DataManager

System Design Description (SysDD)

Abstract
This document describes a system useful for managing (sensor) data from Arrowhead-compliant systems. In essence, the DataManager system provides features for producers and consumers of data to store and retrieve sensor data.

ARTEMIS Innovation Pilot Project: Arrowhead
THEME [SP1-JTI-ARTEMIS-2012-AIPP4 SP1-JTI-ARTEMIS-2012-AIPP6]
[Production and Energy System Automation Intelligent-Built environment and urban infrastructure for sustainable and friendly cities]

Contents

1 Overview

This document describes the DataManager (DM) of the Eclipse Arrowhead [1] system, which provides services for short- and long-term data storage, retrieval and filtering. Common use-cases are;

- Short term caching for systems running on battery-powered, sleepy, devices
- Mass storage of sensor data in a backend database
- Providing data for backup purposes and dash boards
- and others...

For example, the DataManager's Proxy service can be used by any sensor data producing system to cache exactly one message. Since the DataManager must be deployed on a sufficiently powerful server, it can offload processing loads from embedded devices. It also enables sleepy devices that spend most of their time offline in order to conserve energy. This is aechieved by allowing a sleepy device to make up, measure and update its virtual endpoint in the Proxy. The device can then go back to sleep. Consumers however, can always get the latest measurement by reading from the always-on DataManager. Another example is the Historian which provides similar features. The main difference is that the Historian uses a database as its backend, and can thus save all data from producers, not just the latest value. This allows the Historian to provide trends, backup of data, and filtering of data by time and value. Te DataManager system thus acts as a cache and database front-end in an Arrowhead local cloud, as exemplified in Figure 1.

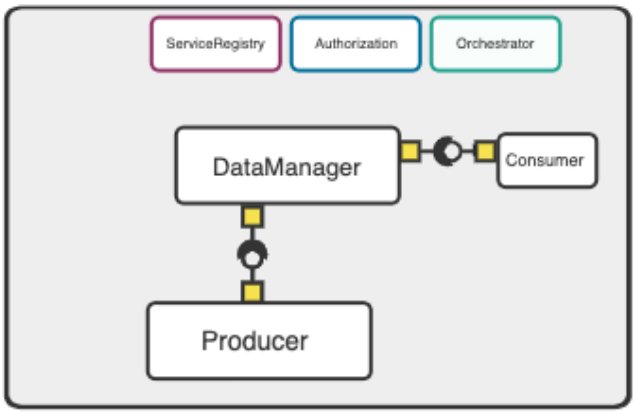

Figure 1: An example of an Arrowhead local cloud which contains the Core systems, one data producer and one data consumer.

The rest of this document is organized as follows. In the remainder of this section we comment on the status of this document. In Section 2, we outline major delimitations of the system, which is a work-in-progress. Section 3 presents how the DataManager is used in a local cloud. Finally, In Section 4, we describe which services the DataManager system consumes and produces.

1.1 Status of this Document

This document presents the current state of the DataManager. However, since Eclipse Arrowhead is an actve open source project, changes will most like happen in the future. Features will be added, modified or in other ways altered. This document will thus be updated when needed.

This document, an all other such part of the same Eclipse Arrowhead Core proposal, are still to be considered early drafts and might to have to undergo several significant revisions before becoming sufficient for most kinds of industrial deployments.

If the reader has ay comments or suggestions regarding the design or implementations, please contact Jens Eliasson <jens.eliasson@thingwave.eu>, who the maintainer of the DataManager system.

2 Important Delimitations

The primary purpose of the DataManager system is to provide services and features for management of sensor data. Currently, the following features are out of scope;

- No storage of files, or other types of non-structured data
- Only SenML in JSON is considered
- There are no features for advanced analytics
- There are no features for alarm generation, etc

3 System Role

As stated in Section 1, the DataManager system performs two main roles. Firstly, it manages caching of data for sleepy devices, and secondly, it provides an Arrowhead compliant interface for database operations such as store, fetch an filter.

