PUBLIC

SAP Cloud for Energy

Sample API Requests

Contents

SAP Samples — Introduction

You have a valid *SAP Cloud for Energy* license and want to get familiar with the *Energy Data Services* API? Then this guide in combination with the sample request messages we offer for download on SAP Samples are a great resource for you:

You will learn about the basic concepts of the CIM-based Energy Data Services API, the CIM core data model, and the CIM objects in detail. CIM stands for Common Information Model, which is an international standard globally accepted for modeling the information exchanges required in the electric utility industry.

You will get familiar with the API by using simple payload examples first and achieve expert status by using the more complex sample payloads. The examples are designed in a way to help you learn how to use the API and at the same time gather concept knowledge about the corresponding objects and entities.

This offering is a supplement to the product documentation available on the SAP Help Portal. You must still read and follow the instructions given in the Energy Data Services API Guide. The product documentation, this guide, and the sample request messages in combination will enable you to successfully use the *Energy Data Services* API and *SAP Cloud for Energy*.

Note Support and maintenance claims do not apply to this additional offering on SAP Samples.

Target Groups

The sample request messages are for beginners, advanced users, and experts.

Beginner:
In this section, you will be guided step by step how to use the examples, and you will learn about the expected result of each operation.
The advanced section will help you to deepen your knowledge.

Advanced:
In this section, you will find more complex examples.

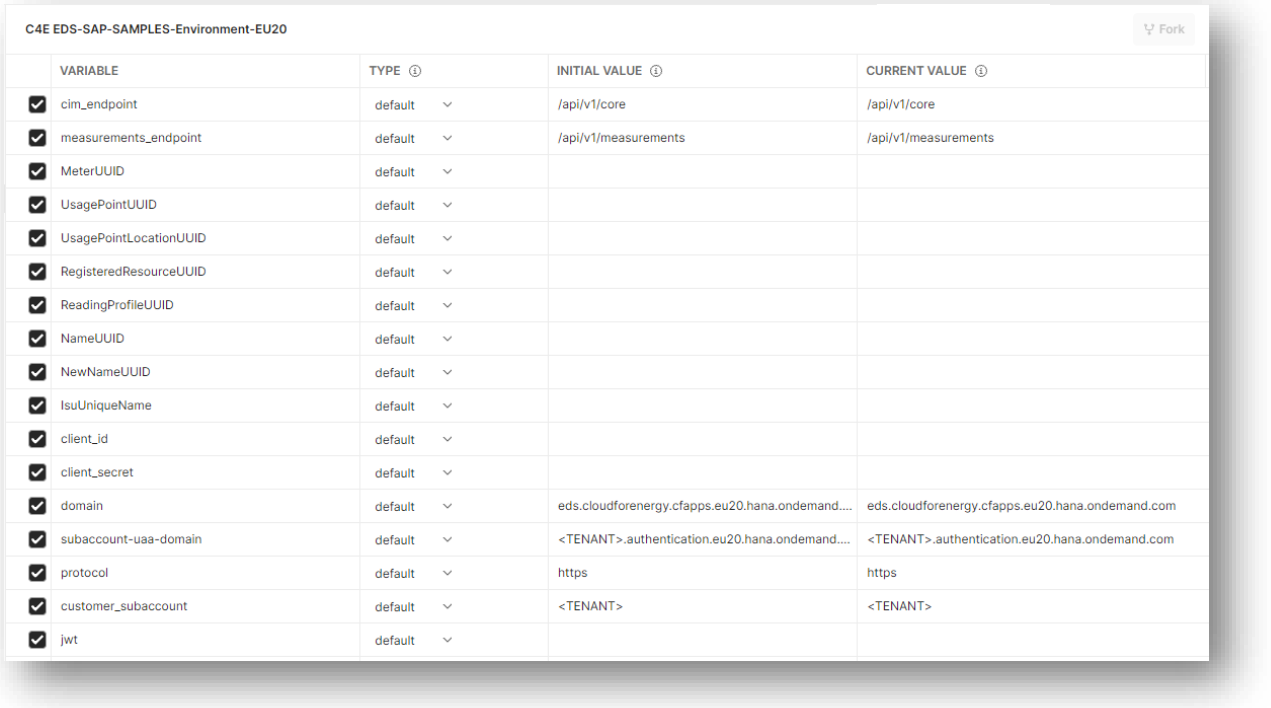
Expert:
The examples in this section are partly guided, others are for your reference only.

Prerequisites

You know how to work with API endpoints in the cloud and are familiar with the basic concepts of security in the context of APIs.

Before you can use the collection of sample request messages, you must make sure to also fulfill the following prerequisites:

- You have a valid *SAP Cloud for Energy* license.
- You have access to *SAP Cloud for Energy* via API endpoint (credentials!).
- You have downloaded the sample Postman collection and added the collection as well as the environment to your local Postman installation.
- You have adjusted the sample Postman environment.
  The required information is available in the SAP BTP (Business Technology Platform) cockpit, in the corresponding subaccount for which you have provisioned *SAP Cloud for Energy*.



C4E EDS-SAP-SAMPLES-Environment-EU20

Fork

	VARIABLE	TYPE	INITIAL VALUE	CURRENT VALUE
☑	cim_endpoint	default	/api/v1/core	/api/v1/core
☑	measurements_endpoint	default	/api/v1/measurements	/api/v1/measurements
☑	MeterUUID	default		
☑	UsagePointUUID	default		
☑	UsagePointLocationUUID	default		
☑	RegisteredResourceUUID	default		
☑	ReadingProfileUUID	default		
☑	NameUUID	default		
☑	NewNameUUID	default		
☑	IsuUniqueName	default		
☑	client_id	default		
☑	client_secret	default		
☑	domain	default	eds.cloudforenergy.cfapps.eu20.hana.ondemand....	eds.cloudforenergy.cfapps.eu20.hana.ondemand.com
☑	subaccount-uaa-domain	default	<TENANT>.authentication.eu20.hana.ondemand....	<TENANT>.authentication.eu20.hana.ondemand.com
☑	protocol	default	https	https
☑	customer_subaccount	default	<TENANT>	<TENANT>
☑	jwt	default		

Figure 1: Environment Variables

You must provide valid values for the following attributes:

client_id	Add this into the environment
client_secret	Add this into the environment
subaccount-uaa-domain	Replace <TENANT> with your tenant's name
customer_subaccount	Replace <TENANT> with your tenant's name

Caution Never share the environment with anyone because it contains the ID and password for accessing the API endpoint for the specific tenant.

This guide is not a training or enablement kit for tools such as *Postman*.

Important Note for Users with Connected SAP S/4HANA Utilities or SAP ECC System

If you use *SAP Cloud for Energy* with an integration to *SAP S/4HANA Utilities* or *Utilities as part of SAP ECC (ERP Central Component)*, you cannot change and delete meter master data via *Energy Data Services* API calls.

In your usage scenario, *SAP S/4HANA Utilities* or *SAP ECC* are the leading systems for device management and the meter master data is replicated to *SAP Cloud for Energy*. You can only read (“get”) the meter information using the API.

In any usage scenario (with or without *SAP S/4HANA Utilities* integration), all operations on meter readings are relevant.

The Collection of Sample Request Messages

For each target group, you can find specific folders with a bundle of sample request messages.

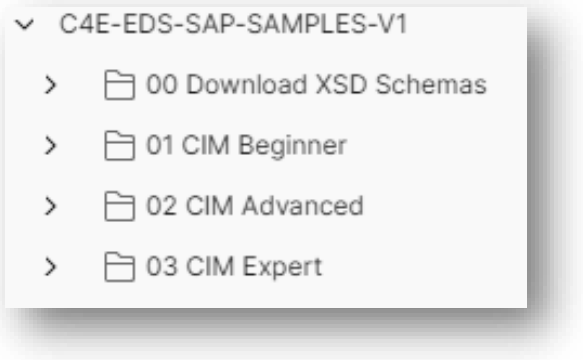


Figure 2: Sample Collection

You don't necessarily have to follow this suggested sequence of steps, but we highly recommend doing so. Experience shows that even the simple examples are a particularly good starting point also for experienced users.

Note We do not guarantee the correctness or completeness of the sample request messages. Not all (new) product or API features are reflected in this sample

About CIM and the Sample Request Messages

To understand and work with the content, you should know the following:

XML Schema

Download the XSDs (XML Schema Definitions) for Profiles and Messages.

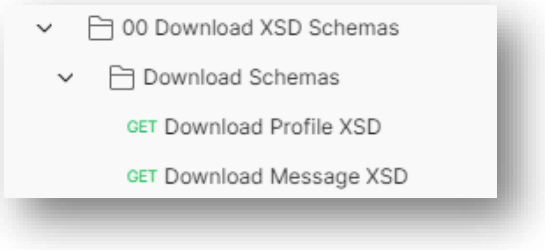


Figure 3: Schemas to Validate the XML

Request Message Header

The *Energy Data Services* API is based on the *Common Information Model* (CIM). The request messages in the CIM format always have a *Header* section, which consists of a *Verb* and a *Noun*.

For instance, the following message header means that this is a request message to read (*get)* information about a meter (*MeterConfig*):

```
<msg:Header>
      <msg:Verb>get</msg:Verb>
      <msg:Noun>MeterConfig</msg:Noun>
</msg:Header>
```

This next example is about creating (*create)* a logical device (*UsagePointConfig)*:

```
<msg:Header>
      <msg:Verb>create</msg:Verb>
      <msg:Noun>UsagePointConfig</msg:Noun>
</msg:Header>
```

The following message header means that this is a request message to read (*get)* measurement data (*MeterReadings)*:

```
<msg:Header>
      <msg:Verb>get</msg:Verb>
      <msg:Noun>MeterReadings</msg:Noun>
</msg:Header>
```

More request message headers for master data and measurement data-related request messages can be found in the product documentation on the SAP Help Portal under *Request Messages to the Energy Data Services API.*

Variables in the Requests

Variables are defined in the environment or set via pre-request scripts.

Variables are taken from the environment or, to keep the information available for later request messages, written to the environment.

Variables are in **{{**now**}}**, as shown in the following example:

```
<m:effectiveDateTime>{{now}}</m:effectiveDateTime>
```

Meter

A fully configured device is more than just the MeterConfig.

It consists of a minimum of 4 important CIM objects which do have relations to each other:

- UsagePointLocationConfig
- UsagePointConfig
- MeterConfig
- MasterDataLinkageConfig

Identification of CIM Objects

CIM objects are identified either using IDs, in this case UUIDs (Universally Unique Identifier), or using *Name*. *Name* can be any item registered as a name. Names must be UNIQUE (see examples).

Measurement Data Ingestion – Reading Type, Channel / Register

SAP Cloud for Energy checks if the meter has a configured reading type (channel/register) for the measurement data that you want to ingest. If, for example, you want to ingest hourly water consumption time series values, the meter needs to be configured in the same way that it can hold hourly water consumption time series values.

The decimals before and after the decimal point (“.”) are also checked during the measurement data ingestion. If the data that you try to ingest does not match the meter configuration, the API sends a negative response.

Measurement Data Ingestion – Time Zone of the Meter and Summer-/Wintertime

SAP Cloud for Energy stores measurement data in UTC (Universal Time Coordinated). If you ingest measurement data, you must consider the time zone in which the corresponding meter is installed and the offset from UTC. Read the chapter in the product documentation on the SAP Help Portal and look at the examples for meters located in

Berlin, Montréal, or New York. Pay special attention to daylight saving (summer-/wintertime).

Measurement Data Ingestion for Dates in the Future

SAP Cloud for Energy does not allow measurement data ingestion for dates in the future. If you try to ingest measurement data for dates in the future, a corresponding error message will be returned.

Operation Set

An operation set is a collection of single requests put together in one request message.

Response

For each request message that you post, you get a response.

To keep this guide readable, we won't show each response. You can recognize a positive response by the following content:

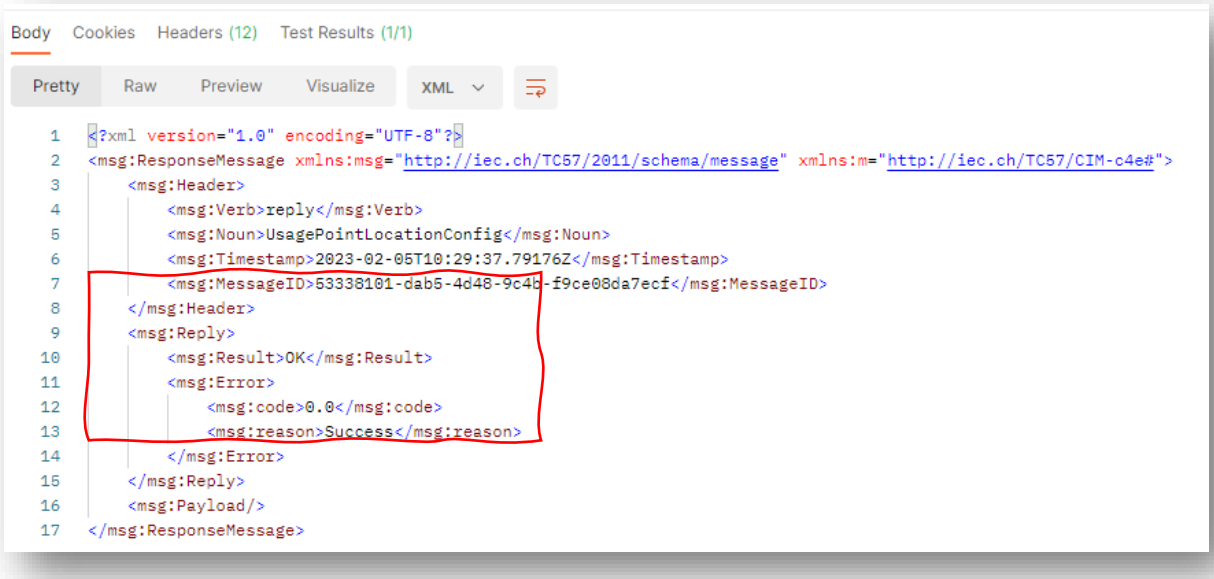


Figure 4: Positive Response

Screenshots with style “soft edge oval” are fragments of a bigger picture. They show how the request message could look on your side.

Figure 5: Soft Edge Oval Screenshots

Before You Get Started — Understanding the IEC 61968-9 Basic Concepts

SAP Cloud for Energy consists of several microservices. Each microservice targets a certain topic. Since microservices are developed and operated by SAP, you don't need to worry about them.

However, to understand how the *Energy Data Services* API works, you should have a basic understanding of the IEC 61968-9 standard's objects and how they have been implemented in *SAP Cloud for Energy*.

Energy Data Services API

Energy Data Services is SAP Cloud for Energy's core component, which provides an API to create, read, update, and delete measurement data and meter master data in the cloud.

The base URL for the *Energy Data Services* API is:
`https://eds.cloudforenergy.cfapps.eu20.hana.ondemand.com` (EU tenants)
`https://eds.cloudforenergy.cfapps.us21.hana.ondemand.com` (U.S. tenants)

You access the *Energy Data Services* API by using the base URL and adding the URL extension for the respective endpoint as shown in the table below.

Energy Data Services API - HTTP Endpoints

Endpoint	HTTP Method	Action	URL
Endpoint for meter master data	*POST*	Send / retrieve meter master data	`<base URL>/api/v1/core`
Endpoint for measurement data	*POST*	Send / retrieve MeterReadings	`<base URL>/api/v1/measurements`
Configuration endpoint	*POST*	Make configuration settings	`<base URL>/api/v1/configuration`
	GET	Retrieve configuration settings	
XML schema definition (XSD) for message envelopes	*GET*	Retrieve XSD for the message envelopes	`<base URL>/api/v1/schema/message.xsd/`
XML schema definition (XSD) for payloads	*GET*	Retrieve XSD for the payloads	`<base URL>/api/v1/schema/profile.xsd/`

Figure 6: API Endpoints

The *Energy Data Services* API is protected with OAuth 2.0 client credentials.

In general, the exchange of information between a client and *Energy Data Services* is based on the norms IEC 61968-9, IEC 61968-100, and IEC 61968-900 (see REFERENCES).

The product documentation describes where the *Energy Data Services* API adheres to the IEC standard, where and how the IEC standard is interpreted in a specific way, which restrictions and deviations exist, and which extensions are provided. More information: Energy Data Services API Guide on the SAP Help Portal.

CIM-based Core Data Model

This section focuses on the most important and central parts of the data model and its concepts.

Energy Data Services uses a data model that follows the standard IEC 61968-9, which is part of the Common Information Model (CIM). CIM is an international standard globally accepted for modeling the information exchanges required in the electric utility industry.

The scope of the IEC 61968 standard is the exchange of information between a Metering System and other systems within a utility. Part 9 describes the "Interface Standard for Meter Reading and Control".

The data model is based on the description of the entities in IEC61968-9 and on the UML model provided.

However, as outlined in the standard:

> (...) it is necessary to understand that the IEC 61968-9 standard provides normative messages by which enterprise systems can exchange information; however, there is no requirement that any given system implement an internal data model that is based upon or even consistent with the CIM.

(IEC 61968-9, pg. 85)

It is also important to understand the following:

> The IEC 61968 series of standards is intended to facilitate inter-application integration as opposed to intra-application integration. Intra-application integration is aimed at programs in the same application system, usually communicating with each other using middleware that is embedded in their underlying runtime environment, and tends to be optimized for close, real-time, synchronous connections and interactive request/reply or conversation communication models. IEC 61968, by contrast, is intended to support the inter-application integration of a utility enterprise that needs to connect disparate applications that are already built or new (legacy or purchased applications), each supported by dissimilar runtime environments.

(IEC 61968-9, pg. 12)

The goal of the CIM model is to standardize the external interface of an application.

Object Relations (Simplified)

The following diagram shows the main entities.

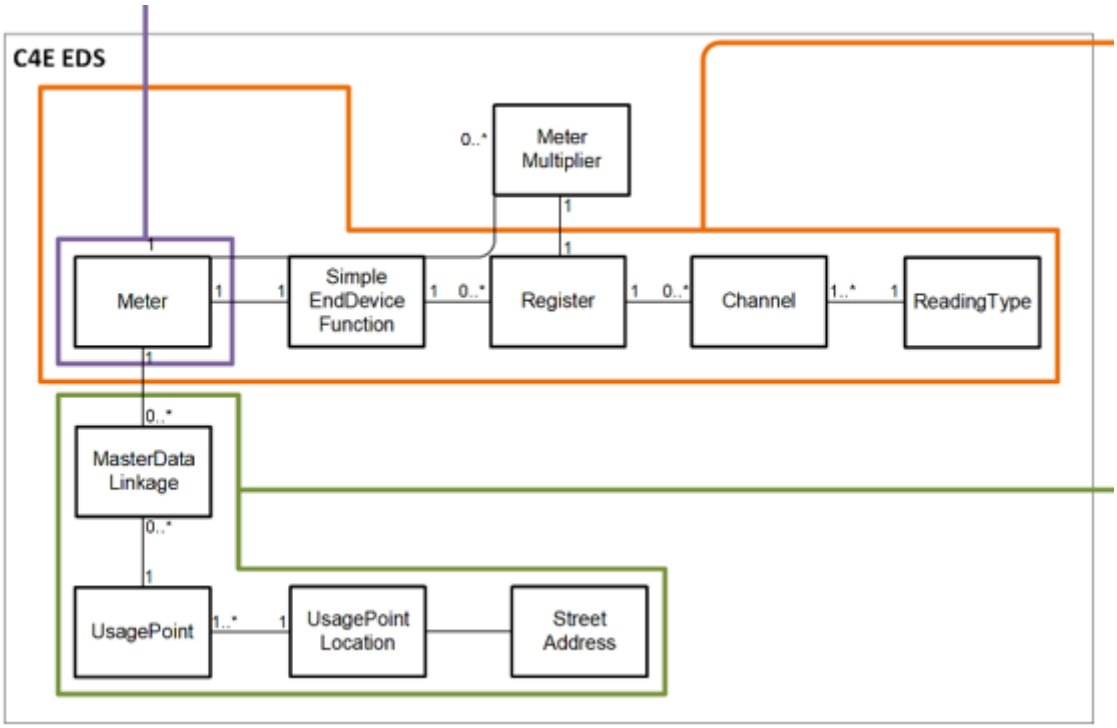


Figure 7: Object Relations

The sample request messages for beginners, advanced users, and experts also refer to those entities.

Note: All cardinalities are given from the data model perspective.
The payload may be more restrictive.

The CIM Objects in Detail

Object Identity

An important characteristic that all master data objects in *Energy Data Services* share is the **Object Identity.** An object that is "addressable" has an object identity that consists of an **MRID** (Master Resource Identifier), which is a UUID (Universally Unique Identifier) uniquely identifying the addressed object. In addition, objects have **names** that are also unique identifiers addressing the object.
A name is an alternative key for an object.

Meter

The central part of the *Energy Data Services* model is the meter and its structure.
A **Meter** is a physical device used for measuring consumption and detecting events.

Meter Revisions

Energy Data Services tracks all changes to meters. This lets us know exactly when the **meter revision** was valid and used. Whenever a meter's data and its structure is used to manage, for example, meter readings, the meter revision is important. With this information, you can, for example, find out which master data was used to validate incoming meter readings.

Meter Channels

The meter is organized in a hierarchical structure. You can imagine a meter as a group of **channels**.
These channels share common fields from the meter hierarchy.

The standard gives the following **definition of a channel**:

> A single path for the collection or reporting of register values over a period. For example, a register which measures forward energy can have two channels, one providing bulk quantity readings and the other providing interval readings of a fixed interval size.

(IEC 61968-9, pg. 28)

Bulk quantity reading is another term for meter reading value. Interval reading is another term for consumption value. You can find more information about the types of data managed with *SAP Cloud for Energy* and the terminology on the SAP Help Portal.

A channel has an object identity, that means it can be addressed and linked if necessary.

The channel contains the following fields:

Field	Cardinality	Description
identificationSystemCode	0..1	OBIS code or other code used by an external system to identify the channel *(SAP Extension)*.
isVirtual	0..1	If true, the data is calculated by an enterprise system rather than metered directly.
measurementTask	0..1	Measurement Task *(SAP Extension)*
veeCode	0..1	VEE Code *(SAP Extension)*
ReadingType	0..1	Reading type for register values reported/collected by this channel

The most important field in a channel is the **reading type.** The reading type describes the data that is reported/collected by this channel. Most of the other fields in a channel are extensions by SAP that do not exist in CIM.

Since measurement data reported/collected by a channel is addressed using the object identity of a meter and the reading type of a channel, the reading type must be unique

within a meter. As the identificationSystemCode is mapped to a reading type for data delivery and retrieval, the same is true for this field.

The channel provides additional fields from the hierarchical structure of the meter. Channels are grouped/organized in registers, which provide for example left and right digits of the values reported/collected by a channel. These registers are part of an end device function (or Simple End Device Function), which provides the kind of meter (gas, electricity, water, metrology ...) the channel is part of.

A register indicates or records measured quantities.

