>69</usagePhase1Wh>
      <usagePhase2Wh>105</usagePhase2Wh>
      <usagePhase3Wh>227</usagePhase3Wh>
    </details>
  </entry>
  <entry>
    <energyBalance>366</energyBalance>
    <endTime>2018-06-19T22:15:00.000Z</endTime>
    <startTime>2018-06-19T22:00:00.000Z</startTime>
    <correctedUsageWh>366</correctedUsageWh>
    <numIntervals>450</numIntervals>
    <details>
      <usagePhase1Wh>68</usagePhase1Wh>
      <usagePhase2Wh>50</usagePhase2Wh>
      <usagePhase3Wh>248</usagePhase3Wh>
    </details>
  </entry>
  <entry>
    <energyBalance>170</energyBalance>
    <endTime>2018-06-19T22:30:00.000Z</endTime>
    <startTime>2018-06-19T22:15:00.000Z</startTime>
    <correctedUsageWh>170</correctedUsageWh>
    <numIntervals>451</numIntervals>
    <details>
      <usagePhase1Wh>80</usagePhase1Wh>
      <usagePhase2Wh>50</usagePhase2Wh>
      <usagePhase3Wh>40</usagePhase3Wh>
    </details>
  </entry>
  <entry>
    <energyBalance>205</energyBalance>
    <endTime>2018-06-19T22:45:00.000Z</endTime>
    <startTime>2018-06-19T22:30:00.000Z</startTime>
    <correctedUsageWh>205</correctedUsageWh>
    <numIntervals>450</numIntervals>
    <details>
      <usagePhase1Wh>62</usagePhase1Wh>
      <usagePhase2Wh>119</usagePhase2Wh>
      <usagePhase3Wh>24</usagePhase3Wh>
    </details>
```

```
  </entry>
</energyData>


# Beispiel bei keiner Angabe des Formats (Standard = JSON)

Fritzs-MacBook-Pro-4:~ fritz$ curl  --location "http://192.168.1.174/api/MeterData/getHistoricalData?
startTime=2018-06-19T22:00:00.000Z
&endTime=2018-06-19T23:00:00.000Z&granularity=15m" --header 'apiKey: myCustomAPIKey'
[{"energyBalance":401,"endTime":"2018-06-19T22:00:00.000Z","startTime":"2018-06-19T21:45:00.000Z","
correctedUsageWh":406,"numIntervals":445,"details":{"usagePhase1Wh":69,"usagePhase2Wh":105,"usagePhase3Wh":
227}},
{"energyBalance":366,"endTime":"2018-06-19T22:15:00.000Z","startTime":"2018-06-19T22:00:00.000Z","
correctedUsageWh":366,"numIntervals":450,"details":{"usagePhase1Wh":68,"usagePhase2Wh":50,"usagePhase3Wh":248}},
{"energyBalance":170,"endTime":"2018-06-19T22:30:00.000Z","startTime":"2018-06-19T22:15:00.000Z","
correctedUsageWh":170,"numIntervals":451,"details":{"usagePhase1Wh":80,"usagePhase2Wh":50,"usagePhase3Wh":40}},
{"energyBalance":205,"endTime":"2018-06-19T22:45:00.000Z","startTime":"2018-06-19T22:30:00.000Z","
correctedUsageWh":205,"numIntervals":450,"details":{"usagePhase1Wh":62,"usagePhase2Wh":119,"usagePhase3Wh":24}}]
```

Beschreibung der Felder in Response

Feld	Bedeutung	Beschreibung
endTime	Ende Zeitpunkt der Messperiode (UTC)	
startTime	Start Zeitpunkt der Messperiode (UTC)	
energyBalance	Aggregierte Energiebilanz über den Zeitraum	Hier wird folgende Nomenklatur verwendet: • *negative* Werte entsprechen einem *Verbrauch* • *positive* Werte entsprechen einer *Einspeisung* Physikalische Einheit: Wh
correctedUsageWh	Linear korrigierte Energiebilanz über den Zeitraum	Falls nicht alle Intervalle im gegebenen Zeitraum empfangen wurden, wird auf den Soll-Wert linear interpoliert. Dieses Feld enthält den berechneten Verbrauch. Hier wird folgende Nomenklatur verwendet: • *negative* Werte entsprechen einem *Verbrauch* • *positive* Werte entsprechen einer *Einspeisung* Physikalische Einheit: Wh *Nur verfügbar bei granularity = 15m!*
numIntervals	Anzahl der in Betracht gezogenen Detail-Intervalle des Zählers	Dient als Basis zur Ermittlung des Felds correctedUsageWh *Nur verfügbar bei granularity = 15m!*
details	Detailwerte für Verbrauch /Einspeisung pro Phase	*Nur verfügbar bei granularity = 15m!*
details/usagePhaseXWh		Durchschnittlicher Verbrauch/Einspeisung im gegebenen Zeitbereich für Phase X (X=1,2,3) in der physikalischen Einheit Wattstunden

Abfrage von Handelsdaten vom Marktplatz

Die API ist hier verfügbar:

Protokoll	URL
HTTP	http://<cubeIp>/api/MeterData/TradingSummary