One Proxy use case can be the following:

1. An Arrowhead Local cloud wants to ensure that the data from a sensor system, deployed on a battery-powered device, is available for other consumers when when the device is offline.
2. An orchestration rule is created and sent to the sensor system,
3. The sensor system starts to perform the most common approach for energy conservation, duty-cycling. The sensor system starts to periodically wake up from sleep, activate its sensor(s) and get sensor data. The data is pushed to the Proxy service for caching. After a successful update, the device returns to sleep and the whole cycle restarts.
4. A consumer that needs the sensor data will also get an orchestration rule to fetch data from the DataManager's Proxy service instead of getting it from the device itself.

One Historian use case can be the following:

1. One data producing system must store large amounts of data over time. At the same time, multiple consumers must have access to that data.
2. An orchestration rule is created that will ensure that the data producers stores all of its generated sensor data at the Historian service.
3. Orchestration rules are also dispatched to all the consumer systems to instruct them to fetch both real-time and historic data from the Historian.

3.1 Data models

The table below shows the different encodings that can be used with SenML [2]. Today, only plain text JSON is used. Future versions of the DataManagar can be extended to support XML and CBOR [3], and also use other data models than SenML.

Encoding	Description
JSON	Standard JSON. Pros; easy to work with. Cons: relatively verbose.
XML	Standard XML. Pros; Lots of tools, Cons: very verbose.
CBOR	Binary format. Pros: highly efficient, Cons: hard to work with

Other data models that would be good to support are; SNON [4], GeoJSON [5], SensorML [6].

4 Services

The DataManager systems produces and consumes the following services, as described in Figure 2.

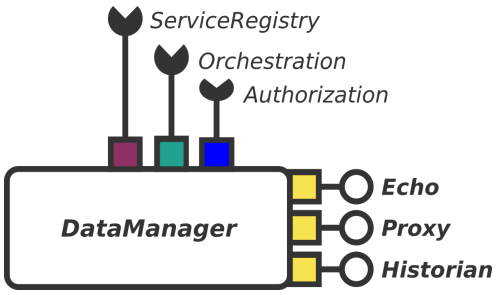

Figure 2: The Arrowhead DataManager system.

More details regarding the consumed and produced services are given in the following subsection.

4.1 Consumed Services

This section presents an overview of consumed services.

4.1.1 Service Registry

This service is consumed to make sure that the DataManager system becomes accessible to other systems.

4.1.2 Orchestration

This service is currently not used, but will be important when the DataManager starts to consume other systems' services in the future.

4.1.3 Authorization

This service provides information about access control to the DataManager's services.

4.2 Produced Services

This section presents an overview of provided services.

4.2.1 Echo

The Echo service is a simple service that can be used to determine if a system is available or not.

4.2.2 Proxy

The Proxy service provides a mechanism to cache one message for systems residing on low-power, resource-constrained wireless devices.

4.2.3 Historian

The Historian service provides features for producers and consumers to interact with a database in order to store and fetch structured data.

5 References

[1] J. Delsing, “Iot automation : Arrowhead framework,” 2017.

[2] C. Jennings and Z. Shelby, “Sensor Measurement Lists (SenML),” RFC 8428, 2018, RFC Editor. [Online]. Available: https://doi.org/10.17487/RFC8428

[3] C. Bormann and P. Hoffman, “Concise Binary Object Representation (CBOR),” RFC 7049, 2013, RFC Editor. [Online]. Available: https://doi.org/10.17487/RFC7049

[4] “The Sensor Network Object Notation,” Online, 2019, . [Online]. Available: http://www.snon.org/

[5] H. Butler and M. Daly, “ The GeoJSON Format,” RFC 7946, 2016, RFC Editor. [Online]. Available: https://doi.org/10.17487/RFC7946

[6] Open Geospatial Consortium, “Sensor Markup Language,” Online, 2018, . [Online]. Available: https://www.ogc.org/standards/sensorml

6 Revision History

6.1 Amendments

No.	Date	Version	Subject of Amendments	Author
1	2018-09-21	G4.0	First draft	Jens Eliasson
2	2018-10-29	G4.0	Use cases updated	Jens Eliasson
3	2019-03-28	G4.0	text improvements	Jens Eliasson
4	2020-11-17	G4.1.3	Updated to 4.1.3	Jens Eliasson
5	2020-12-02	G4.1.3	New template	Jens Eliasson
6				

6.2 Quality Assurance

No.	Date	Version	Approved by
1			