The channels that are grouped in a register have the following common fields derived from the register:

Field	Cardinality	Description
isVirtual	0..1	If true, the data it produces is calculated or measured by a device other than a physical end device/meter. Otherwise, any data streams it produces are measured by the hardware of the end device/meter itself.
leftDigitCount	0..1	Number of digits (dials on a mechanical meter) to the left of the decimal place; default is normally 5.
rightDigitCount	0..1	Number of digits (dials on a mechanical meter) to the right of the decimal place.
touTierName	0..1	Name used for the time of use tier (also known as bin or bucket). For example, "peak", "off-peak", "TOU Category A", etc.
touTier	0..1	Clock time interval for register to begin/cease accumulating time of usage (e.g., start at 8am, stop at 5pm).
RegisterMultiplier	0..1	All multipliers applied at this register.
Channels	0..n	All channels that collect/report values from this register.

The **RegisterMultiplier** is a specialization of a **MeterMultiplier** defined as follows:

Field	Cardinality	Description
kind	0..1	Kind of multiplier. (see <MeterMultiplierKind>)
value	0..1	Multiplier value.

If a register is virtual, all the channels contained in the register are considered virtual, independent of the individual value of isVirtual on channel level. A register has an object identity and can be addressed.

If we look at the fields of the register, many of the important fields defining a channel are on register level, e.g., TOU defining fields, or digits, which are important for validation.

The registers are organized and grouped under an End Device Function, or, in the specific case of the Energy Data Services data model, under the specialization SimpleEndDeviceFunction. A meter provides exactly one end device function in Energy Data Services. The SimpleEndDeviceFunction has an object identity and can be addressed. The end device function contains 0.n **Register** objects.

The most essential information of the end device function is the definition of the kind of meter (of type EndDeviceFunctionKind), whether it is a water meter, an electricity, or gas meter.

SimpleEndDeviceFunction also has an *enabled* flag which is set to true if the end device function is enabled.

The fields of the SimpleEndDeviceFunction are the following:

Field	Cardinality	Description
enabled	0..1	True if the function is enabled.
kind	0..1	Kind of this function.
Registers	0..n	All registers for quantities metered by this end device function.

EndDeviceFunctionKind is one of the following values:

Value	Description
autonomousDst	Autonomous application of daylight savings time (DST).
demandResponse	Demand response functions.
electricMetering	Electricity metering.
gasMetering	Gas metering.
metrology	Presentation of metered values to a user or another system (always a function of a meter but might not be supported by a load control unit).
onRequestRead	On-request reads.
outageHistory	Reporting historical power interruption data.
relaysProgramming	Support for one or more relays that may be programmable in the meter (and tied to TOU, time pulse, load control or other functions).
reverseFlow	Detection and monitoring of reverse flow.
waterMetering	Water metering.

Top level fields of a meter

as the container for a simple end device function and all channels:

Field	Cardinality	Description
amrSystem	0..1	Automated meter reading (AMR) system responsible for communications to this end device.
formNumber	0..1	Meter form designation per ANSI C12.10 or another applicable standard. An alphanumeric designation denoting the circuit arrangement for which the meter is applicable and its specific terminal arrangement.
isVirtual	0..1	If true, there is no physical device. As an example, a virtual meter can be defined to aggregate the consumption for two or more physical meters. Otherwise, this is a physical hardware device.
serialNumber	0..1	Serial number of this asset.
timeZone	1	Time zone for the location of this end device.
DataUsageStatus	0.. n	Structures used for blocking access to the Meter and its data
SimpleEndDeviceFunction	0..1	All end device functions this end device performs.
EndDeviceInfo	0..1	End device data.
lifecycle	0..1	Lifecycle dates for this asset.
MeterMultipliers	0..1	All multipliers applied at this meter.

The time zone is used to correctly identify the time zone of the meter location. This information is used in the context of measurement data validations. The time zone is also important for detecting daylight saving time switches.

The **EndDeviceInfo** contains the AssetModel which is a specialized ProductAssetModel in Energy Data Services and describes the manufacturer and the model number of the asset.

Lifecycle
The **lifecycle** contains the following fields for lifecycle event dates of the meter:

Field	Cardinality	Description
installationDate	0..1	(If applicable) Date current installation was completed, which may not be the same as the in-service date. Asset may have been installed at other locations previously. Ignored if asset is (1) not currently installed (e.g., stored in a depot) or (2) not intended to be installed (e.g., vehicle, tool).
receivedDate	0..1	Date the asset was received and first placed into inventory.
removalDate	0..1	(If applicable) Date when the asset was last removed from service. Ignored if (1) not intended to be in service, or (2) currently in service.
retiredDate	0..1	(If applicable) Date the asset is permanently retired from service and may be scheduled for disposal. Ignored if asset is (1) currently in service, or (2) permanently removed from service.

SimpleEndDeviceFunction is the root for the (hierarchical) substructure of the meter.
A meter provides exactly one end device function in Energy Data Services.
The SimpleEndDeviceFunction has an object identity.

Meter Multiplier is defined as follows:

Field	Cardinality	Description
kind	0..1	Kind of multiplier. <see MeterMultiplierKind>
value	0..1	Multiplier value.

MeterMultiplierKind is defined as follows:

Value	Description
ctRatio	Current transformer ratio used to convert associated quantities to real measurements.
kE	Test constant.
kH	Meter kh (watthour) constant. The number of watthours that must be applied to the meter to cause one disk revolution for an electromechanical meter or the number of watthours represented by one increment pulse for an electronic meter.
kR	Register multiplier. The number to multiply the register reading by to get kWh.
ptRatio	Potential transformer ratio used to convert associated quantities to real measurements.
transformerRatio	Product of the CT ratio and PT ratio.

Hierarchical Structure of the Meter

A meter is a container for channels. These channels are defined in a hierarchical structure with SimpleEndDeviceFunction at the top. The various levels of the hierarchical structure define common fields shared by the lower hierarchy.

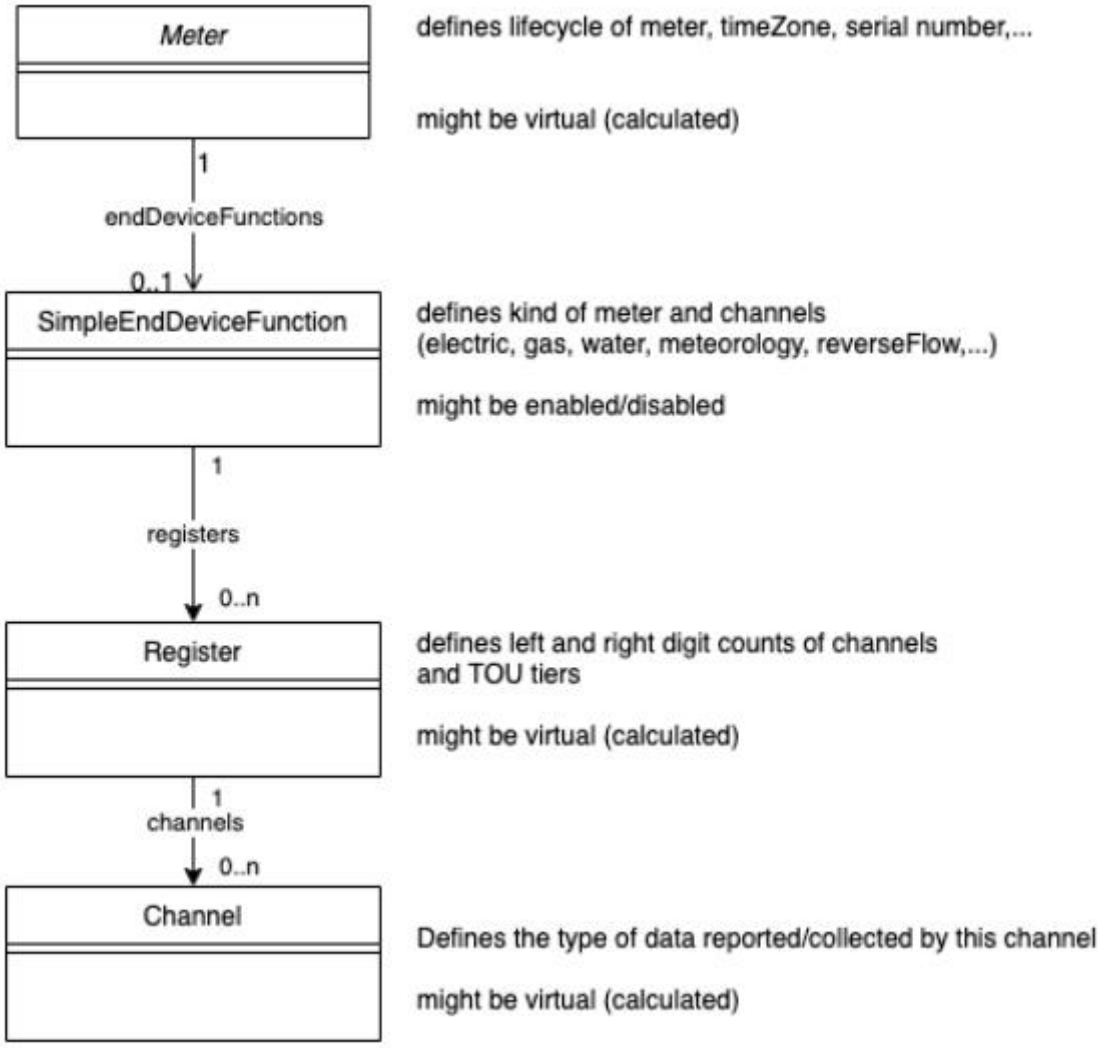


Figure 8: Hierarchical Structure and Object Relation

The fields of the meter define the container, with the lifecycle, identity/serial number etc. The fields that are hierarchically grouped under SimpleEndDeviceFunction describe the channels.

Various parts of the hierarchical structure can be declared as virtual. The virtual parts manage calculated data in contrast to measured data. If the hierarchy above declares something as virtual, all the levels below are also assumed to be virtual.

If there are multiple channels in a meter, you can either provide a register for every channel, or combine channels with a common configuration in the same register. A criterion for combining channels could be, for example, the same right and left digit counts. Another criterion could be that the channels are in the same register.

Reading Type

A reading type describes the

> Type of data conveyed by a specific Reading.

(IEC 61968-9, pg. 31)

Key to the reporting of a value from a meter is the reading type. The Reading Type is used to identify the reading's meaning and the data type of the value. The reading type defines the values collected/measured by a channel. It is a field of the channel, and it is also associated with measurement values. The reading type tries to describe the (reading) data as best as possible. The description is provided by a data producer. The idea is that the data should be self-describing so that consumers who access the (reading) data can figure out for themselves the suitability for use of the data.

Reading types returned in a response to a meter reading request message might differ slightly from the reading types in the original request message. The reason for this is that the queried system (SAP Cloud for Energy) selects the right data and responds with a best effort approach way. It could also happen that the system returns more details than requested in the reading type.

The name of a reading type is special in the sense that it follows a defined naming convention (Annex C of IEC 61968-9). SAP Cloud for Energy typically ignores the name type and name type authority of a reading type name. From an existing list of names, the system only uses the very first name of that list.

Reading Type Names can be described in terms of 18 key attributes (including compound attributes.) Every attribute has the feature that a value of zero (“0”) means that it is not applicable to the description. The only exception to this rule is in the compound attributes. In this case dual zeros describe that the attribute is not applicable.

The values in attributes of the reading type name allow the creation of recommended codes to be used for identifying a reading value as follows:

```
<macroPeriod>.<aggregate>.<measuringPeriod>.<accumulation>.<flowDirection>.<commodity>.
<measurementKind>.<interharmonic.numerator>.<interharmonic.denominator>.<argument.numer
ator>.<argument.denominator>.<tou>.<cpp>.<consumptionTier>.<phases>.<multiplier>.<unit>
.<currency>
```

Field	Description
macroPeriod	Time period of interest reflecting how the reading is viewed or captured over a long time.
aggregate	Salient attribute of the reading data aggregated from individual endpoints. This is mainly used to define a mathematical operation carried out over a 'macroPeriod', but may also be used to describe an attribute of the data when the 'macroPeriod' is not defined.
measuringPeriod	Time attribute inherent or fundamental to the reading value (as opposed to 'macroPeriod' that supplies an "adjective" to describe aspects of a time period with regard to the measurement). It refers to the way the value was originally measured and not to the frequency at which it is reported or presented. For example, an hourly interval of consumption data would have value 'hourly' as an attribute. However, in the case of an hourly sampled voltage value, the meterReadings schema would carry the 'hourly' interval size information. It is common for meters to report demand in a form that is measured over the course of a portion of an hour, while enterprise applications however commonly assume the demand (in kW or kVAr) normalized to 1 hour. The system that receives readings directly from the meter therefore must perform this transformation before publishing readings for use by the other enterprise systems. The scalar used is chosen based on the block size (not any sub-interval size).
accumulation	Accumulation behaviour of a reading over time, usually 'measuringPeriod', to be used with individual endpoints (as opposed to 'macroPeriod' and 'aggregate' that are used to describe aggregations of data from individual endpoints).
flowDirection	Flow direction for a reading where the direction of flow of the commodity is important (for electricity measurements this includes current, energy, power, and demand).

Field	Description
commodity	Commodity being measured.
measurementKind	Identifies "what" is being measured, as refinement of 'commodity'. When combined with 'unit', it provides detail to the unit of measure. For example, 'energy' with a unit of measure of 'kWh' indicates that active energy is being measured, while 'kVAh' or 'kVArh' indicate apparent energy and reactive energy, respectively. 'power' can be combined in a similar way with various power units of measure: Distortion power ('distortionVoltAmperes') with 'kVA' is different from 'power' with 'kVA'.
interharmonic.numerator	Indication of a "harmonic" or "interharmonic" basis for the measurement. Value 0 in 'numerator' and 'denominator' means not applicable.
interharmonic.denominator	Indication of a "harmonic" or "interharmonic" basis for the measurement. Value 0 in 'numerator' and 'denominator' means not applicable.
argument.numerator	Argument used to introduce numbers into the unit of measure description where they are needed (e.g., 4 where the measure needs an argument such as CEMI(n=4)). Most arguments used in practice however will be integers (i.e., 'denominator'=1). Value 0 in 'numerator' and 'denominator' means not applicable.
argument.denominator	Argument used to introduce numbers into the unit of measure description where they are needed (e.g., 4 where the measure needs an argument such as CEMI(n=4)). Most arguments used in practice however will be integers (i.e., 'denominator'=1). Value 0 in 'numerator' and 'denominator' means not applicable.
tou	Time of use (TOU) bucket the reading value is attributed to. Value 0 means not applicable.
cpp	Critical peak period (CPP) bucket the reading value is attributed to. Value 0 means not applicable. Even though CPP is usually considered a

Field	Description
	specialised form of time of use 'tou', this attribute is defined explicitly for flexibility.
consumptionTier	In case of common flat-rate pricing for power, in which all purchases are at a given rate, 'consumptionTier'=0. Otherwise, the value indicates the consumption tier, which can be used in conjunction with TOU or CPP pricing. Consumption tier pricing refers to the method of billing in which a certain "block" of energy is purchased/sold at one price, after which the next block of energy is purchased at another price, and so on, all throughout a defined period. At the start of the defined period, consumption is initially zero, and any usage is measured against the first consumption tier ('consumptionTier'=1). If this block of energy is consumed before the end of the period, energy consumption moves to be reconed against the second consumption tier ('consumptionTier'=2), and so on. At the end of the defined period, the consumption accumulator is reset, and usage within the 'consumptionTier'=1 restarts.
phases	Meteric-specific phase code.
multiplier	Metering-specific multiplier.
unit	Metering-specific unit.
currency	Metering-specific currency.

(Based on definition in IEC 61968-9)

For example, the ReadingType string 0.0.0.1.1.1.12.0.0.0.0.0.0.0.0.3.72.0 is the encoding for bulk forward active energy. (“Bulk” designates a cumulative value and is a synonym for a meter reading value). Forward active energy is a measure of the active rather than reactive electrical energy.

In *SAP Cloud for Energy*, all reading types must be created together with the meter payloads. When a reading type is created, *SAP Cloud for Energy* checks if the reading type already exists. *SAP Cloud for Energy* uses only the first name and not the name type and name type authority for this check. If the name of the reading type already exists in the system, the existing reading type is used, and *SAP Cloud for Energy* does not create a new reading type. The system builds a "library" of reading types.

See *Annex C "Procedure for the generation of a ReadingType name" of IEC 61968-9* for a detailed definition of the reading type. This section describes the technique for constructing and offers recommended enumerations for the ReadingTypeId textual name and mRID.

Some additional remarks regarding the reading type when querying or creating meter readings:

> While the ReadingType structure is present in the MeterReadings message, it should be expected that in normal practice it will not be populated, as it is anticipated that the consuming system will already be configured with the appropriate ReadingType definitions.

(IEC 61968-9, pg. 52)

The reading types used in the meter reading payloads are only used for referencing the already existing reading types of the meter. If a reading type is included in a meter reading payload together with the complete name construct (NameType and NameTypeAuthority), the system will consider the first name, and use only the name part of it.

Supported Reading Types

You only can process (ingest) data for which a valid reading type exists.

See the list of supported reading types in the product documentation on the SAP Help Portal. You can filter the list by all kinds of criteria.

Example: Reading Types for 15min Interval Electricity Data:

ReadingType ID Code	macroPeriod	aggregate	measuringPeriod	accumulation	flowDirection	commodity	measurementKind	tou	consumption tier
Search within the column	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter	Filter
0.0.2.4.0.1.12.0.0.0.0.0.0.0.0.3.72.0	"0" n/a	"0" n/a	"2" (fifteenMinute) 15 minutes interval length	"4" (deltaData) Interval consumption	"0" n/a	"1" (electricitySecondaryMetered) Division electricity	"12" (energy) Energy	"0" n/a	"0" n/a
0.0.2.4.1.1.12.0.0.0.0.0.0.0.0.3.72.0	"0" n/a	"0" n/a	"2" (fifteenMinute) 15 minutes interval length	"4" (deltaData) Interval consumption	"1" (forward) Supply	"1" (electricitySecondaryMetered) Division electricity	"12" (energy) Energy	"0" n/a	"0" n/a
0.0.2.4.19.1.12.0.0.0.0.0.0.0.0.3.72.0	"0" n/a	"0" n/a	"2" (fifteenMinute) 15 minutes interval length	"4" (deltaData) Interval consumption	"19" (reverse) Feed-in	"1" (electricitySecondaryMetered) Division electricity	"12" (energy) Energy	"0" n/a	"0" n/a
0.8.2.6.0.1.8.0.0.0.0.0.0.0.0.3.38.0	"0" n/a	"8" (maximum) The highest value observed in an interval	"2" (fifteenMinute) 15 minutes interval length	"6" (indicating) Demand	"0" n/a	"1" (electricitySecondaryMetered) Division electricity	"8" (demand) Demand	"0" n/a	"0" n/a
0.8.2.6.1.1.8.0.0.0.0.0.0.0.0.3.38.0	"0" n/a	"8" (maximum) The highest value observed in an interval	"2" (fifteenMinute) 15 minutes interval length	"6" (indicating) Demand	"1" (forward) Supply	"1" (electricitySecondaryMetered) Division electricity	"8" (demand) Demand	"0" n/a	"0" n/a

Figure 9: List of Supported Reading Types in the Product Documentation

More reading types are continuously added to *SAP Cloud for Energy*.

Meter Revisions

Meter revisions are specific to *SAP Cloud for Energy* and cannot be found in the original CIM description. The meter object and the hierarchical structure of the meter (simple end device function, register, channel, etc.) are managed in a way that keeps all the revisions of these objects.

These revisions are necessary, because the meter and the channels that are part of the meter, are associated with measurement values reported over a period of time. To understand the reported measurement values, it is necessary to look at the meter configuration used at the time when a measurement value was reported. The typical flow is that the system uses the timestamp of a measurement value to find the revision of the meter that was valid at that point in time, and then uses the configuration/data of this revision as a basis for data validation and processing.

A revision is automatically created whenever the configuration of a meter or its structure is changed.

There are also operations available that can be used to change meter revisions, so that revisions can be corrected without creating new revisions.

Even if revisions exist, the meter can still be queried and addressed with its object identity, because the object identity remains revision independent.

Usage Point

A **Usage Point** is defined as follows:

> Logical point on a Distribution Network to which Meter Readings and/or End Device Events can be attributed.

(IEC 61968-9, pg. 32)

A customer's premise may have one or more meters. The usage point is used to associate the meters with the premise.

Usage points can be used for a variety of purposes. The main purpose of a usage point in *SAP Cloud for Energy* is the definition of a physical location where a meter is installed (e.g., apartment). A usage point can also be used as a logical construct to which readings are attributed when no meter is present.

The usage point has an object identity and can be addressed and referenced.

The following fields are available for the usage point:

Field	Cardinality	Description
amiBillingReady	0..1	Tracks the lifecycle of the metering installation at a usage point with respect to readiness for billing via advanced metering infrastructure reads. Typically, is one of "amiCapable","amiDisabled","billingApproved","enabled","nonAmi","nonMetered","operable".
checkBilling	0..1	True if because of an inspection or otherwise, there is a reason to suspect that an earlier billing may have been performed with

Field	Cardinality	Description
		erroneous data. Value should be reset once this potential discrepancy has been resolved.
isSdp	0..1	If true, this usage point is a service delivery point, i.e., a usage point where the ownership of the service changes hands.
isVirtual	0..1	If true, this usage point is virtual, i.e., no physical location exists in the network where a meter could be located to collect the meter readings. For example, one may define a virtual usage point to serve as an aggregation of usage for all a company's premises distributed widely across the distribution territory. Otherwise, the usage point is physical, i.e., there is a logical point in the network where a meter could be located to collect meter readings.
minimalUsageExpected	0..1	If true, minimal or zero usage is expected at this usage point for situations such as premise vacancy, logical or physical disconnection. It is used for readings validation and estimation.
readCycle	0..1	Cycle day on which the meter for this usage point will normally be read. Usually correlated with the billing cycle.
readRoute	0..1	Identifier of the route to which this usage point is assigned for meter reading. Typically used to configure hand held meter reading systems prior to collection of readings.
UsagePointLocation	0..1	Location of this usage point.

Considering the description of the field "isVirtual", the usage point can be used for a variety of purposes.

The usage point can be used for other logical abstractions such as the measurement location. It can be used for calculations that are made based on data connected to a meter. It can abstract the technical infrastructure used for collecting the readings (the

meter, its channels, and installation) and associate, for instance, consumption to an apartment.

Usage Point Location

A **Usage Point Location** is defined as follows:

> Location of an individual usage point. For residential or most businesses, it is typically the location of a meter on the customer's premises.

(IEC 61968-9, pg. 32)

A usage point location can be addressed via an object identity, but typically it is directly referenced from a usage point. A usage point location provides the location of an individual usage point. In essence it is an address with some additional information.

Field	Cardinality	Description
siteAccessProblem	0..1	Problems previously encountered when visiting or performing work at this location. Examples include bad dog, violent customer, verbally abusive occupant, obstructions, safety hazards, etc.
mainAddress	0..1	Main address of the location. (StreetAddress)

StreetAddress is

Field	Cardinality	Description
streetDetail	0..1	Street detail: Street details, in the context of address
townDetail	0..1	Town detail: Town details, in the context of address

Street detail is

Field	Cardinality	Description
name	0..1	Name of the street
number	0..1	Designator of the specific location on the street

Town detail is

Field	Cardinality	Description
code	0..1	Town code
country	0..1	Name of the country
name	0..1	Name of the town

Master Data Linkage

The master data linkage is used to define relationships between objects that can be addressed with an object identity. In addition to a relationship created via master data linkage, there are also relationships explicitly defined in the objects (such as the connection between a usage point location and a usage point).