```
 Fritzs-MacBook-Pro-4:~ fritz$ curl  --location http://192.168.1.174/api/MeterData/TradingSummary --header
'apiKey: myCustomAPIKey'
{
   "aggregatedData":{
      "producer":false,
      "consumer":true,
      "pv":15,
      "water":289,
      "biogas":0,
      "wind":0,
      "eFriendsEnergy":782,
      "toCommunity":0,
      "toSupplier":0,
      "toGrid":0,
      "communityShareable":0,
      "fromCommunity":304,
      "fromSupplier":782,
      "fromGrid":1086,
      "communityConsumable":0,
      "balance":-782,
      "productionDetails":{
         "lastProductionPowerW":0,
         "lastACBatteryInputW":0,
         "lastDCBatteryInputW":0,
         "lastBatterySOC":0,
         "lastBatteryOutputW":0,
         "lastInverterPowerW":0,
         "inverterProducedTodayWh":0
      }
   },
   "cubeName":"P01_3000101",
   "startTime":"2021-12-09T09:42:50.000Z",
   "endTime":"2021-12-09T09:43:00.000Z",
   "energyBalance":-1085.6,
   "totalOrderVolume":304,
   "consumable":0,
   "remainingEnergyBalance":-781.6,
   "unixTimestamp":1639042980000,
   "details":{
      "power1Watt":-260.8,
      "power2Watt":-697.2,
      "power3Watt":-128.4,
      "current1Ampere":1.86,
      "current2Ampere":3.32,
      "current3Ampere":1.42,
      "voltage1Volt":227.8,
      "voltage2Volt":225,
      "voltage3Volt":229
   },
   "inverterDetails":{
      "endTime":1639042980000,
      "avgValue":0,
      "numAvgValues":2,
      "producedTodayWh":0,
      "avgInterval":10000
   },
   "batteryDetails":{
      "endTime":1639042980000,
      "numAvgValues":2,
      "batterySOC":3,
      "batteryPBat":0,
      "batteryPGrid":-1172,
      "batteryPInput":0,
      "avgInterval":10000
   }
}
```

Beschreibung der Felder in Response

Feld	Bedeutung	Beschreibung
aggregatedData/ producer bzw. consumer	Rolle des Cubes in der vergangenen Periode	Wurde in der letzten Handelsperiode von 10 Sekunden gekauft oder verkauft, wird das jeweilige Feld auf true gesetzt.
aggregatedDate/ pv water biogas wind	Energiemix Information	Angabe in W, wie viel Energie innerhalb der letzten 10 Sekunde von der jeweiligen Produktionssparte PV, Kleinwasserkraft, Biogas oder Wind gekauft wurde
aggregatedData/ toCommunity toSupplier toGrid communityShareable bzw. fromCommunity fromSupplier fromGrid communityConsumable	Energiebilanz	In Abhängigkeit ob produziert oder konsumiert wurde (jeweils Summe der letzten 10 Sekunden): Als Produzent: • toCommunity: Menge in W, die an Community verkauft wurde • toSupplier: Menge in W, die an Energielieferant verkauft wurde • toGrid: Menge in W, die in Summe ins Netz eingespeist wurde • communityShareable: Menge in W, die noch an Community gehen hätte können (gleicher Wert wie toSupplier) Als Konsument: • fromCommunity: Menge in W, die von Community gekauft wurde • fromSupplier: Menge in W, die vom Energielieferanten gekauft wurde • fromGrid: Menge in W, die in Summe vom Netz bezogen wurde • communityConsumable: Menge in W, die in der Community noch verfügbar gewesen wäre, aber nicht benötigt wurde

Push Notifikation / WebSocket Interface

Am Cube ist mittels socket.io eine Websocket Schnittstelle eingerichtet, mit Hilfe derer man sich auf 10-Sekunden Aggregate von Messwerten registrieren kann.

Protokoll	URL
WebSockets	https://<cube>/MeterDataAPI
Event Name	/api/MeterData/10sData

Ist man erfolgreich registriert, erhält man alle 10 Sekunden eine JSON Nachricht zugesandt.

Beschreibung der Felder in Response

Feld	Bedeutung	Beschreibung
endTime	Ende Zeitpunkt der Messperiode (UTC)	
startTime	Start Zeitpunkt der Messperiode (UTC)	
energyBalance	Gemessene Energiebilanz des Haushalts	Hier wird folgende Nomenklatur verwendet: • *negative* Werte entsprechen einem *Verbrauch* • *positive* Werte entsprechen einer *Einspeisung*
details	Detailwerte für Strom, Spannung und Leistung pro Phase	
details /powerXWatt		Durchschnittliche Leistung im gegebenen Zeitbereich für Phase X (X=1,2,3) in der physikalischen Einheit Watt
details /currentXAmpere		Durchschnittliche Stromstärke im gegebenen Zeitbereich für Phase X (X=1,2,3) in der physikalischen Einheit Ampère
details /voltageXVolt		Durchschnittliche Spannung im gegebenen Zeitbereich für Phase X (X=1,2,3) in der physikalischen Einheit Volt

Weitere Websocket Events

2-Sekunden Rohdaten vom Zähler

Event-Name: "rawPowerMessage"

