Interface Design Description (IDD) wm-data service

**Abstract**

This document describes the interfaces, communication profiles and other desirable information about using the Windmill systems.

1. Interface Design Description Overview 3
2. Service Interfaces 3
   1. Interface 1 4
      1. Data Model 4
      2. Status and Error handling 5
      3. Interaction with consumers 5
   2. Interface 2 5
      1. Information Model 6
      2. Status and Error handling 6
3. Security 6
   1. Certificate 6
   2. Payload protection 7
   3. Audit of service exchange 7
4. References 7
5. Revision history 8
   1. Amendments 8
   2. Quality Assurance 8

## Interface Design Description Overview

The interface’s black box description can be found in Service Description (SD) document for this service. See Table 1.

**Table 1 Pointers to SD documents**

|  |  |
| --- | --- |
| **Realised Service Description** | **Location** |
| wm-data | documentation/SD\_windmill v2.0.docx |

Communication method

* Protocol
  + HTTPS
* Encoding
  + JSON
* Compression
  + None
* Base path: /wm-data-json
  + Example path: https://<IP>:<Port>/wm-data-json

Language and libraries

The provider and consumer are built in Java using the Arrowhead [client-skeleton-java-spring](https://github.com/arrowhead-f/client-skeleton-java-spring) library. The client skeleton utilizes [Maven](https://maven.apache.org/) and [Spring framework](https://spring.io/) for building the provider and consumer.

In addition to that, the library [OpenCSV](https://mvnrepository.com/artifact/com.opencsv/opencsv) is used to read and parse the CSV file. For sending the data to the DataManager as SenML encoded JSON strings, team-ethernet’s [SenML\_API](https://github.com/team-ethernet/SenML_API) is used.

## Service Interfaces

The Windmill system only provides one service, wm-data. The wm-data service provides sensor data from a connected windmill. The provider utilizes a first in first out queue for the data sent by the windmill.

## Service 1: wm-data

* The data model is application/json
* No ontologies are in use
* No schema is currently defined
* Payload is encrypted by TLS automatically

**Table 4 INTERFACE description**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Interface name** | **Relative URL path** | **Method** | **Input** | **Output** |
| wm-data | /wm-data-json | GET | param-request-next=next | One measurement of sensor data from the windmill |

## Data Model

The data is sent as a JSON object, with two Double values timeStamp, speed and finally an array of acceleration values.

Here is an example output after a successful GET request to the service:

{”timeStamp”: <time>, ”speed” : <speed>, ”accelerometer” : <value1> ,…. , ”accelerometer” : <value16384> }

**Table 5 Data Object description**

|  |  |  |  |
| --- | --- | --- | --- |
| **Object Field** | **Description** | **Format/limitations** | **Required** |
| timeStamp | Time of measurement | Double | Yes |
| speed | Speed at measurement | Double | Yes |
| accelerometer | Timeseries of Gs measured | List<Double> | Yes |

## Status and Error handling

|  |  |  |
| --- | --- | --- |
| **Code** | **Meaning** | **Comment** |
| 200 | Successful request | Success |
| 401 | Unauthorized | Access Denied |
| 400 | Bad request | Wrong parameters |
| 500 | Internal server error | Error getting sensor data etc. |

## Interaction with consumers

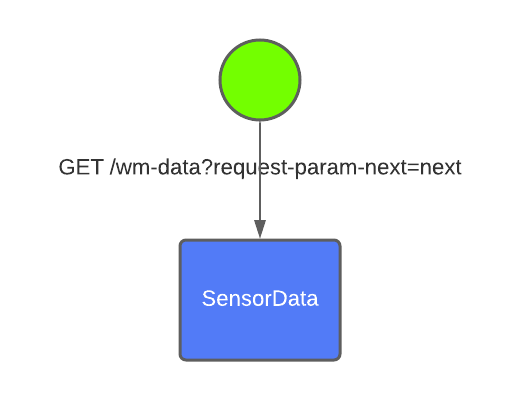


Figure 1: Consumer interaction

## Security

The service uses HTTPS with token-based authorization provided by the Authorization core system.

All messages sent by the service is also stored in the DataManager’s historian service, this provides a good history of exchanges made.

## Certificate

The service uses client certificates.

## Payload protection

TLS payload encryption is used as it is a part of the [client-skeleton-java-spring](file:///C:\Users\rjons\Downloads\client-skeleton-java-spring) libraries.

## Audit of service exchange

The service uses the DataManager to store all sent sensor data. This is done before sending the data to the consumer. When a request is made for a data point, the provider first sends the data to the DataManager before sending it to the consumer that requested the data.

## References

[OpenJDK](https://openjdk.java.net/projects/jdk/14/)

[client-skeleton-java-spring](https://github.com/arrowhead-f/client-skeleton-java-spring)

[OpenCSV Maven repository](https://mvnrepository.com/artifact/com.opencsv/opencsv)

[SenML\_API Library Github](https://github.com/team-ethernet/SenML_API)

## Revision history

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| No. | Date | Version | Subject of Amendments | Author |
| 1 | 2021-01-25 | 1.0 | First draft | Robin Jonsson |
| 2 | 2021-02-04 | 2.0 | Major update to content | Robin Jonsson |

## Quality Assurance

|  |  |  |  |
| --- | --- | --- | --- |
| No. | Date | Version | Approved by |
| 1 |  |  |  |
| 2 |  |  |  |