SAP Cloud for Energy currently supports the following relationships:

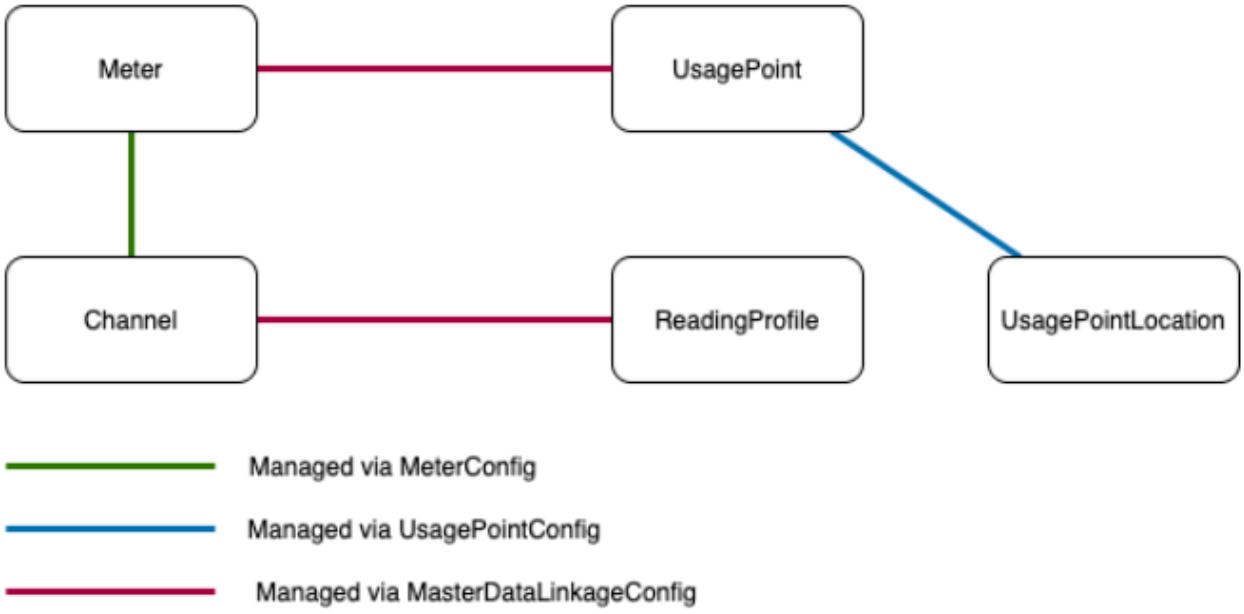


Figure 10: Object Relation

There can be restrictions regarding the number of linkages that can exist at a time.

All linkages are time-dependent, or time-sliced. That means they have a *valid from* and *valid to* timestamp that define exactly when a linkage is valid.

A linkage can be deleted, which ends the relationship between two objects. The deletion of a linkage sets the *valid to* timestamp to the time when the linkage has been deleted.

Linkages can be removed from the system by using time slice operations on linkages. In addition, linkages are automatically removed from the system if one of the objects that is referenced is deleted from the system.

Meter Readings

Meter readings are the representation of register values over a period that are reported or calculated on behalf of a meter.

CIM does not support versioning of meter readings in its definition of meter readings. In *SAP Cloud for Energy*, however, meter readings are versioned. To enable the versioning, the model has been enhanced by a *createdAt* timestamp, that shows when *SAP Cloud for Energy* received the value. The system typically determines this timestamp when it receives a meter reading.

A **Reading** is defined as follows:

> Specific value measured by a meter or other asset. Each Reading is associated with a specific Reading Type.

(IEC 61968-9, pg. 31)

MeterReadings have the following fields:

Field	Cardinality	Description
Meter	1	MRID of the meter providing this reading.
Reading Type	1	Type information for this reading value.
createdAt	1	Timestamp when SAP Cloud for Energy received this reading version
time	1	Timestamp of the reading value
value	0..1	Reading value
reportedDateTime	0..1	(used only when there are detailed auditing requirements) Date and time at which the reading was first delivered to the metering system. The optional reportedDateTime can be used to identify when the reading was reported.

Field	Cardinality	Description
timePeriodStart	0..1	Start of the period for those readings whose type has a time attribute such as 'billing', seasonal' or 'forTheSpecifiedPeriod'.
timePeriodEnd	0..1	End of the period for those readings whose type has a time attribute such as 'billing', seasonal' or 'forTheSpecifiedPeriod'.
source	0..1	System that originally supplied the reading (e.g., customer, AMI system, handheld reading system, another enterprise system, etc.).
reason	0..1	Reason for this reading being taken.
ReadingQualities	0..n	All qualities of this reading

To find a list of values, the system uses Meter, ReadingType, and time.

If we look at the timestamp of the reading value contained in the time field, it is always the time when the reading was taken. When an interval is measured, it is the **end time stamp** of this interval (as you can only know at the end of an interval, what was measured in the interval). When a timestamp is used, it is within a time range from 00:00:00 to 23:59:59. Note that as defined by ISO (International Standards Organization) 8601, the time 24:00 of the current day is the same as 00:00 on the next day. The day starts and ends at 00:00.

Reading qualities are defined as:

> Quality of a specific reading value or interval reading value. Note that more than one quality may be applicable to a given reading. Typically, not used unless problems or unusual conditions occur (i.e., quality for each reading is assumed to be good unless stated otherwise in associated reading quality type). It can also be used with the corresponding reading quality type to indicate that the validation has been performed and succeeded.

(IEC 61968-9, pg. 31)

Reading Qualities fields are

Field	Cardinality	Description
source	0..1	System acting as the source of the quality code.
comment	0..1	Elaboration on the quality code.
timeStamp	0..1	Date and time at which the quality code was assigned or ascertained.
category	1	High-level nature of the reading value quality.
subCategory	1	More specific nature of the reading value quality, as a further sub-categorisation of 'category'.
systemId	1	Identification of the system which has declared the issue with the data or provided commentary on the data.

(category, subCategory and systemId are from Annex D of IEC 61968-9)

It is important to understand that the fields of a meter reading in the data model and the CIM requests are not identical. *Energy Data Services* understands CIM XML elements like

interval readings or interval blocks, but there is no representation of these elements in the data model. *Energy Data Services* can also receive a meter status (including a timestamp) with the CIM message to create meter readings. This status is currently not used (see CIM XML Schema).

As already mentioned in the reading type description, it is possible that *SAP Cloud for Energy* responds to a meter reading request with a different reading type than with the one requested. This can happen because the system selects the reading data with which to respond in a best effort approach.

You request meter readings using a meter identifier (mRID or name) and a reading type (or identifier that can be resolved to a unique reading type such as the identification system code). In addition, it is possible to query readings based on different objects according to the schemas for "GetMeterReadings":

```
 <xs:complexType name="GetMeterReadings">
        <xs:sequence>
            <xs:element name="EndDevice" type="m:EndDevice" minOccurs="0"
maxOccurs="unbounded"/>
            <xs:element name="EndDeviceGroup" type="m:EndDeviceGroup" minOccurs="0"
maxOccurs="unbounded"/>
            <xs:element name="MeterReadings" type="m:MeterReadings" minOccurs="0"/>
            <xs:element name="Reading" type="m:Reading" minOccurs="0"
maxOccurs="unbounded"/>

            <xs:element name="ReadingQuality" type="m:ReadingQuality" minOccurs="0"
maxOccurs="unbounded"/>
            <xs:element name="ReadingType" type="m:ReadingType" minOccurs="0"
maxOccurs="unbounded"/>
            <xs:element name="TimeSchedule" type="m:TimeSchedule" minOccurs="0"
maxOccurs="unbounded"/>
            ...
            <xs:element name="UsagePoint" type="m:UsagePoint" minOccurs="0"
maxOccurs="unbounded"/>
            <xs:element name="UsagePointGroup" type="m:UsagePointGroup" minOccurs="0"
maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:complexType>
```

Figure 11: Schema

It is possible to query readings for groups of end devices (meters), for usage points, or groups of usage points. Please note that according to IEC 61968-900 (pg. 38) EndDevice, EndDeviceGroup, UsagePoint and UsagePointGroup specify the Meters to query, whereas ReadingType, ReadingQuality, TimeSchedule, MeterReadings are filtering criteria against which the returned data are matched.

Working with the Sample Request Messages — 01 CIM Beginner

You want to create a meter to manage 15min electricity consumption time series data. Once the meter is created, you ingest measurement data for the newly created meter.

To verify if everything went well, you read the meter master data and the measurement values.

The last step is to review your data in the *SAP Cloud for Energy* apps.

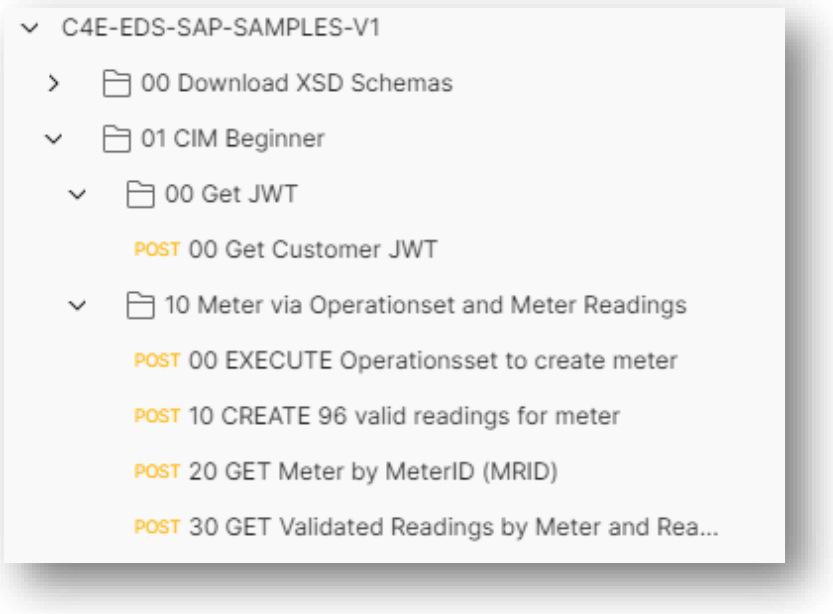


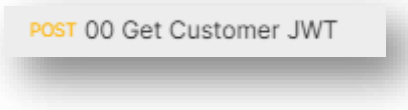
Figure 12: Folder 01 CIM Beginner

Get Customer JWT (Token)

Working with the API only works if you care about the security measures. No API Operation works without a token. The Jason Web Token (JWT) is valid for a certain duration, not every call needs a new token.

To get a JWT, execute the prepared request.

Run: “00 Get Customer JWT” to get your JWT to have access to APIs.



Result: Access token.

Check if the result looks like this:

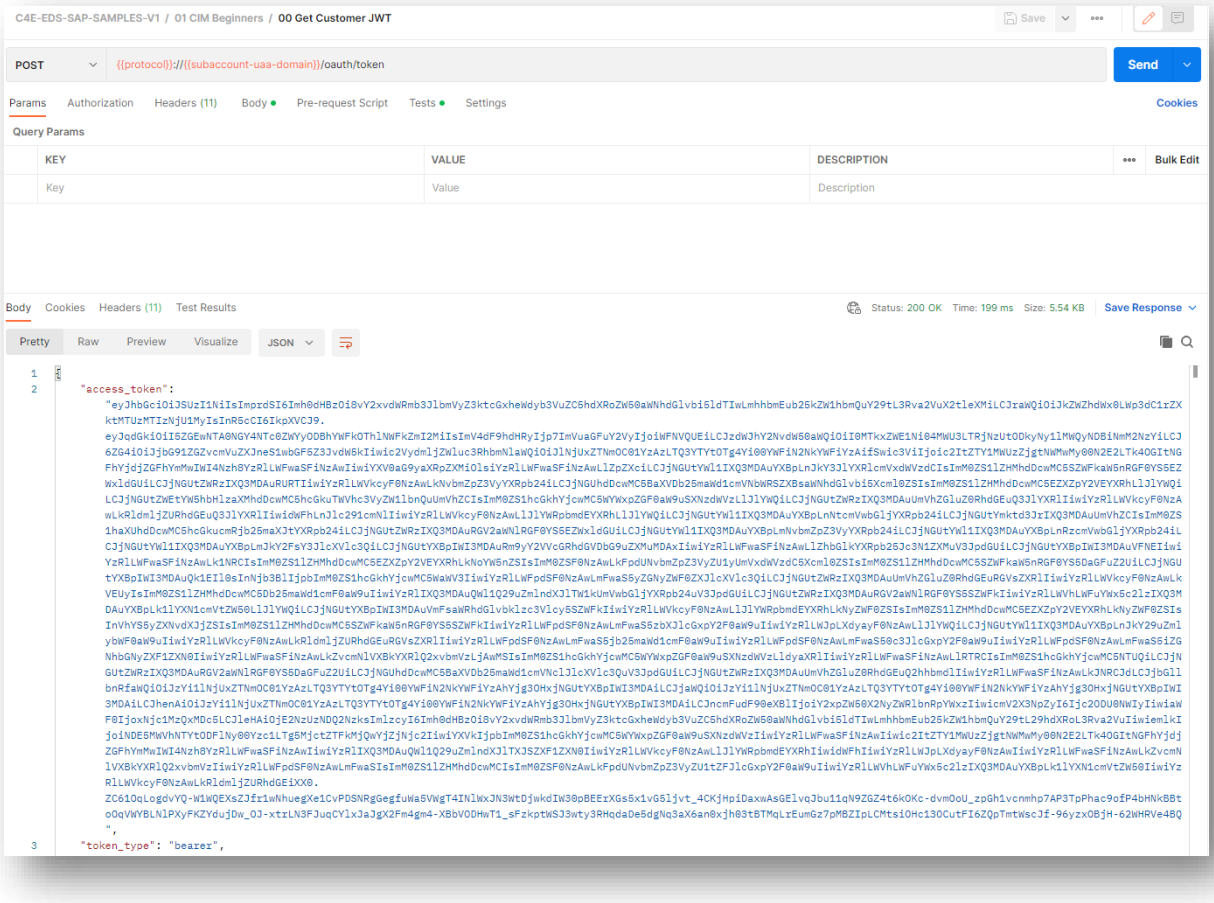


Figure 13: Result of Get Customer JWT Request

In Case of Errors

If you cannot see the token but a message showing an error, check if the Environment is assigned and used, or if the Environment is correctly set up for the following:

- client_id
- client_secret
- subaccount-uaa-domain
- customer_subaccount

Review the Provided Examples

Open the folder “10 Meter via Operation Set and Meter Readings” and review the sample request messages.

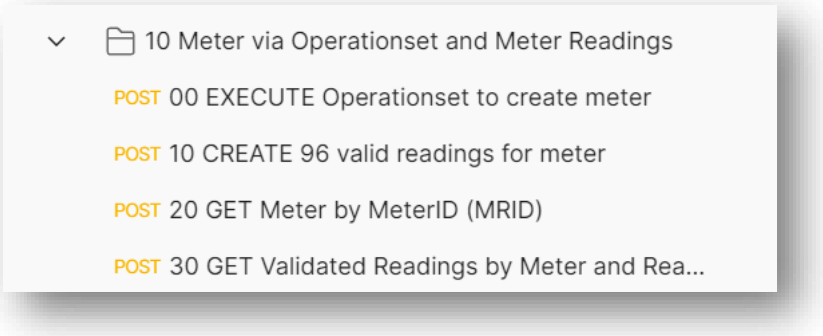


Figure 14: Folder 10 Sample Request Messages

a) **00 EXECUTE Operation Set to create meter**
Create a meter with all related sub-entities.
The meter is configured to have a register to store 15min electricity consumption time series data. (ReadingType 0.0.2.4.1.1.12.0.0.0.0.0.0.0.3.72.0)
Register is configured, address etc. is all maintained.

b) **10 CREATE 96 valid readings for meter**
Create values (96) for an entire day for this newly created meter.
The ingestion of the data will automatically trigger the Validation and Estimation within *SAP Cloud for Energy*. *(What kind of rules are executed is configured in an app: Configure Validation Rules app)*

c) **20 GET Meter by MeterID (MRID)**
Review (get) Meter data

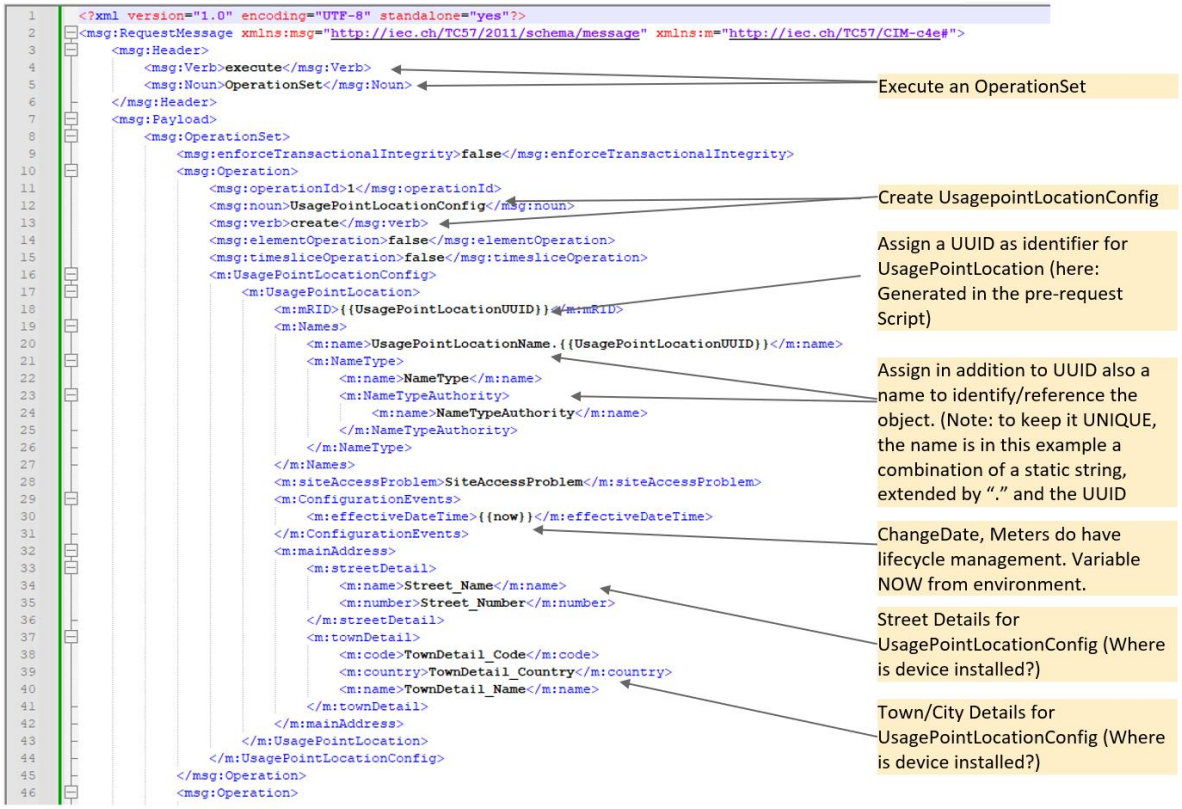
d) **30 GET Validated Readings by Meter and ReadingType**
Review (get) Meter Readings you have ingested, and which passed the *SAP Cloud for Energy* validation & estimation engine.

Execute Operationset to Create Meter

To create your first meter with preconfigured items and maintained relations, use “00 Execute Operationset to create meter”.

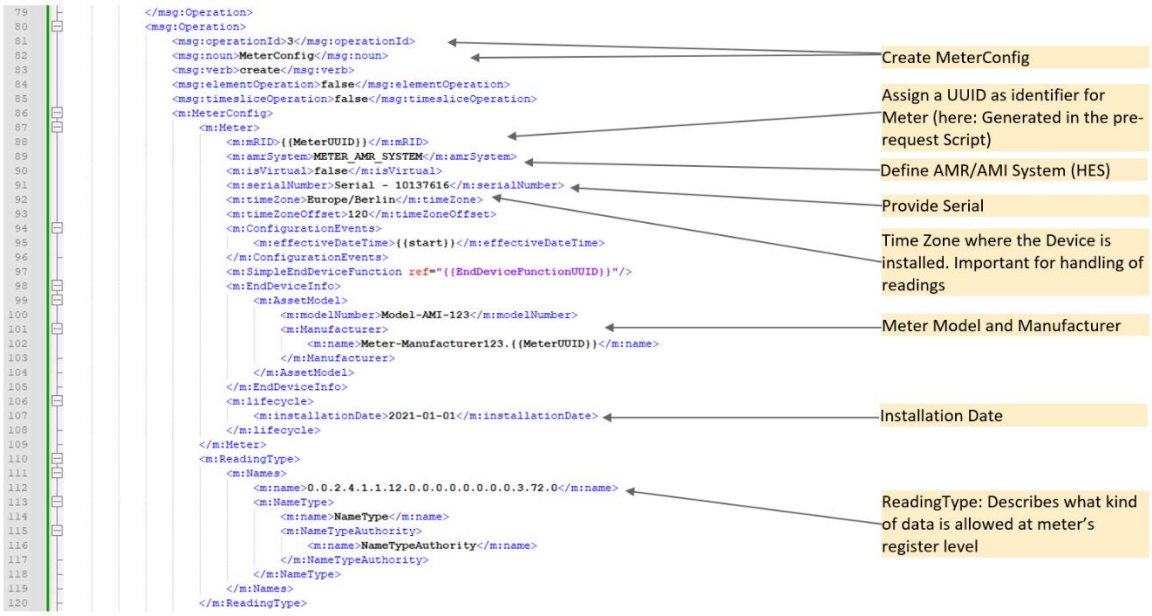
Understand the Operation Set and its main entities:

Part1



Part 2

Part 3




```
</msg:Operation>
<msg:Operation>
    <msg:operationId>3</msg:operationId>
    <msg:noun>MeterConfig</msg:noun>
    <msg:verb>create</msg:verb>
    <msg:elementOperation>false</msg:elementOperation>
    <msg:timesliceOperation>false</msg:timesliceOperation>
    <m:MeterConfig>
        <m:Meter>
            <m:mRID>{{MeterUUID}}</m:mRID>
            <m:amrSystem>METER_AMR_SYSTEM</m:amrSystem>
            <m:isVirtual>false</m:isVirtual>
            <m:serialNumber>Serial - 10137616</m:serialNumber>
            <m:timeZone>Europe/Berlin</m:timeZone>
            <m:timeZoneOffset>120</m:timeZoneOffset>
            <m:ConfigurationEvents>
                <m:effectiveDateTime>{{start}}</m:effectiveDateTime>
            </m:ConfigurationEvents>
            <m:SimpleEndDeviceFunction ref="{{EndDeviceFunctionUUID}}"/>
            <m:EndDeviceInfo>
                <m:AssetModel>
                    <m:modelNumber>Model-AMI-123</m:modelNumber>
                    <m:Manufacturer>
                        <m:name>Meter-Manufacturer123.{{MeterUUID}}</m:name>
                    </m:Manufacturer>
                </m:AssetModel>
            </m:EndDeviceInfo>
            <m:lifecycle>
                <m:installationDate>2021-01-01</m:installationDate>
            </m:lifecycle>
        </m:Meter>
        <m:ReadingType>
            <m:Names>
                <m:name>0.0.2.4.1.1.12.0.0.0.0.0.0.0.3.72.0</m:name>
                <m:NameType>
                    <m:name>NameType</m:name>
                    <m:NameTypeAuthority>
                        <m:name>NameTypeAuthority</m:name>
                    </m:NameTypeAuthority>
                </m:NameType>
            </m:Names>
        </m:ReadingType>
```

