PowerMeasurement HTTP/TLS/JSON

Interface Design Description

Abstract
This document describes the PowerMeasurement service interface using HTTP with TLS security and JSON payload encoding. The interface provides secure access to electrical power measurement data from the PowerSensor subsystem within the AI ore handling control and optimization system.

Contents

1 Overview

This document describes the **PowerMeasurement** service interface, which provides access to electrical power measurement data from the **PowerSensor_SysD** subsystem.

This IDD implements the interface defined in the *PowerMeasurement – Service Description* document and enables higher-level control and optimization systems to retrieve secure and structured power readings from industrial equipment such as conveyor motors.

It's implemented using protocol, encoding and security as stated in Table 1.

Profile Type	Type	Version / Example
Transfer protocol	HTTP	1.1
Data encryption	TLS	1.3
Encoding	JSON	RFC 8259 [1]
Compression	N/A	–
Semantics	SenML	RFC 8428 [**?**]
Ontology	N/A	–
Security	TOKEN	JWT (Arrowhead Token)

Table 1: Communication and semantics details used for the PowerMeasurement service interface

Explanation: This communication profile ensures secure, interoperable, and machine-readable data exchange between subsystems. Communication occurs over HTTP 1.1 with end-to-end encryption using TLS 1.3. Data is encoded in JSON following the SenML format for consistent measurement representation. Token-based authentication (Arrowhead Token) is employed, where each system obtains an authorization token from a trusted service before accessing the PowerMeasurement interface.

The rest of this document describes how to realize the PowerMeasurement service interface in detail. Both in terms of its operations (Section 2) and its information model (Section 3).

Figure 1: (Optional) SysML model of the PowerMeasurement interface, its operations, datamodels and implementation.

2 Service Operations

The interfaces of the PowerMeasurement service, its operations, data models and implementation are provided below. A SysML service overview may be included in Figure 1.

All operations in this section respond with the status code `200 OK` if called successfully. The error codes are: `400 Bad Request` if request is malformed, `401 Unauthorized` if an invalid or expired token is provided, `500 Internal Server Error` if communication fails.

2.1 GET /powermonitor/measurement

Operation: GetPowerMeasurement
Output: PowerMeasurement

Called to retrieve the current electrical power measurement data from the connected motor or feeder sensor, as exemplified in Listing 1.

```
GET /powermonitor/measurement HTTP/1.1
Host: 192.168.0.110
Accept: application/json
Authorization: Bearer <ArrowheadToken>
```

Listing 1: A GetPowerMeasurement invocation.

```
{
  "timestamp": "2025-10-15 12:00:00",
  "voltage": 380.0,
  "current": 5.2,
  "power": 1976.0,
  "unit": "W",
  "sensorId": "TP-2"
}
```

Listing 2: A GetPowerMeasurement response.

3 Data Models

Here, all data objects that can be part of the service calls associated with this service are listed in alphabetic order. Note that each subsection, which describes one type of object, begins with the *struct* keyword, which is meant to denote a JSON Object that must contain certain fields, or names, with values conforming to explicitly named types. As a complement to the primary types defined in this section, there is also a list of secondary types in Section 3.2, which are used to represent things like identifiers and timestamps.

3.1 struct PowerMeasurement

This structure is used as the response payload for the GetPowerMeasurement operation.

Object Field	Value Type	Description
”timestamp”	DateTime	Time when the measurement was taken.
”voltage”	Float	Measured voltage in volts.
”current”	Float	Measured current in amperes.
”power”	Float	Calculated electrical power in watts.
”unit”	String	Measurement unit, e.g. `"W"`.
”sensorId”	String	Identifier of the PowerSensor providing data.

3.2 Primitives

As all messages are encoded using the JSON format , the following primitive constructs, part of that standard, become available.

JSON Type	Description
Value	Any out of Object, Array, String, Number, Boolean or Null.
Object <A>	An unordered collection of [String: Value] pairs, where each Value conforms to type A.
Array <A>	An ordered collection of Value elements, where each element conforms to type A.
String	An arbitrary UTF-8 string.
Number	Any IEEE 754 binary64 floating point number [2], except for *+Inf*, *-Inf* and *NaN*.
Boolean	One out of `true` or `false`.
Null	Must be `null`.

With these primitives now available, we proceed to define the types without a direct equivalent among the JSON types. Concretely, we define primitives either as *aliases* or *structs*.

3.2.1 alias DateTime = String

Pinpoints a moment in time in the format of ”YYYY-MM-DD HH:mm:ss”, where ”YYYY” denotes year (4 digits), ”MM” denotes month starting from 01, ”DD” denotes day starting from 01, ”HH” denotes hour in the 24-hour format (00-23), ”MM” denotes minute (00-59), ”SS” denotes second (00-59). ” ” is used as separator between the date and the time. An example of a valid date/time string is ”2025-10-15 12:00:00”.

3.2.2 alias Float = Number

A Number intended to be used for real-valued measurement quantities such as voltage, current and power.

4 References

[1] T. Bray, “The JavaScript Object Notation (JSON) Data Interchange Format,” RFC 8259, Dec. 2017. [Online]. Available: https://rfc-editor.org/rfc/rfc8259.txt

[2] M. Cowlishaw, “IEEE Standard for Floating-Point Arithmetic,” *IEEE Std 754-2019 (Revision of IEEE 754-2008)*, July 2019. [Online]. Available: https://doi.org/10.1109/IEEESTD.2019.8766229

5 Revision History

5.1 Amendments

No.	Date	Version	Subject of Amendments	Author
1	2025-10-15	1.0	Initial version of PowerMeasurement_IDD (TOKEN-based security)	ricbli-7

5.2 Quality Assurance

No.	Date	Version	Approved by
1	2025-10-15	1.0	–