```
{
  "timestamp":"2019-06-26T15:23:38.738Z",
  "power1Watt":"-1542",
  "power2Watt":"389",
  "power3Watt":"261",
  "current1Ampere":"7.1",
  "current2Ampere":"1.9",
  "current3Ampere":"1.6",
  "voltage1Volt":"226.9",
  "voltage2Volt":"223.1",
  "voltage3Volt":"231.3",
  "powerTotal":"-892"
}
```

10-Sekunden Daten inkl. Inverter und Batterie

Event-Name: "10sEnergyEvent"

```
{
  "startTime":"2019-06-26T15:34:20Z",
  "endTime":"2019-06-26T15:34:30Z",
  "energyBalance":-2,
  "inverterPower":468.3333333333333,
  "inverterProductionDailyWh":23403,
  "batterySOC":86,
  "batteryPInput":294.6666666666667,
  "batteryPGrid":-2.6666666666666665,
  "batteryPBat":-47.333333333333336
}
```

Handelsdaten vom Marktplatz

Event-Name: "/api/MeterData/TradingSummary"

Siehe Beispiel oben für GET Request /api/MeterData/TradingSummary

Beispiel-Implementierung Javascript

```
<script src="//cdnjs.cloudflare.com/ajax/libs/socket.io/4.6.0/socket.io.min.js"></script>
<script src="//code.jquery.com/jquery-2.1.4.min.js"></script>

<script>
(function($) {
  $(document).ready(function() {
      var remoteSocket = 'http://192.168.1.174/MeterDataAPI';
      console.log("Connecting to: " + remoteSocket);

      var socket = io.connect(remoteSocket, {auth: {token: 'myCustomAPIKey'}});
      socket.on("rawPowerMessage", function(data) {
    console.log(data);
      }
  });
})(jQuery);
</script>
```