Part 4


```
</m:ReadingType>
<m:SimpleEndDeviceFunction>
    <m:mRID>{{EndDeviceFunctionUUID}}</m:mRID>
    <m:Names>
        <m:name>EndDeviceFunctionName.{{EndDeviceFunctionUUID}}</m:name>
        <m:NameType>
            <m:name>SimpleEndDeviceFunctionID</m:name>
            <m:NameTypeAuthority>
                <m:name>EMO_327</m:name>
            </m:NameTypeAuthority>
        </m:NameType>
    </m:Names>
    <m:enabled>true</m:enabled>
    <m:Registers>
        <m:Names>
            <m:name>RegisterName.{{MeterUUID}}</m:name>
            <m:NameType>
                <m:name>UtilitiesMeasurementTaskID</m:name>
                <m:NameTypeAuthority>
                    <m:name>EMO_327</m:name>
                </m:NameTypeAuthority>
            </m:NameType>
        </m:Names>
        <m:isVirtual>true</m:isVirtual>
        <m:leftDigitCount>9</m:leftDigitCount>
        <m:rightDigitCount>3</m:rightDigitCount>
        <m:Channels>
            <m:Names>
                <m:name>ChannelName.{{MeterUUID}}</m:name>
                <m:NameType>
                    <m:name>UtilitiesMeasurementTaskID-Channel</m:name>
                    <m:NameTypeAuthority>
                        <m:name>EMO_327</m:name>
                    </m:NameTypeAuthority>
                </m:NameType>
            </m:Names>
            <m:identificationSystemCode>1-1:1.29.0</m:identificationSystemCode>
            <m:measurementTask>1234task</m:measurementTask>
            <m:slpProfileID>b14c5cab-501b-4035-890c-5d3d02bb31aa</m:slpProfileID>
            <m:veeCode>1004</m:veeCode>
            <m:ReadingType ref="0.0.2.4.1.1.12.0.0.0.0.0.0.0.3.72.0"/>
        </m:Channels>
```

Part 5

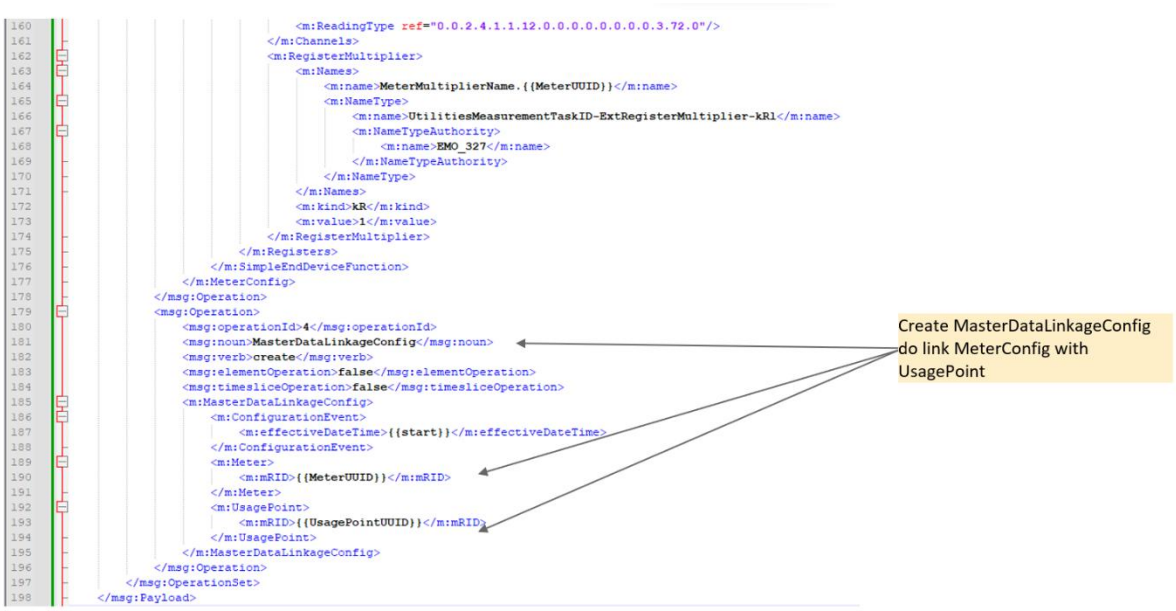


Figure 15: Operation Set and Main Entities

Run the Request.

Check that the result is “ok” with code “0.0” and shows “Success”.

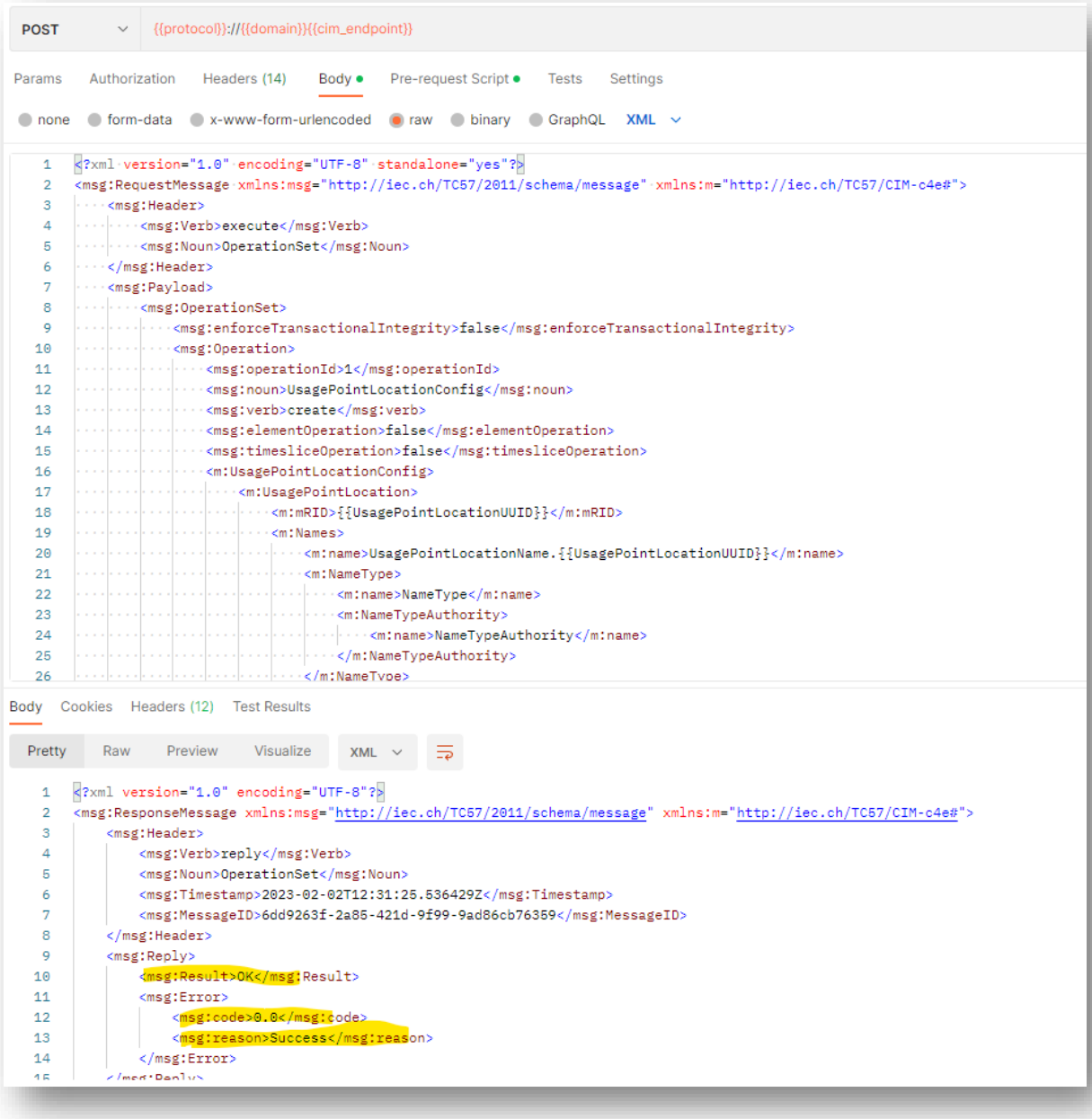
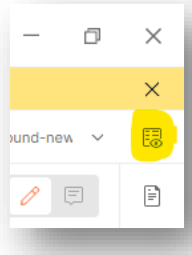


Figure 16: Result of Request

Review the created UUIDs for the objects.

Make use of the "*Environment Quick Look*" or review the "Console"

Environment:



Understand the UUIDs.

This is just an example; with each run, the UUIDs will be different!

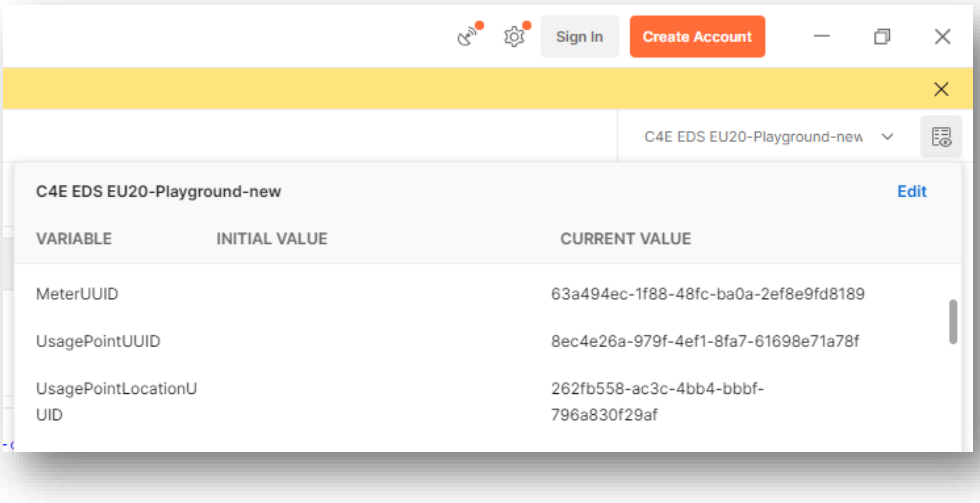


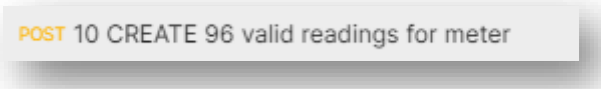
Figure 17: Environment

Console:

The final payload (request message) is shown in the Console.
You can review & copy the values in the variables.

Figure 18: Console

Ingest Measurement Values for the Newly Created Device



Let's understand what is done here:

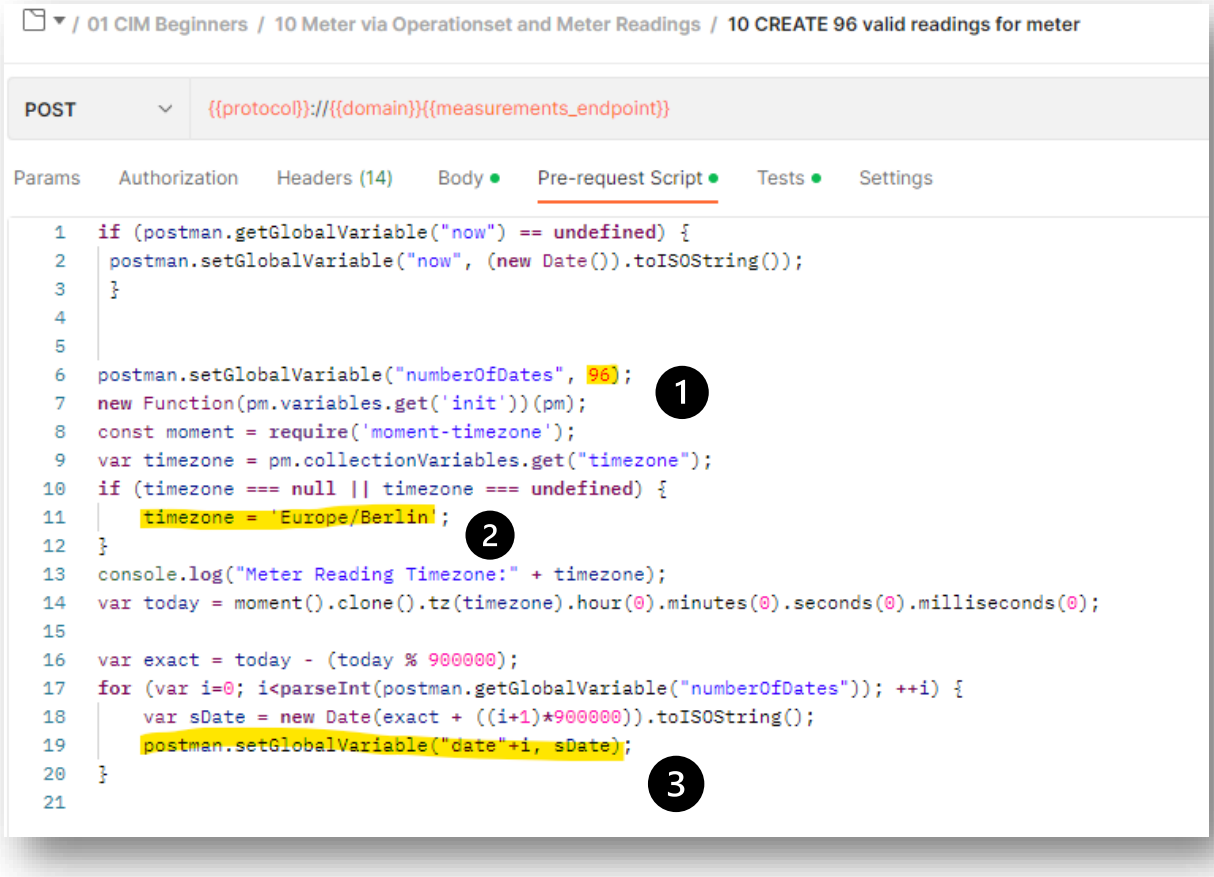


Figure 19: Pre-Request Script

1) 96 values are considered.
2) Time zone is set to "**Europe/Berlin**".
3) Date0 to Date95 is created (to have 96 values).

Open the message body:

As an example, here are two out of the 96 blocks to ingest measurement values.

The variable {{date0}} highlighted in yellow is replaced with the result of the pre-request script.

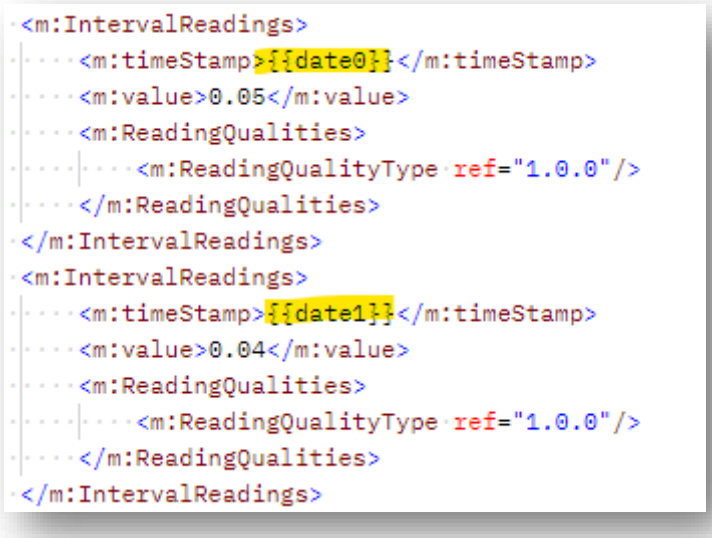


Figure 20: Variable in Message Body

The Reading Type defines the type of data (commodity, interval length, unit of measurement, etc.).

For more information about the reading type, see the list of supported reading types in the product documenation on the SAP Help Portal.

The mRID refers to the meter for which the measurement data will be ingested.

{{MeterUUID}} refers to the environment variable – in this case the UUID from the previous step was written into the environment and can now be used.

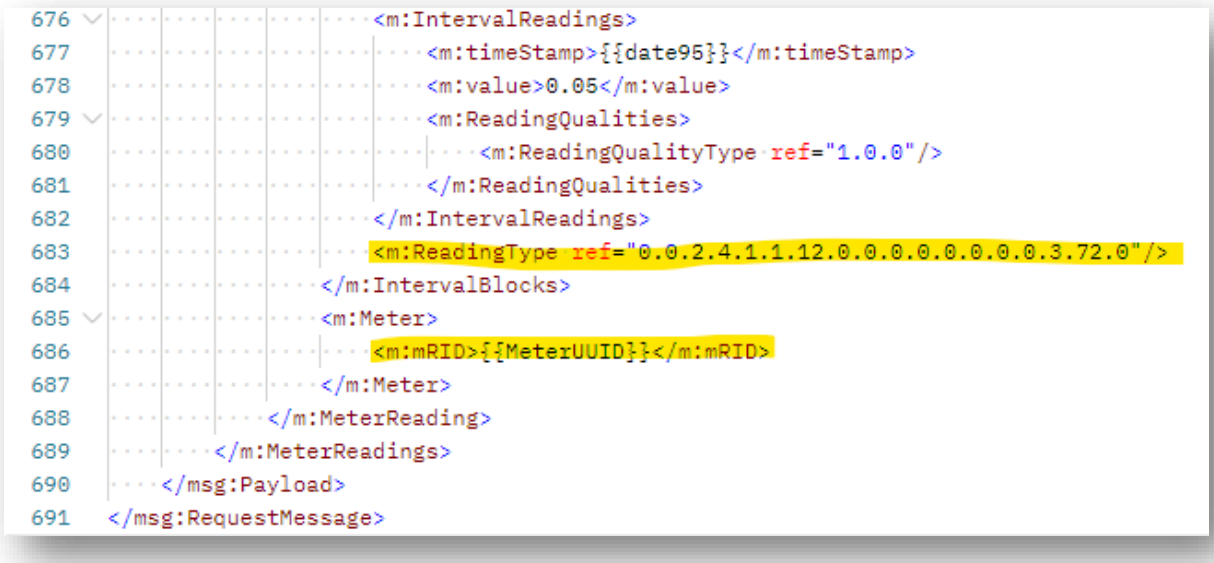


Figure 21: Reading Type and Variable in Message Body

Execute the request and review if the result is “ok” with code “0.0” and shows “Success”.

Get Meter by MeterId

POST 20 GET Meter by MeterID (MRID)

The goal is to read (get) Meter data by UUID.

The {{MeterUUID}} highlighted in yellow represents the meter.

In this scenario, the UUID is taken from the environment to make it easier and failsafe.

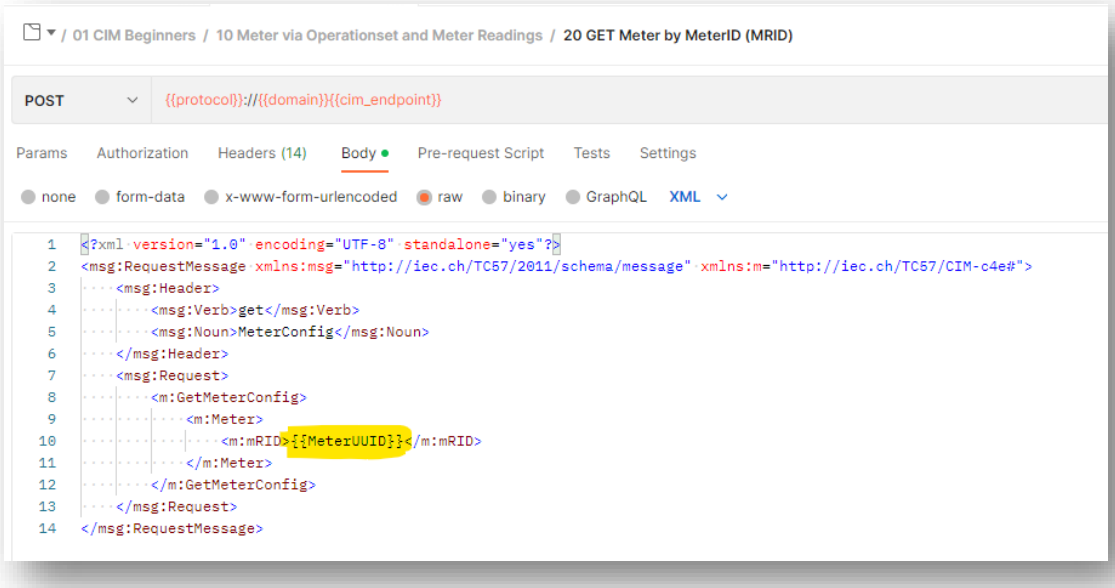


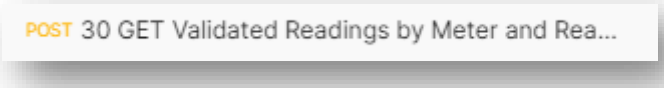
Figure 22: Meter mRID in Message Body

Run the request

The result of this API call shows all attributes.

Figure 23: Result Fragment

Get Validated Readings



Open the request body:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<msg:RequestMessage xmlns:msg="http://iec.ch/TC57/2011/schema/message" xmlns:m="http://iec.ch/TC57/CIM-c4e#">
    <msg:Header>
        <msg:Verb>get</msg:Verb>
        <msg:Noun>MeterReadings</msg:Noun>
    </msg:Header>
    <msg:Request>
        <m:GetMeterReadings>
            <m:Meter>
                <m:mRID>{{MeterUUID}}</m:mRID>
            </m:Meter>
            <m:ReadingType>
                <m:Names xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="m:ReadingTypeName">
                    <m:name>0.0.2.4.1.1.12.0.0.0.0.0.0.0.3.72.0</m:name>
                    <m:NameType>
                        <m:name>NameType</m:name>
                        <m:NameTypeAuthority>
                            <m:name>NameTypeAuthority</m:name>
                        </m:NameTypeAuthority>
                    </m:NameType>
                </m:Names>
            </m:ReadingType>
            <m:TimeSchedule>
                <m:scheduleInterval>
                    <m:end>{{end}}</m:end>
                    <m:start>{{start}}</m:start>
                </m:scheduleInterval>
            </m:TimeSchedule>
        </m:GetMeterReadings>
    </msg:Request>
</msg:RequestMessage>
```

Figure 24: Request Body

1) {{MeterUUID}} refers to the meter from which we want to query data. In our example, the variable is taken from the Environment.

2) 0.0.2.4.1…. is the Reading Type we are looking for.

3) End/start is the period for which we want to read the data.

Result

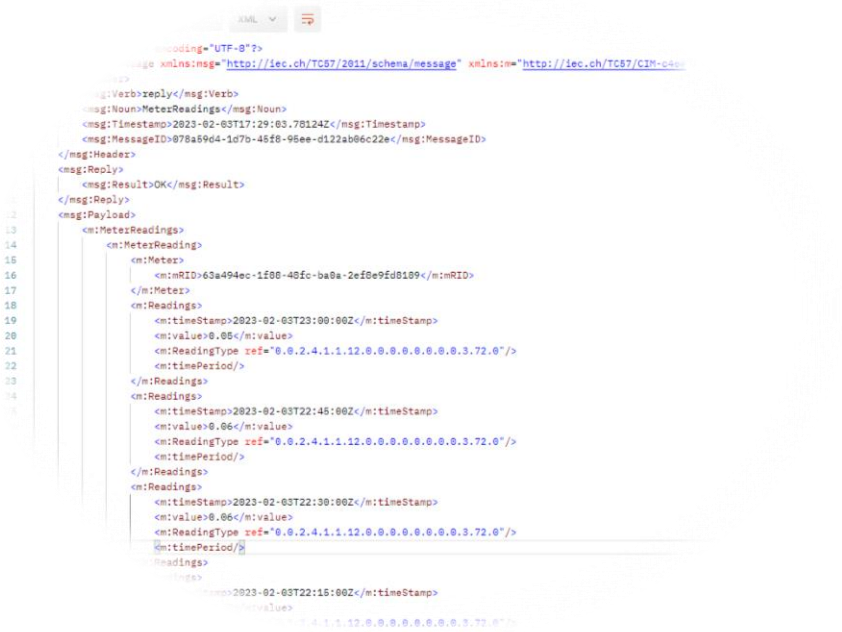


Figure 25: Result Fragment

Review Meter and Measurement Data in the App

In this section, we will step away from the API and open the *SAP Cloud for Energy* Launchpad.

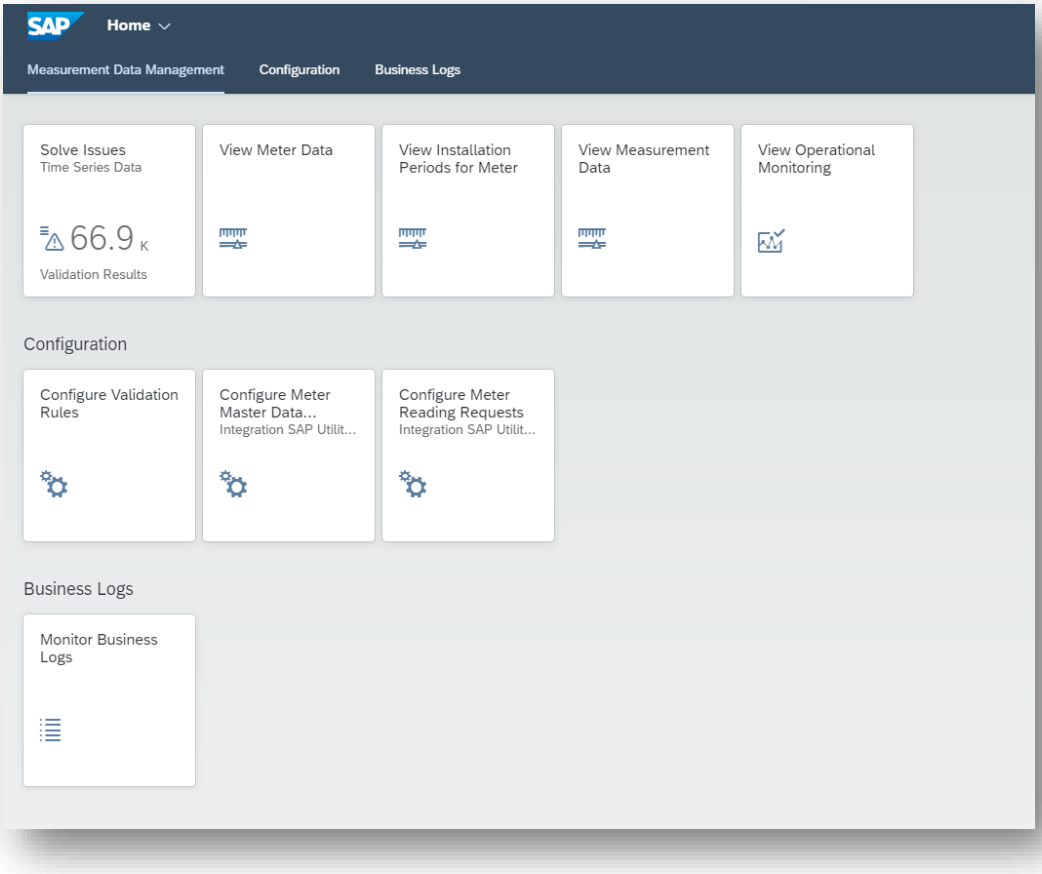
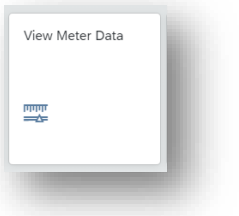


Figure 26: SAP Cloud for Energy Launchpad

In the View Measurement Data app, review the Meter you created.

Remember: In this example it was the meter with UUID 63a494ec-1f88-48fc-ba0a-2ef8e9fd8189 *(Adjust this to your UUIDs!)*

Click the tile to open the *View Meter Data* app.



Choose *Adapt filters* and select the Meter ID.

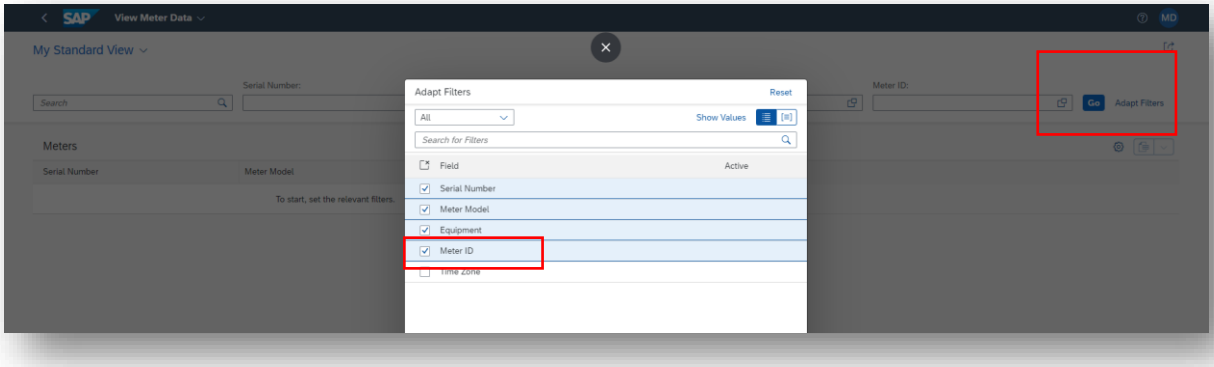
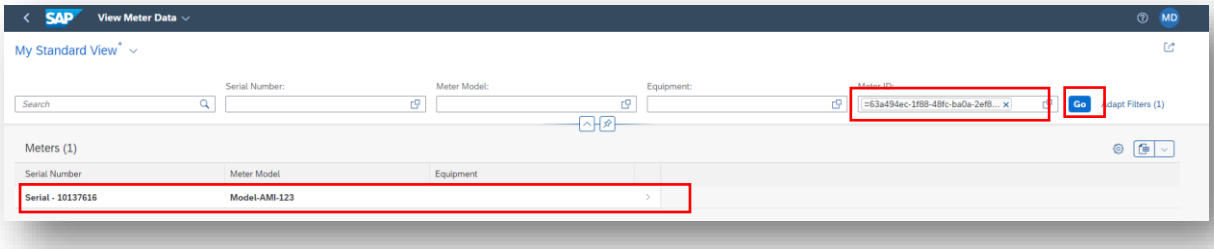


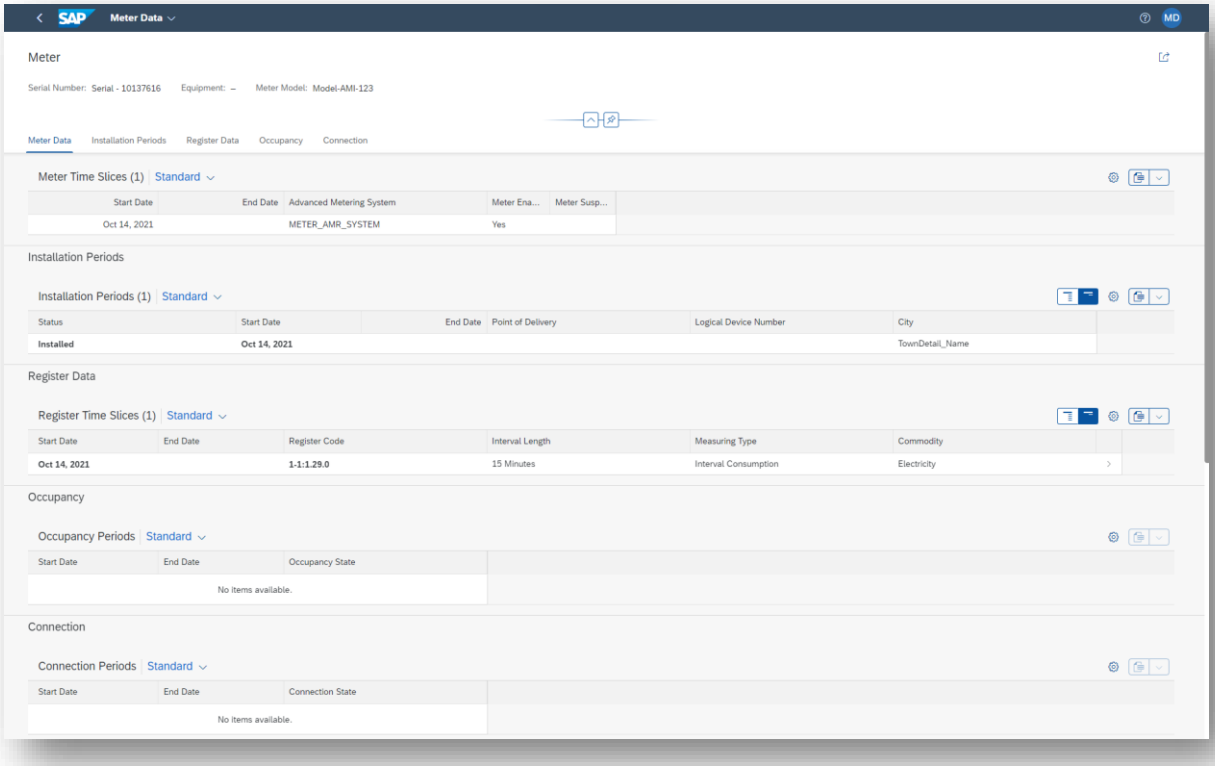
Figure 27: SAP Cloud for Energy Launchpad

This adds the *Meter ID* filter to the screen so that you can search for the meter.

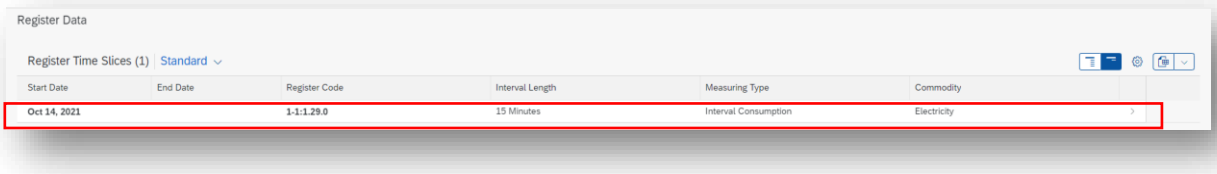
Enter the UUID, choose *Go* and find the meter in the result list.

To display the meter attributes, choose > at the end of the table row:





To access the measurement data, choose > at the end of the table row:



App View Measurement Data

Open the *View Measurement Data* app, which shows the data in a chart and a table.

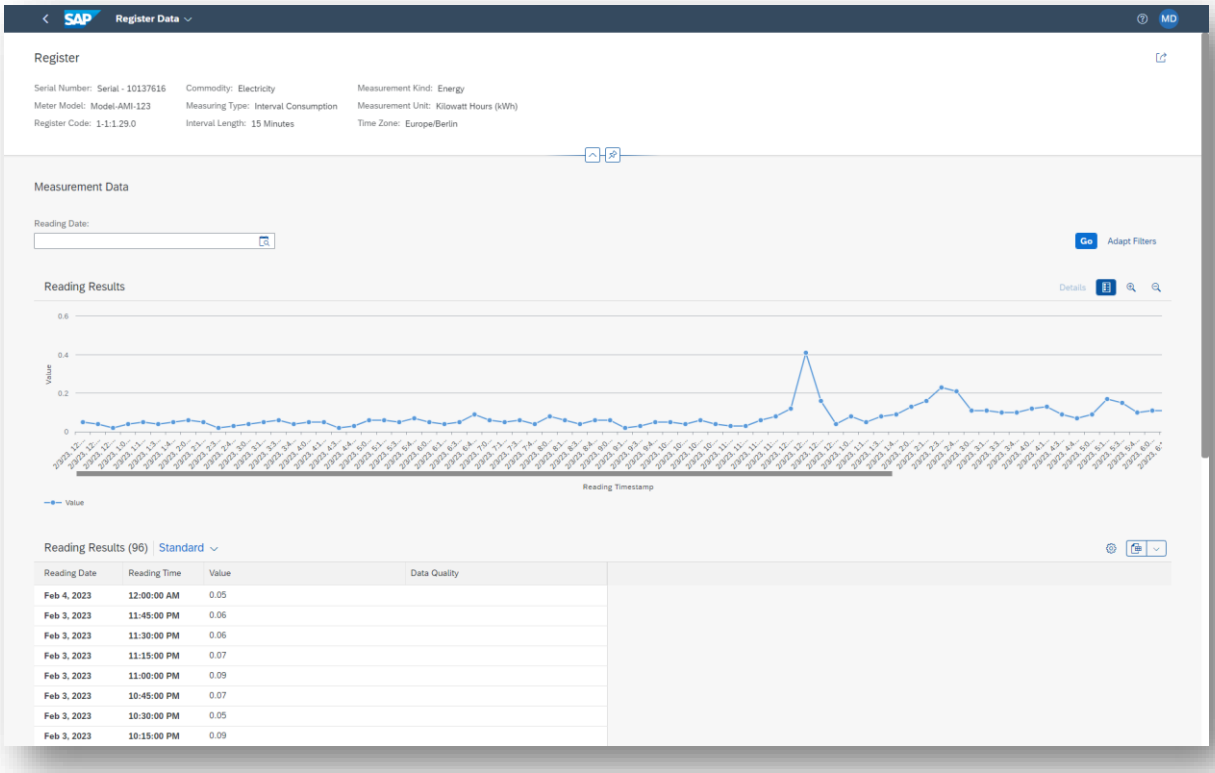


Figure 27: View Measurement Data App

Note: If you have used different timestamps, you might need to select a reading date (range) first to find the data that you have ingested.

Working with the Sample Request Messages — 02 CIM Advanced

You will understand the object relations and create item by item to have a full meter installed and configured. This meter is configured with a register to hold hourly meter readings. You ingest hourly water meter reading timeseries values and m^3 as unit of measurement. You ingest values for an entire day.

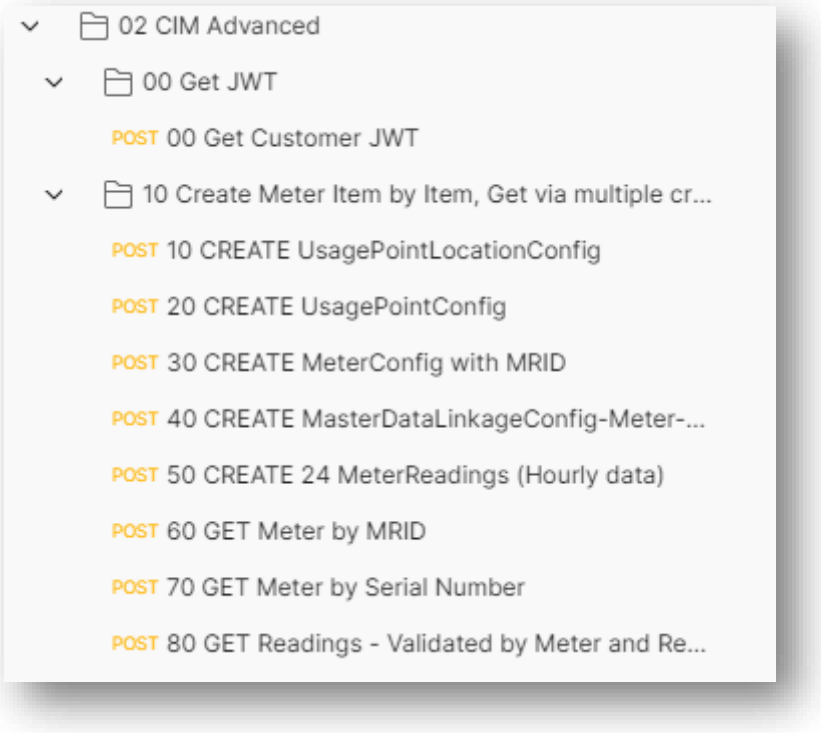
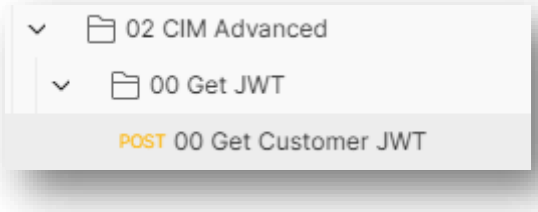


Figure 28: Folder 02 CIM Advanced

Get Customer JWT (Token)

Do not start without a valid JWT.

Use 00 GET Customer JWT to receive the token.



UsagePointLocationConfig

This element holds address data and explains where the meter is physically installed.

POST 10 CREATE UsagePointLocationConfig

Run the request to create the UsagePointLocationConfig.

As a result, the item will be created and the UUID of this object will be stored in the environment.

(The UUID is required in the next steps, and since it is stored in the environment, you won't need to remember it or enter it manually later).

If the object is created successfully, you will receive a positive response and you will find the UUID in the environment.

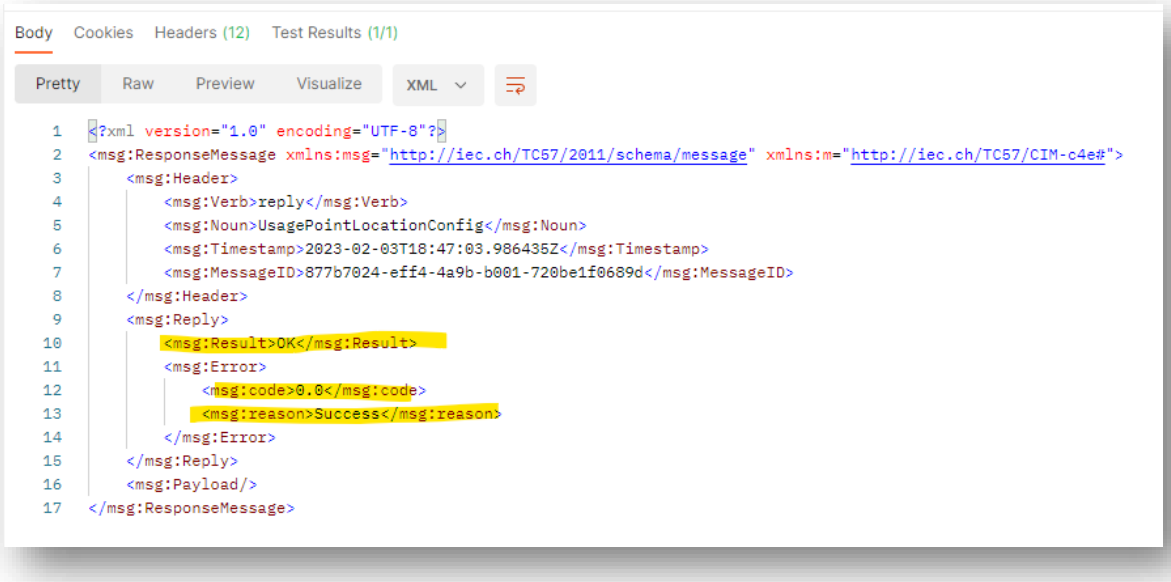


Figure 29: Response Message

Environment:

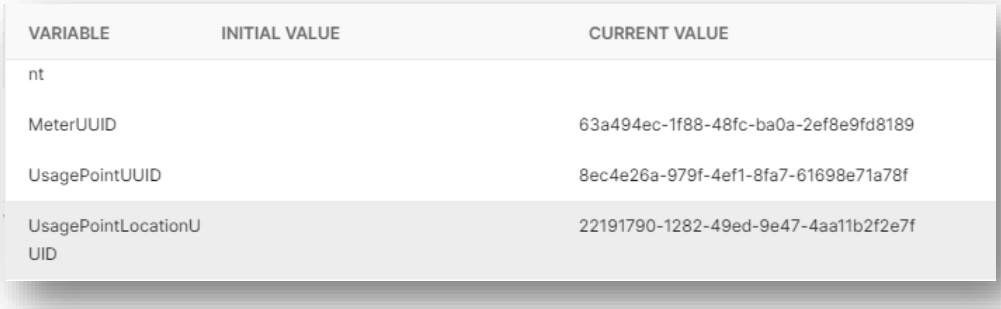


Figure 30: Environment

UsagePointConfig

POST 20 CREATE UsagePointConfig

This element refers to the UsagePointLocation (Config) highlighted in yellow.

Note: In this example, the {{UsagePointLoctionUUID}} is taken from the environment (it was written to the environment in the previous step).

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<msg:RequestMessage xmlns:msg="http://iec.ch/TC57/2011/schema/message" xmlns:m="http://iec.ch/TC57/CIM-c4e#">
    <msg:Header>
        <msg:Verb>create</msg:Verb>
        <msg:Noun>UsagePointConfig</msg:Noun>
    </msg:Header>
    <msg:Payload>
        <m:UsagePointConfig>
            <m:UsagePoint>
                <m:mRID>{{UsagePointUUID}}</m:mRID>
                <m:Names>
                    <m:name>UsagePointConfig.{{UsagePointUUID}}</m:name>
                    <m:NameType>
                        <m:name>NameType</m:name>
                        <m:NameTypeAuthority>
                            <m:name>NameTypeAuthority</m:name>
                        </m:NameTypeAuthority>
                    </m:NameType>
                </m:Names>
                <m:amiBillingReady>AmiBillingReady</m:amiBillingReady>
                <m:checkBilling>false</m:checkBilling>
                <m:isVirtual>false</m:isVirtual>
                <m:minimalUsageExpected>false</m:minimalUsageExpected>
                <m:readCycle>ReadCycle</m:readCycle>
                <m:readRoute>ReadRoute</m:readRoute>
                <m:ConfigurationEvents>
                    <m:effectiveDateTime>{{start}}</m:effectiveDateTime>
                </m:ConfigurationEvents>
                <m:UsagePointLocation>
                    <m:mRID>{{UsagePointLocationUUID}}</m:mRID>
                </m:UsagePointLocation>
            </m:UsagePoint>
        </m:UsagePointConfig>
    </msg:Payload>
</msg:RequestMessage>
```

Figure 31: Request Message

Run the request and check if it was successful (response OK, code 0.0, Success).
Important UUIDs are also stored in the environment for further processing.

MeterConfig

POST 30 CREATE MeterConfig with MRID

This element creates the meter. This includes the serial number, time zone, manufacturer, AMI system, channels, registers, and others.
The reading type is configured. In this case: hourly meter reading time series values, commodity water, unit of measure m^3.

Remember the serial number you have provided in the payload, e.g.,
`<m:serialNumber>Serial-10137616</m:serialNumber>`
You will need it later.

Run the request, check if it was successful and check if the MeterUUID (mrID) was written in the environment.

The MeterUUID could look like this: `68ebc19f-caf8-437c-8adf-bfab565e9aaf`

MasterDataLinkageConfig

POST 40 CREATE MasterDataLinkageConfig-Meter-...

This entity creates the linkage between Meter and UsagePoint.
Previously created items – identified by their UUID are linked.
Run the request and check if it was successful.

Result

The meter and its dependent items are now fully configured.

Create Hourly Meter Readings (Meter Reading Time Series Values)

Create 24 values (hourly data for an entire day)

POST 50 CREATE 24 MeterReadings (Hourly data)

- This element creates 24 values.
- Reading Type: hourly water meter reading time series values in m^3.
- Values are assigned to the meter that you previously created.

Run the request and check if it was successful.

Review Meter Master Data and Measurement Data in an App

Open the *SAP Cloud for Energy* Launchpad.

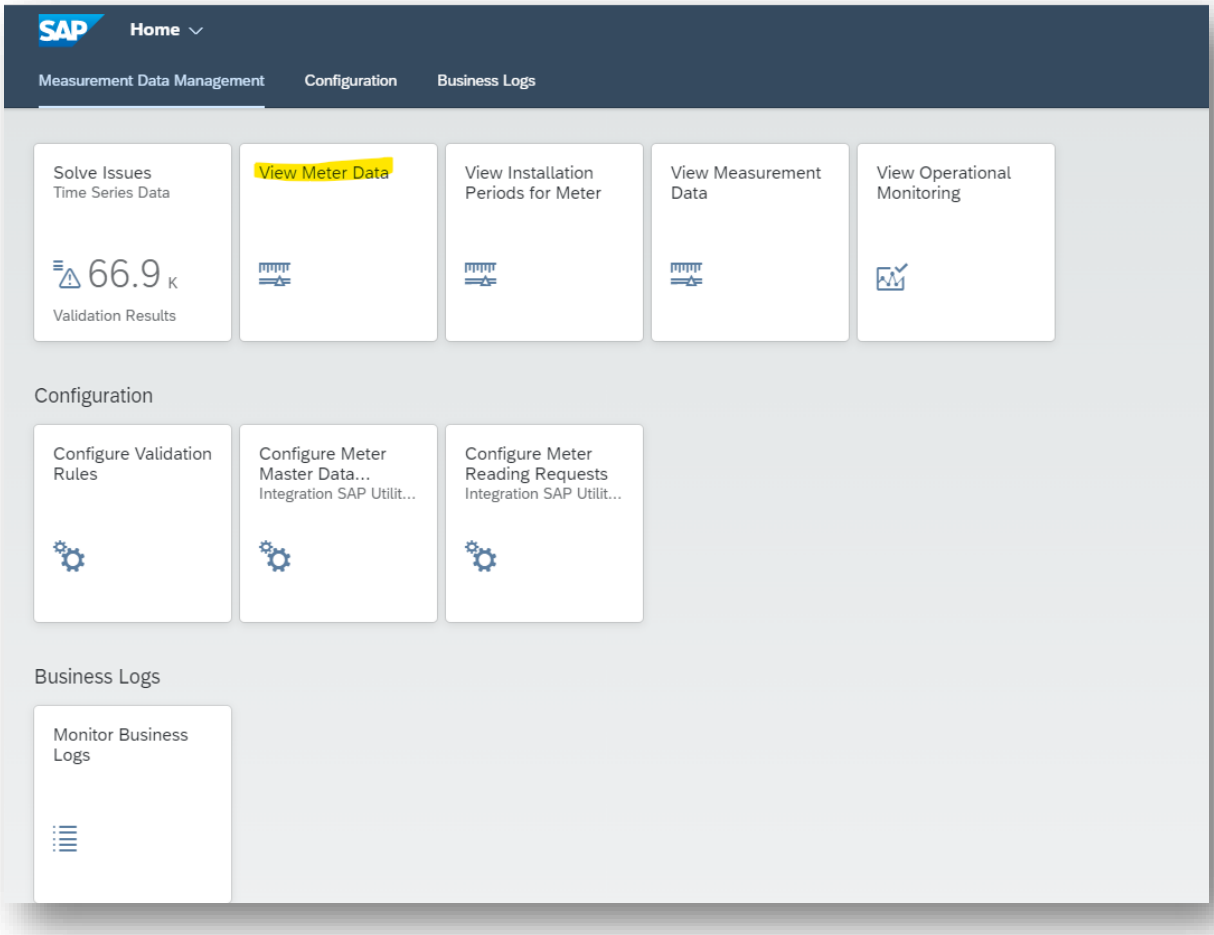


Figure 32: SAP Cloud for Energy Launchpad

In the app, review the measurement data you ingested for the meter.

Remember: In this example it was the meter with UUID
68ebc19f-caf8-437c-8adf-bfab565e9aaf
(Adjust this to your UUIDs!)

Click the tile to open the *View Meter Data* app.

Enter the Meter ID of the meter you have created in the step “MeterConfig”. (e.g., 68ebc19f-caf8-437c-8adf-bfab565e9aaf) and choose *Go.*

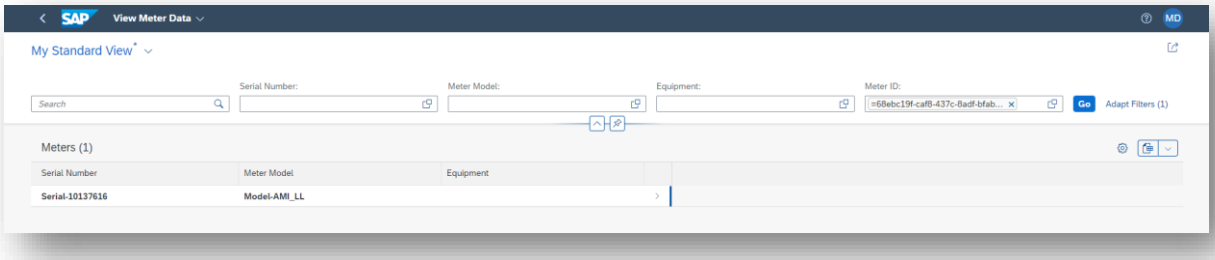


Figure 33: View Meter Data App

To review the meter data, choose > at the end of the table row.

The serial number, channel/register details are displayed along with other information such as location and AMR system:

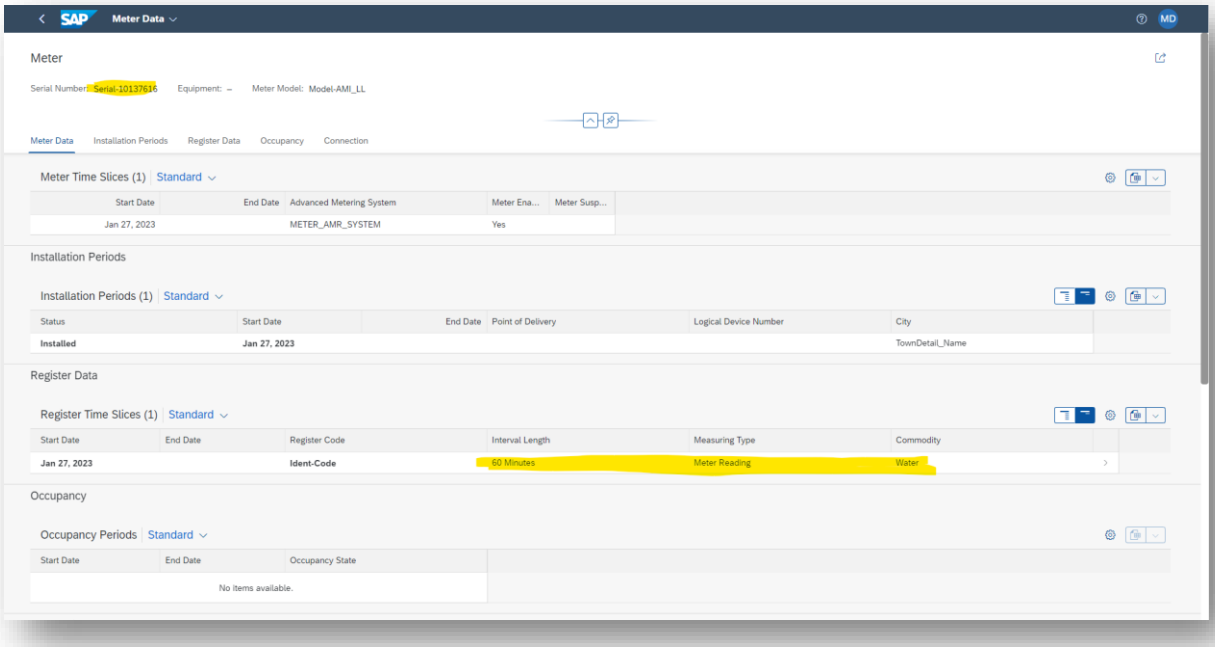


Figure 34: View Meter Data App

To view the measurement data, choose > at the end of the row with the register data.

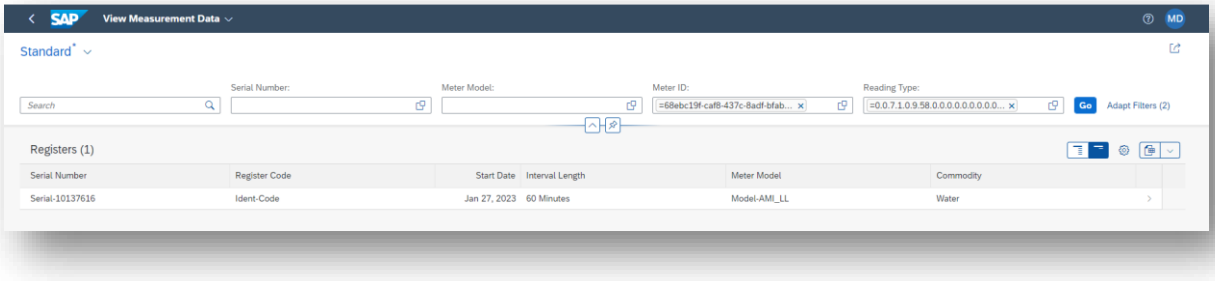


Figure 35: View Measurement Data App

To view the data in a table or chart, choose > at the end of the row with the register data.

Note: If you have ingested data for a different period than “today”, you might need to select the reading date (range) first to find the data that you have ingested.

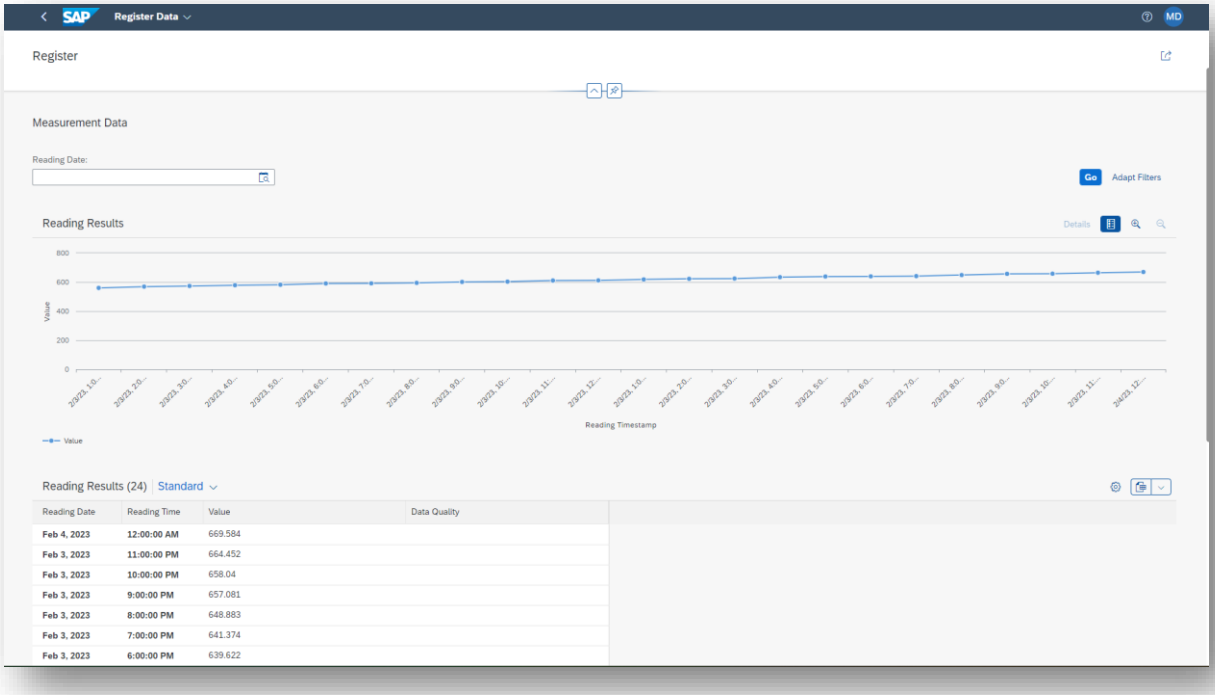


Figure 36: View Measurement Data App

Get Meter by MRID (Meter UUID)

POST 60 GET Meter by MRID

You read the previously created meter.

The item highlighted in yellow is taken from the environment. Here you can also add the UUID of the meter.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<msg:RequestMessage xmlns:msg="http://iec.ch/TC57/2011/schema/message" xmlns:m="http://iec.ch/TC57/CIM-c4e#">
    <msg:Header>
        <msg:Verb>get</msg:Verb>
        <msg:Noun>MeterConfig</msg:Noun>
    </msg:Header>
    <msg:Request>
        <m:GetMeterConfig>
            <m:Meter>
                <m:mRID>{{MeterUUID}}</m:mRID>
            </m:Meter>
        </m:GetMeterConfig>
    </msg:Request>
</msg:RequestMessage>
```

Figure 37: Meter mRID in Message Body

Run the request and review the response.
Response should be OK and contain the attributes of the meter.

Figure 38: Result Fragment

Get Meter by Serial Number

POST 70 GET Meter by Serial Number

Read the meter by serial number, not by Meter UUID.

This request to retrieve meter details does not expect a UUID, but the serial number:

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<msg:RequestMessage xmlns:msg="http://iec.ch/TC57/2011/schema/message" xmlns:m="http://iec.ch/TC57/CIM-c4e#">
    <msg:Header>
        <msg:Verb>get</msg:Verb>
        <msg:Noun>MeterConfig</msg:Noun>
    </msg:Header>
    <msg:Request>
        <m:GetMeterConfig>
            <m:Meter>
                <m:serialNumber>Serial - 10137616</m:serialNumber>
            </m:Meter>
        </m:GetMeterConfig>
    </msg:Request>
</msg:RequestMessage>
```

Figure 39: Serial Number in Message Body

When you created the meter, it was mentioned that you should remember its serial number. Now you need this serial number to read the meter data.

Get Meter Readings (Time Series Data)

POST 80 GET Readings - Validated by Meter and Re... •••

You have ingested measurement data and want to read it.

You need to enter the meter UUID and the reading type in the request message.

In addition, the period (start/end) is also a required parameter.

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<msg:RequestMessage xmlns:msg="http://iec.ch/TC57/2011/schema/message" xmlns:m="http://iec.ch/TC57/CIM-c4e#">
    <msg:Header>
        <msg:Verb>get</msg:Verb>
        <msg:Noun>MeterReadings</msg:Noun>
    </msg:Header>
    <msg:Request>
        <m:GetMeterReadings>
            <m:Meter>
                <m:mRID>{{MeterUUID}}</m:mRID>
            </m:Meter>
            <m:ReadingType>
                <m:Names xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="m:ReadingTypeName">
                    <m:name>0.0.7.1.0.9.58.0.0.0.0.0.0.0.0.0.42.0</m:name>
                    <m:NameType>
                        <m:name>NameType</m:name>
                        <m:NameTypeAuthority>
                            <m:name>NameTypeAuthority</m:name>
                        </m:NameTypeAuthority>
                    </m:NameType>
                </m:Names>
            </m:ReadingType>
            <m:TimeSchedule>
                <m:scheduleInterval>
                    <m:end>{{end}}</m:end>
                    <m:start>{{start}}</m:start>
                </m:scheduleInterval>
            </m:TimeSchedule>
        </m:GetMeterReadings>
    </msg:Request>
</msg:RequestMessage>
```

Figure 40: ScheduleInterval in Message Body

Result: Measurement values are returned for the selected period, meter, and reading type:

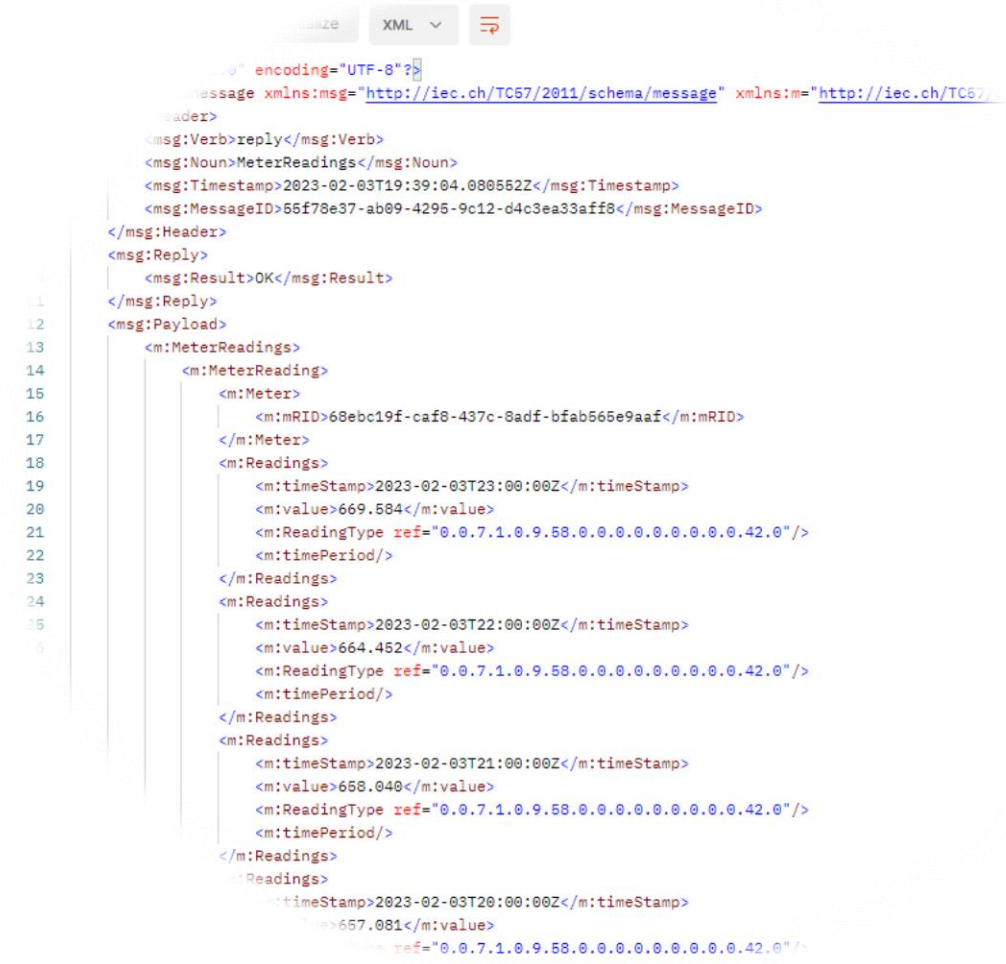


Figure 41: Result Fragment

Working with the Sample Request Messages — 03 CIM Expert

In this section, you will review and execute more complex payloads.

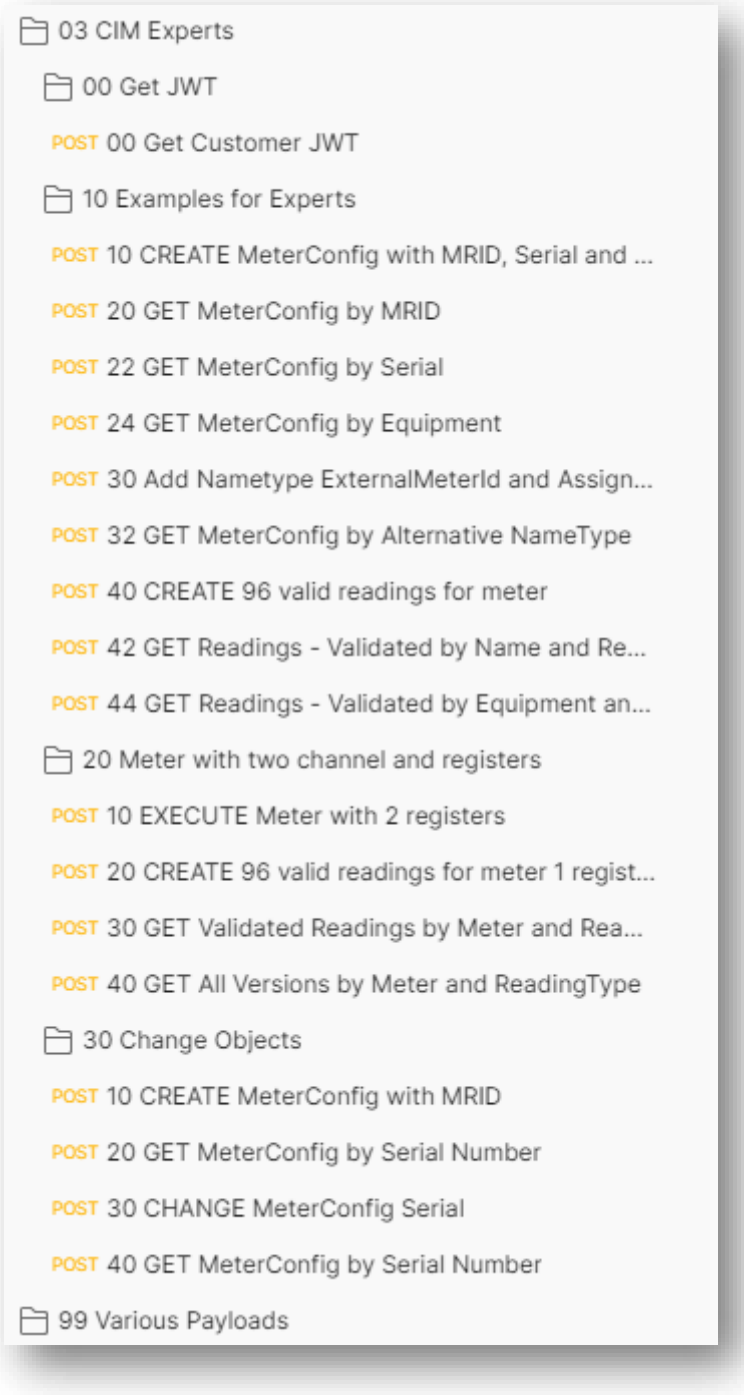


Figure 42: Folder 03 CIM Experts

Objects are identified by UUIDs. But there are more options. You will now get familiar with alternative keys and identifiers.

Alternative Keys & Identifiers — Naming and Addressing of Objects

The following CIM objects can be identified by their **mRID or names**:

- Meter
- UsagePoint
- UsagePointLocation
- RegisteredResource
- ReadingProfile
- Channel
- MeterMultiplier
- ReadingType
- Register
- SimpleEndDeviceFunction

Note: Only UUIDs are accepted as mRIDs.

When an mRID and names are specified, only the mRID is used and names are ignored.

Names are only used when no mRID is specified.

Properties of Names

- Names must be unambiguous:
  It is not possible to use the same name for two objects of the same type, such as a meter, with different mRIDs.
- Names are time independent.
  If a name is assigned to an object, the name is automatically valid for the whole lifespan of the object. It is, for example, not possible to change the name of a single time slice of a meter with a change request.
- Names are initially created along with the corresponding entity, such as a meter, via MeterConfig.
- Name changes are, however, only allowed via ObjectNamesConfig.

Names and mRIDs

- An mRID is an inherent attribute of an object.
  mRIDs cannot be changed or deleted.
- Name is an optional attribute of an object.
  Names can be created, changed, or deleted.

Get Customer JWT (Token)

Do not start without a valid JWT.
Use 00 GET Customer JWT to receive the token.

POST 00 Get Customer JWT

Examples for Experts — Folder 10

Create a meter with UUID, Serial and Equipment

You want to create a meter with a Meter ID (UUID), but also add a serial number and equipment number.

Attribute	Example Value	Comment
Meter ID (UUID)	<Generated via Pre-Request Script>	Mandatory attribute, meter cannot exist without Meter - UUID
Serial number	`123454321 (fix in this example)`	Optional attribute
Equipment	`2312731303 (fix in this example)`	Optional attribute

Open the example, but **do not run it!**

POST 10 CREATE MeterConfig with MRID, Serial and ...

Understand the payload:

Serial Number (here 123454321) is shown in this color in the request example.
This is a standard predefined attribute.

Equipment (here 2312731303) is shown in this color in the request example and configured as Name and NameType.

Note:
For equipment, the NameType is fix: “UtilitiesDeviceID”.

(If you replicated data from SAP S/4HANA Utilities to *SAP Cloud for Energy* using the built-in device replication, the equipment will be stored as UtilitiesDeviceID).

```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<msg:RequestMessage xmlns:msg="http://iec.ch/TC57/2011/schema/message" xmlns:m="http://iec.ch/TC57/CIM-c4e#">
    <msg:Header>
        <msg:Verb>create</msg:Verb>
        <msg:Noun>MeterConfig</msg:Noun>
    </msg:Header>
    <msg:Payload>
        <m:MeterConfig>
            <m:Meter>
                <m:mRID>{{MeterUUID}}</m:mRID>
                <m:Names>
                    <m:name>2312731303</m:name>
                    <m:NameType>
                        <m:name>UtilitiesDeviceID</m:name>
                        <m:NameTypeAuthority>
                            <m:name>EMO_327</m:name>
                        </m:NameTypeAuthority>
                    </m:NameType>
                </m:Names>
                <m:amrSystem>METER_AMR_SYSTEM</m:amrSystem>
                <m:isVirtual>false</m:isVirtual>
                <m:serialNumber>123454321</m:serialNumber>
                <m:timeZone>Europe/Berlin</m:timeZone>
                <m:timeZoneOffset>120</m:timeZoneOffset>
                <m:ConfigurationEvents>
                    <m:effectiveDateTime>{{start}}</m:effectiveDateTime>
                </m:ConfigurationEvents>
                <m:SimpleEndDeviceFunction ref="{{EndDeviceFunctionUUID}}"/>
                <m:EndDeviceInfo>
                    <m:AssetModel>
                        <m:modelNumber>Model-AMI-123</m:modelNumber>
                        <m:Manufacturer>
                            <m:name>Manufacturer.{{MeterUUID}}</m:name>
                        </m:Manufacturer>
```

Figure 43: SerialNumber and Equipment in Message Body

Run the request and check if it was successful.

Review the MeterUUID via Environment/Console.
This number is UNIQUE, and with each run, a new UUID is created. To continue the example, let's say MeterUUID is 249a01f4-9ce8-4e42-bee8-ab4422502530

20 - 24 Get MeterConfig by MRID and Name

Meter is created (earlier step).
Goal: You want to get the meter by certain criteria.
Read the meter via MeterUUID.

20 Get MeterConfig by MRID

You want to read the meter data.

POST 20 GET MeterConfig by MRID

Use the MeterUUID to get the MeterConfig data.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<msg:RequestMessage xmlns:msg="http://iec.ch/TC57/2011/schema/message" xmlns:m="http://iec.ch/TC57/CIM-c4e#">
    <msg:Header>
        <msg:Verb>get</msg:Verb>
        <msg:Noun>MeterConfig</msg:Noun>
    </msg:Header>
    <msg:Request>
        <m:GetMeterConfig>
            <m:Meter>
                <m:mRID>{{MeterUUID}}</m:mRID>
            </m:Meter>
        </m:GetMeterConfig>
    </msg:Request>
</msg:RequestMessage>
```

Figure 44: Meter mRID in Message Body

Run the request and check if it was successful.

22 Get MeterConfig by Serial

POST 22 GET MeterConfig by Serial

Use the Serial number to get the MeterConfig data.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<msg:RequestMessage xmlns:msg="http://iec.ch/TC57/2011/schema/message" xmlns:m="http://iec.ch/TC57/CIM-c4e#">
    <msg:Header>
        <msg:Verb>get</msg:Verb>
        <msg:Noun>MeterConfig</msg:Noun>
    </msg:Header>
    <msg:Request>
        <m:GetMeterConfig>
            <m:Meter>
                <m:serialNumber>123454321</m:serialNumber>
            </m:Meter>
        </m:GetMeterConfig>
    </msg:Request>
</msg:RequestMessage>
```

Figure 45: SerialNumber in Message Body

Run the request and check if it was successful.

24 Get MeterConfig by Equipment

POST 24 GET MeterConfig by Equipment

Use the Equipment number to get the MeterConfig data.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<msg:RequestMessage xmlns:msg="http://iec.ch/TC57/2011/schema/message" xmlns:m="http://iec.ch/TC57/CIM-c4e#">
    <msg:Header>
        <msg:Verb>get</msg:Verb>
        <msg:Noun>MeterConfig</msg:Noun>
    </msg:Header>
    <msg:Request>
        <m:GetMeterConfig>
            <m:Meter>
                <m:Names>
                    <m:name>2312731303</m:name>
                    <m:NameType>
                        <m:name>UtilitiesDeviceID</m:name>
                        <m:NameTypeAuthority>
                            <m:name>EMO_327</m:name>
                        </m:NameTypeAuthority>
                    </m:NameType>
                </m:Names>
            </m:Meter>
        </m:GetMeterConfig>
    </msg:Request>
</msg:RequestMessage>
```

Figure 46: Equipment Number in Message Body

Run the request and check if it was successful.

Add an additional name as identifier for the object “MeterConfig”

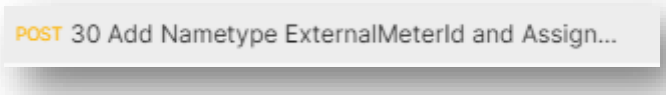
You want to use your own identifers and use alternative names.

Alternative names also require a so-called NameTypeAuthority.
Use Case: Add external ID as unique identifier.

A. Modify a meter with UUID (for example 249a01f4-9ce8-4e42-bee8-ab4422502530)

B. Add name ExternalMeterId as identifier with NameTypeAuthority Test_123.
   You assign ABCD-EXT123-Example as (alternative) Name

C. Read the meter data by the new name ExternalMeterID



Name and NameType are assigned to an existing meter. Meter is identified via MeterUUID.

```
<?xml version="1.0" encoding="UTF-8"?>
<msg:RequestMessage xmlns:msg="http://iec.ch/TC57/2011/schema/message" xmlns:m="http://iec.ch/TC57/CIM-c4e#">
    <msg:Header>
        <msg:Verb>create</msg:Verb>
        <msg:Noun>ObjectNamesConfig</msg:Noun>
    </msg:Header>
    <msg:Payload>
        <m:ObjectNamesConfig>
            <m:ConfigurationEvent>
                <m:effectiveDateTime>{{now}}</m:effectiveDateTime>
            </m:ConfigurationEvent>
            <m:Meter>
                <m:mRID>249a01f4-9ce8-4e42-bee8-ab4422502530</m:mRID>
            </m:Meter>
            <m:Name>
                <m:name>ABCD-EXT123-Example</m:name>
                <m:NameType>
                    <m:name>ExternalMeterId</m:name>
                    <m:NameTypeAuthority>
                        <m:name>Test_123</m:name>
                    </m:NameTypeAuthority>
                </m:NameType>
            </m:Name>
        </m:ObjectNamesConfig>
    </msg:Payload>
</msg:RequestMessage>
```

Figure 47: Name and NameType in Message Body

Run the request and check if it was successful.

Read Meter with Alternative Name

You want to read a meter by an alternative name.

In the earlier steps, you created a meter and assigned the alternative name with NameType “ExternalMeterId”.

(Alternative Name/Identifier)	ABCD-EXT123-Example
NameType	ExternalMeterId
NameTypeAuthority	Test_123

POST 32 GET MeterConfig by Alternative NameType

```xml
<?xml version="1.0" encoding="UTF-8"?>
<msg:RequestMessage xmlns:msg="http://iec.ch/TC57/2011/schema/message" xmlns:m="http://iec.ch/TC57/CIM-c4e#">
    <msg:Header>
        <msg:Verb>get</msg:Verb>
        <msg:Noun>MeterConfig</msg:Noun>
    </msg:Header>
    <msg:Request>
        <m:GetMeterConfig>
            <m:Meter>
                <m:Names>
                    <m:name>ABCD-EXT123-Example</m:name>
                    <m:NameType>
                        <m:name>ExternalMeterId</m:name>
                        <m:NameTypeAuthority>
                            <m:name>Test_123</m:name>
                        </m:NameTypeAuthority>
                    </m:NameType>
                </m:Names>
            </m:Meter>
        </m:GetMeterConfig>
    </msg:Request>
</msg:RequestMessage>
```

Figure 48: Alternative Name in Message Body

Run the request and check if it was successful.

Create Meter Readings for the MeterConfig

You want to create 96 measurement values for the meter.

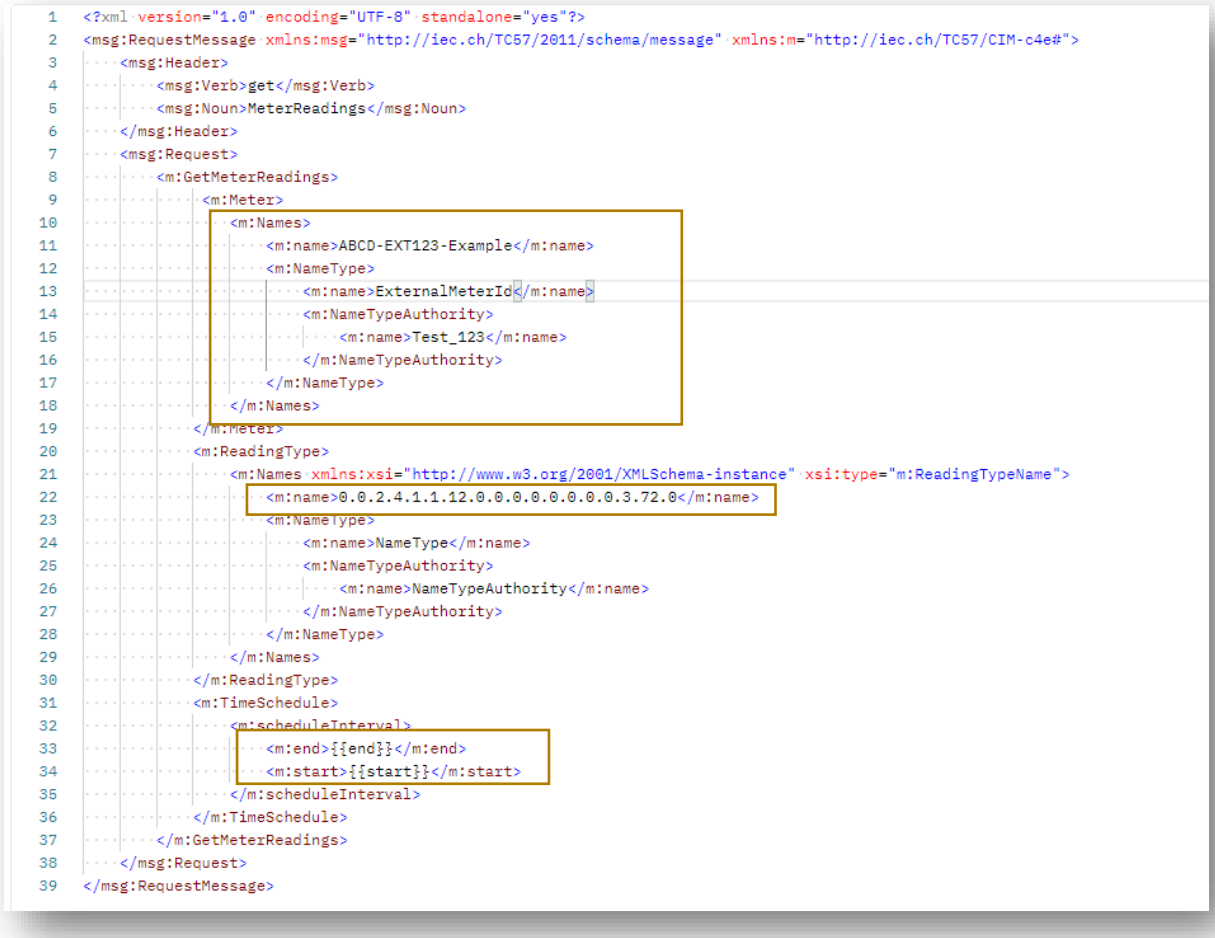
POST 40 CREATE 96 valid readings for meter

Run the request and check if it was successful.

Get Readings by Additional NameType and ReadingType

POST 42 GET Readings - Validated by Name and Re...

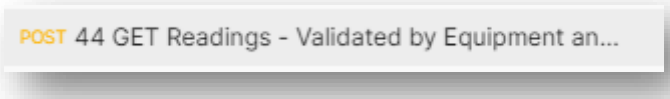
You want to read measurement values by NameType and ReadingType.



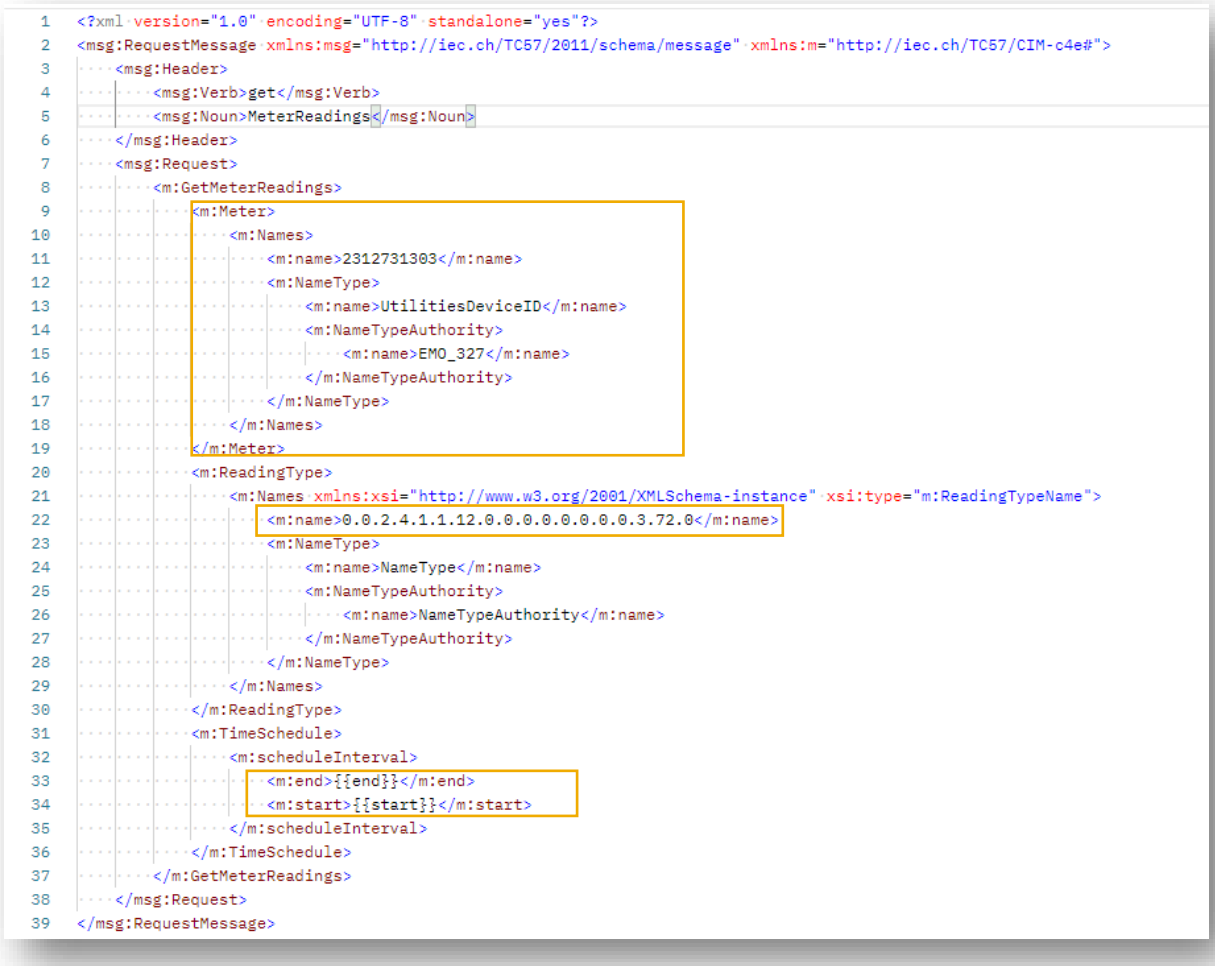
```xml
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<msg:RequestMessage xmlns:msg="http://iec.ch/TC57/2011/schema/message" xmlns:m="http://iec.ch/TC57/CIM-c4e#">
    <msg:Header>
        <msg:Verb>get</msg:Verb>
        <msg:Noun>MeterReadings</msg:Noun>
    </msg:Header>
    <msg:Request>
        <m:GetMeterReadings>
            <m:Meter>
                <m:Names>
                    <m:name>ABCD-EXT123-Example</m:name>
                    <m:NameType>
                        <m:name>ExternalMeterId</m:name>
                        <m:NameTypeAuthority>
                            <m:name>Test_123</m:name>
                        </m:NameTypeAuthority>
                    </m:NameType>
                </m:Names>
            </m:Meter>
            <m:ReadingType>
                <m:Names xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="m:ReadingTypeName">
                    <m:name>0.0.2.4.1.1.12.0.0.0.0.0.0.0.0.3.72.0</m:name>
                    <m:NameType>
                        <m:name>NameType</m:name>
                        <m:NameTypeAuthority>
                            <m:name>NameTypeAuthority</m:name>
                        </m:NameTypeAuthority>
                    </m:NameType>
                </m:Names>
            </m:ReadingType>
            <m:TimeSchedule>
                <m:scheduleInterval>
                    <m:end>{{end}}</m:end>
                    <m:start>{{start}}</m:start>
                </m:scheduleInterval>
            </m:TimeSchedule>
        </m:GetMeterReadings>
    </msg:Request>
</msg:RequestMessage>
```

Figure 49: NameType and ReadingType in Message Body

Get Readings by Equipment and ReadingType



You want to read measurement values by Equipment and ReadingType.



```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<msg:RequestMessage xmlns:msg="http://iec.ch/TC57/2011/schema/message" xmlns:m="http://iec.ch/TC57/CIM-c4e#">
    <msg:Header>
        <msg:Verb>get</msg:Verb>
        <msg:Noun>MeterReadings</msg:Noun>
    </msg:Header>
    <msg:Request>
        <m:GetMeterReadings>
            <m:Meter>
                <m:Names>
                    <m:name>2312731303</m:name>
                    <m:NameType>
                        <m:name>UtilitiesDeviceID</m:name>
                        <m:NameTypeAuthority>
                            <m:name>EMO_327</m:name>
                        </m:NameTypeAuthority>
                    </m:NameType>
                </m:Names>
            </m:Meter>
            <m:ReadingType>
                <m:Names xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:type="m:ReadingTypeName">
                    <m:name>0.0.2.4.1.1.12.0.0.0.0.0.0.0.3.72.0</m:name>
                    <m:NameType>
                        <m:name>NameType</m:name>
                        <m:NameTypeAuthority>
                            <m:name>NameTypeAuthority</m:name>
                        </m:NameTypeAuthority>
                    </m:NameType>
                </m:Names>
            </m:ReadingType>
            <m:TimeSchedule>
                <m:scheduleInterval>
                    <m:end>{{end}}</m:end>
                    <m:start>{{start}}</m:start>
                </m:scheduleInterval>
            </m:TimeSchedule>
        </m:GetMeterReadings>
    </msg:Request>
</msg:RequestMessage>
```

Figure 50: Equipment and ReadingType in Message Body

Meter with Multiple Channels and Registers

You want to create a meter with two channels and registers.
One register is to store consumption time series data; the other register is to store meter reading time series data.

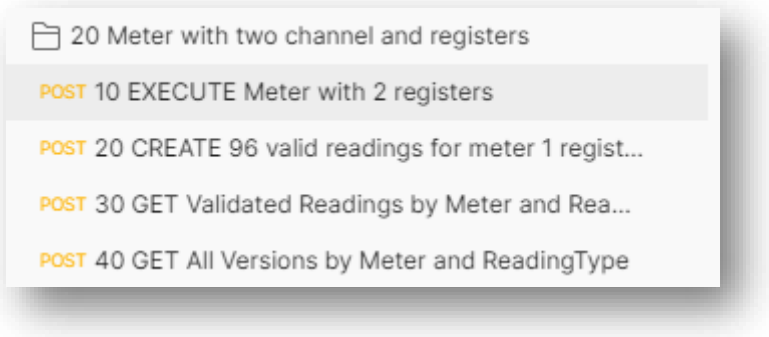


Figure 51: Folder 20

Run the request – you will get a Meter with the other relevant items

- UsagePointLocationConfig
- UsagePointConfig
- MeterConfig
- MasterDataLinkageConfig

and the possibility to process data on two registers.

Review the Register Data Section in the app. You should see that there are 2 registers: one to store consumption time series data, the other to store meter reading time series data:

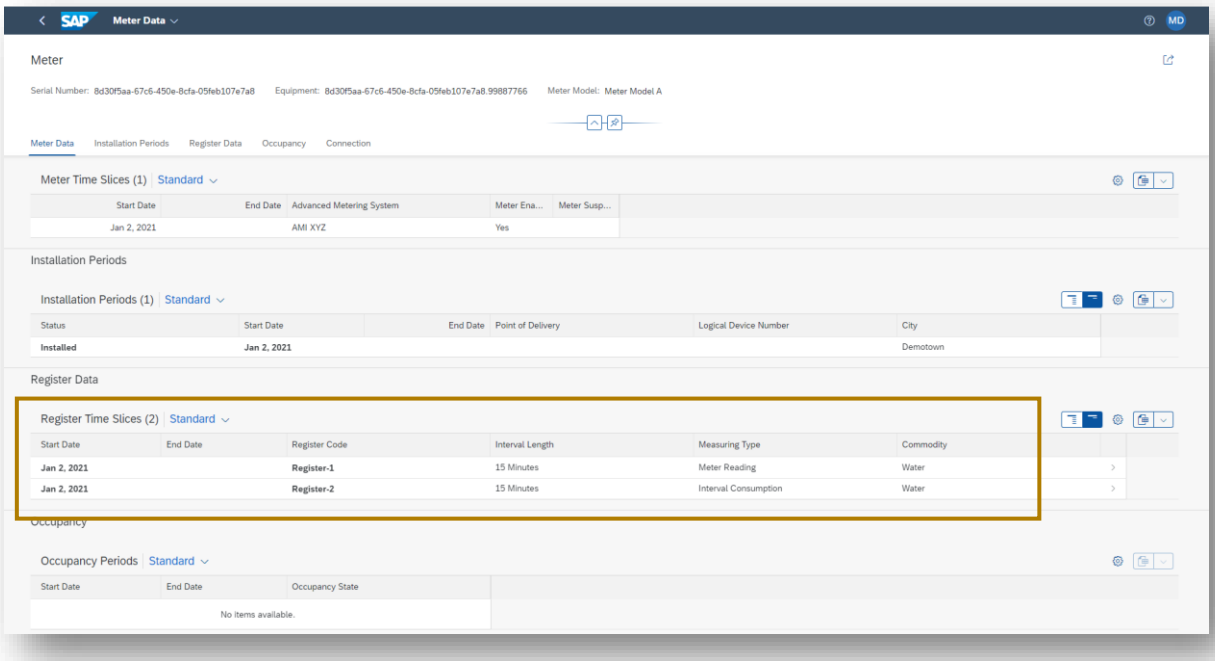


Figure 52: View Meter Data App

Ingest Measurement Data & Check Versioning

SAP Cloud for Energy supports versioning for measurement values. The system does not overwrite existing data but writes a new version with each new ingestion. You can query the measurement data versions. Try it out!

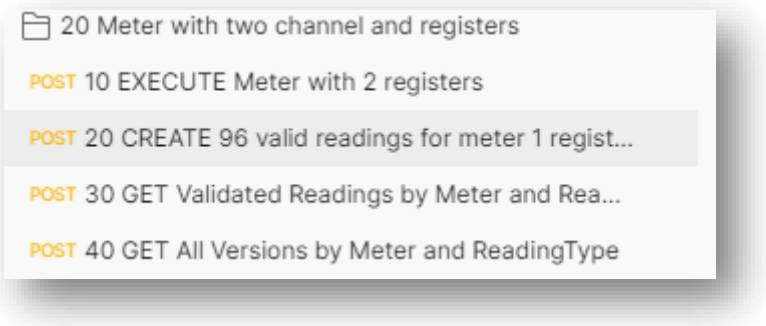


Figure 53: Folder 20-20

Ingest Measurement Values

Ingest 96 readings for the register with reading type: 0.0.2.4.0.9.58.0.0.0.0.0.0.0.0.119.0

Check the result.

Run the POST 30 GET Validated Readings by Meter and Rea... to read the data you just created.

Result:

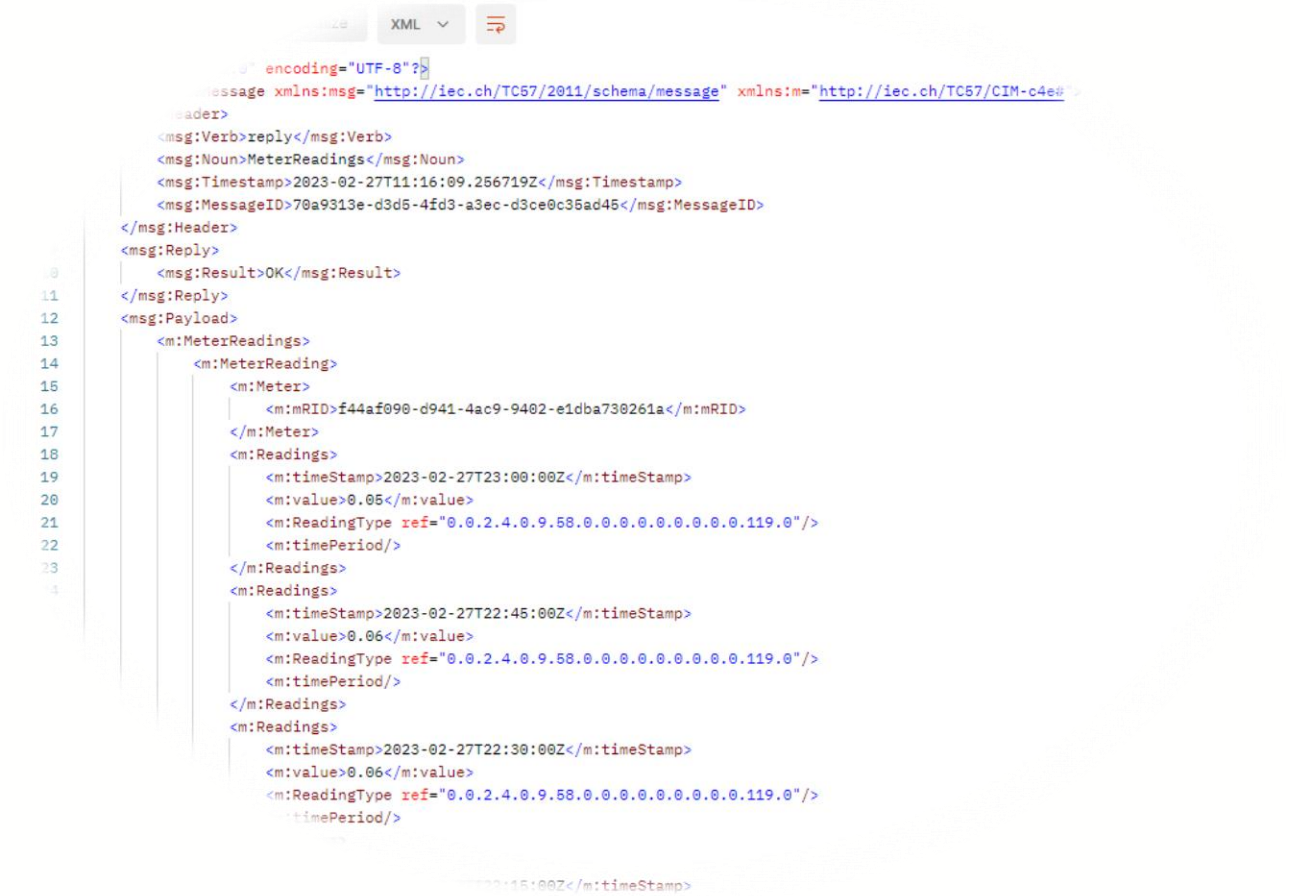


Figure 54: Result Fragment

Change one of the values in the request “20 CREATE 96 valid readings…”.

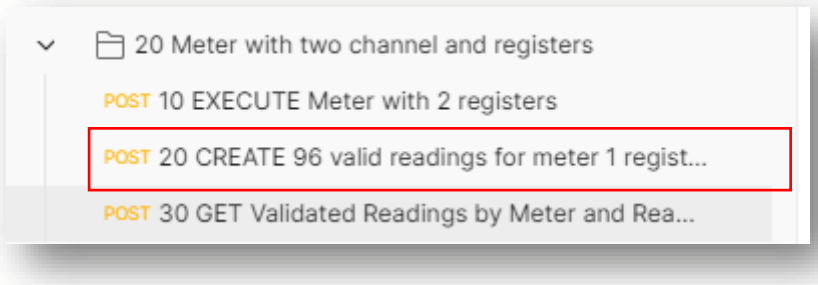


Figure 55: Folder 20

For example, change the value of the last reading from 0.05 to 0.33.

Initial value:

```
668                    </m:IntervalReadings>
669                    <m:IntervalReadings>
670                        <m:timeStamp>{{date94}}</m:timeStamp>
671                        <m:value>0.06</m:value>
672                        <m:ReadingQualities>
673                            <m:ReadingQualityType ref="1.0.0"/>
674                        </m:ReadingQualities>
675                    </m:IntervalReadings>
676                    <m:IntervalReadings>
677                        <m:timeStamp>{{date95}}</m:timeStamp>
678                        <m:value>0.05</m:value>
679                        <m:ReadingQualities>
680                            <m:ReadingQualityType ref="1.0.0"/>
681                        </m:ReadingQualities>
682                    </m:IntervalReadings>
683                    <m:ReadingType ref="0.0.2.4.0.9.58.0.0.0.0.0.0.0.0.0.119.0"/>
684                </m:IntervalBlocks>
685                <m:Meter>
686                    <m:mRID>{{MeterUUID}}</m:mRID>
687                </m:Meter>
688            </m:MeterReading>
689        </m:MeterReadings>
690    </msg:Payload>
691 </msg:RequestMessage>
```

Figure 56: Initial Value

New value:

```
674                        </m:ReadingQualities>
675                    </m:IntervalReadings>
676                    <m:IntervalReadings>
677                        <m:timeStamp>{{date95}}</m:timeStamp>
678                        <m:value>0.33</m:value>
679                        <m:ReadingQualities>
680                            <m:ReadingQualityType ref="1.0.0"/>
681                        </m:ReadingQualities>
682                    </m:IntervalReadings>
683                    <m:ReadingType ref="0.0.2.4.0.9.58.0.0.0.0.0.0.0.0.0.119.0
684                </m:IntervalBlocks>
685                <m:Meter>
686                    <m:mRID>{{MeterUUID}}</m:mRID>
687                </m:Meter>
688            </m:MeterReading>
```

Figure 57: Changed Value

Re-Run this request to update from 0.05 to 0.33

Use `40 GET All Versions by Meter and ReadingType` to query measurement data including all versions:

```
<msg:Payload>
    <m:MeterReadings>
        <m:MeterReading>
            <m:Meter>
                <m:mRID>f44af090-d941-4ac9-9402-e1dba730261a</m:mRID>
            </m:Meter>
            <m:RevisionBlocks>
                <m:timeStamp>2023-02-27T23:00:00Z</m:timeStamp>                    (1)
                <m:ReadingType ref="0.0.2.4.0.9.58.0.0.0.0.0.0.0.0.119.0"/>
                <m:RevisionReadings>
                    <m:created>2023-02-27T11:22:11.457Z</m:created>
                    <m:value>0.33</m:value>                                        (2)
                    <m:measurementDataType>VALIDATED</m:measurementDataType>
                </m:RevisionReadings>
                <m:RevisionReadings>
                    <m:created>2023-02-27T11:22:10.457Z</m:created>
                    <m:value>0.33</m:value>
                    <m:ReadingQualities>                                           (3)
                        <m:ReadingQualityType ref="1.0.0"/>
                    </m:ReadingQualities>
                    <m:measurementDataType>RAW</m:measurementDataType>
                </m:RevisionReadings>
                <m:RevisionReadings>
                    <m:created>2023-02-27T11:05:26.537Z</m:created>
                    <m:value>0.05</m:value>                                        (4)
                    <m:measurementDataType>VALIDATED</m:measurementDataType>
                </m:RevisionReadings>
                <m:RevisionReadings>
                    <m:created>2023-02-27T11:05:25.537Z</m:created>
                    <m:value>0.05</m:value>
                    <m:ReadingQualities>                                           (5)
                        <m:ReadingQualityType ref="1.0.0"/>
                    </m:ReadingQualities>
                    <m:measurementDataType>RAW</m:measurementDataType>
                </m:RevisionReadings>
            </m:RevisionBlocks>
```

Figure 58: All Versions

1) Look for the value with time stamp 2023-02-27T23:00:00Z
2) The value is 0.33 after the validation (VALIDATED)
3) The value 0.33 was received by the API (RAW - before passing the validation)
4) The value is 0.05 after the validation (VALIDATED)
5) The value 0.05 was received by the API (RAW - before passing the validation)

Explanation

The value with time stamp 2023-02-27T23:00:00Z exists twice: in a first version (0.05) and an additional version (0.33).

Change Objects

You want to change (update) an object.

These examples will show you how to create a MeterConfig with a Serial number. Next, you will read the meter data to check if everything went well. You will then change the Serial number and read the meter again by the new and now valid serial number.

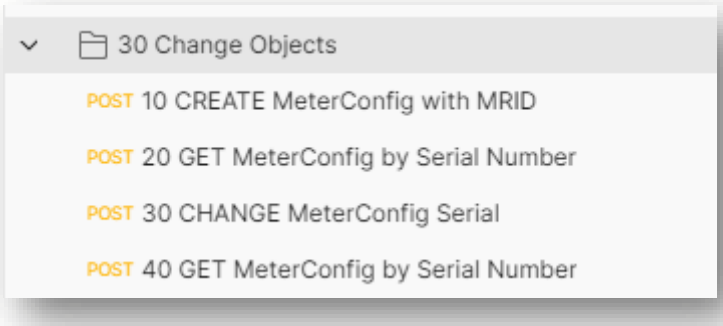
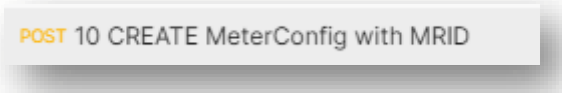


Figure 59: Folder 30

Create Meter with Serial Number.

The example serial number is `<m:serialNumber>Serial-99-99-99</m:serialNumber>`

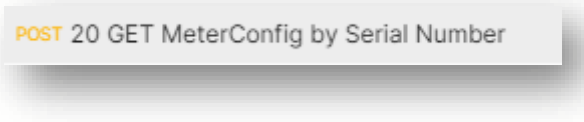
Run the request and check if it was successful.




```
4       <msg:Verb>create</msg:Verb>
5       <msg:Noun>MeterConfig</msg:Noun>
6   </msg:Header>
7   <msg:Payload>
8       <m:MeterConfig>
9           <m:Meter>
10              <m:mRID>{{MeterUUID}}</m:mRID>
11              <m:amrSystem>METER_AMR_SYSTEM</m:amrSystem>
12              <m:isVirtual>false</m:isVirtual>
13              <m:serialNumber>Serial-99-99-99</m:serialNumber>
14              <m:timeZone>Europe/Berlin</m:timeZone>
```

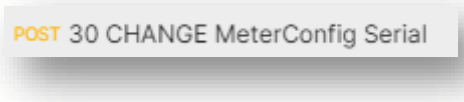
Figure 60: SerialNumber in Message Body

Read the MeterConfig with the serial number that you used before, e.g.,
`<m:serialNumber>Serial-99-99-99</m:serialNumber>`



Result: The MeterConfig is returned in the response.

Change the Serial Number of an Existing MeterConfig.




```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<msg:RequestMessage xmlns:msg="http://iec.ch/TC57/2011/schema/message" xmlns:m="http://iec.ch/TC57/CIM-c4e#">
    <msg:Header>
        <msg:Verb>change</msg:Verb>
        <msg:Noun>MeterConfig</msg:Noun>
    </msg:Header>
    <msg:Payload>
        <m:MeterConfig>
            <m:Meter>
                <m:mRID>{{MeterUUID}}</m:mRID>
                <m:serialNumber>NEW SERIAL-1111</m:serialNumber>
                <m:timeZone>Europe/Berlin</m:timeZone>
                <m:ConfigurationEvents>
                    <m:effectiveDateTime>2023-02-28T11:54:15.465Z</m:effectiveDateTime>
                </m:ConfigurationEvents>
            </m:Meter>
        </m:MeterConfig>
    </msg:Payload>
</msg:RequestMessage>
```

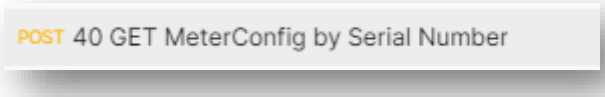
1 2 3

Figure 61: SerialNumber in Message Body

Explanation:
Change – MeterConfig
Provide the new serial number e.g., NEW SERIAL-1111
Add the `effectiveDateTime` which tells when the new serial number will become valid, e.g., Feb.28, 2023 in the required format.

Run the request and check if it was successful.

Query Meter by New Serial Number



Open the request and double-check if the query parameter is the serial number you used in the Change payload.

```
<?xml version="1.0" encoding="UTF-8" standalone="yes"?>
<msg:RequestMessage xmlns:msg="http://iec.ch/TC57/2011/schema/message" xmlns:m="http://iec.ch/TC57/CIM-c4e#">
    <msg:Header>
        <msg:Verb>get</msg:Verb>
        <msg:Noun>MeterConfig</msg:Noun>
    </msg:Header>
    <msg:Request>
        <m:GetMeterConfig>
            <m:Meter>
                <m:serialNumber>NEW SERIAL-1111</m:serialNumber>
            </m:Meter>
        </m:GetMeterConfig>
    </msg:Request>
</msg:RequestMessage>
```

Figure 62: Changed SerialNumber

Run the request and check if it was successful.

The MeterId with Serial Number is returned in the response.

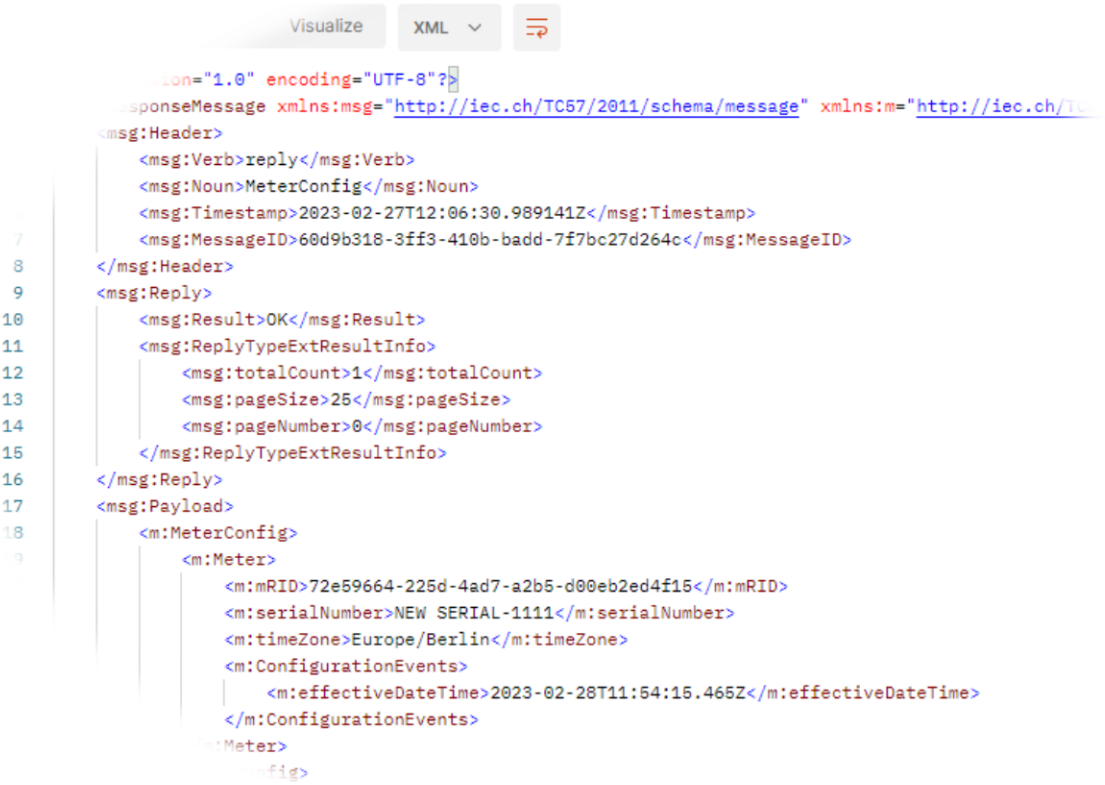


Figure 63: Result Fragment

Various Payloads

The sample collection has a folder called “99 Various Payloads” with subfolders for several topics.

Feel free to review the requests and understand further details related to the *Energy Data Services* API.

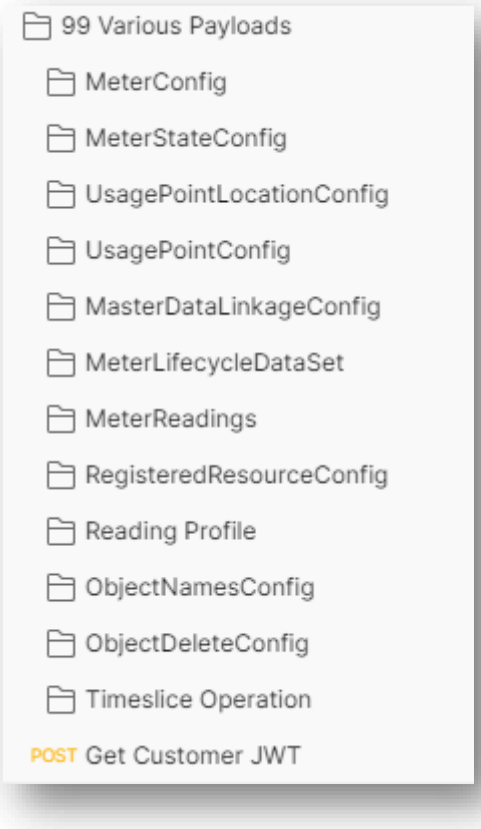


Figure 64: Folder 99

Troubleshooting

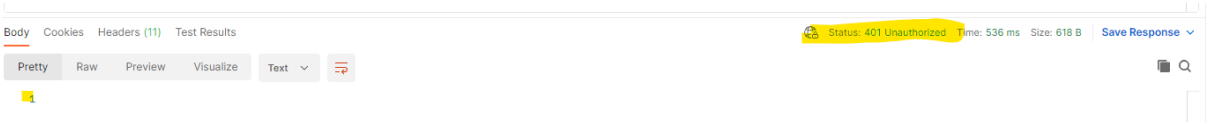
Error: Invalid protocol

Error: Invalid protocol: {{protocol}}:

```
POST {{protocol}}://{{domain}} {{cim_endpoint}}/
```

Resolution: Check if you have assigned an Environment where these variables such as protocol, domain, and cim_endpoint are configured.

Status 401 Unauthorized.



Resolution: Make sure you have a valid Customer JWT.

Try again.

References

IEC 61968-9 (Edition 2.0 2013-10):

Application integration at electric utilities - System interfaces for distribution management - Part 9: Interfaces for meter reading and control

IEC 61968-100 (Edition 1.0 2013-07):

Application integration at electric utilities - System interfaces for distribution management - Part 100: Implementation profiles

IEC 61968-900:

Application integration at electric utilities. System interfaces for distribution management - Part 900: Guidance for implementation of IEC 61968-9

Thank you for investing your time to read this guide. We’re pretty sure it will pay off.
www.sap.com.