

Historian HTTP/TLS/JSON

Interface Design Description

Service ID: *"historian"*

Abstract

This document describes the Historian service IDD for HTTP/TLS/JSON.



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]



ARROWHEAD

Document title
Historian HTTP/TLS/JSON
Date
2020-12-09

Version
0.2
Status
DRAFT
Page
2 (8)

Contents

1 Overview	3
2 Service Interfaces	4
2.1 function ListSystems	4
2.2 function ListSystemService s	4
2.3 function DataStore	4
2.4 function DataFetch	5
3 Information Model	6
3.1 struct SensorData	6
4 References	7
5 Revision History	8
5.1 Amendments	8
5.2 Quality Assurance	8



ARROWHEAD

Document title
Historian HTTP/TLS/JSON
Date
2020-12-09

Version
0.2
Status
DRAFT
Page
3 (8)

1 Overview

This document describes the HTTP/TLS/JSON variant of the Historian Eclipse Arrowhead service. The Historian service is used to exchange messages between a data producing systems and a data consuming systems through the use of a database. One of the design goals is to enable large amounts of data to be stored and retrieved. The Historian service also enables consumers to filter outgoing data by time or value thresholds.

This document exists as a complement to the *Historian – Service Description* (Historian SD) document. For further details about how this service is meant to be used, please consult that document. The rest of this document describes how to realize the Historian service using HTTP [1], TLS [2] and JSON [3], both in terms of its interfaces (Section 2) and its information model (Section 3).

2 Service Interfaces

This section lists the interfaces that must be exposed by Historian services in alphabetical order. In particular, each subsection first names the HTTP method and path used to call the interface, after which it names an abstract interface from the Historian service's SD document, as well as input and output types. All interfaces in this section respond with the HTTP status code 202 OK if called successfully, unless otherwise is stated.

2.1 GET /datamanager/historian

Interface: [ListSystems](#)

Called to list of all active Arrowhead systems, as exemplified in Listing 1.

```
1 GET /datamanager/historian HTTP/1.1
2
3 [
4   "examplesystem1", "examplesystem2", "examplesystemN"
5 ]
```

Listing 1: A [SystemList](#) list.

Code	Type	Description
200	OK	No error
401	UNAUTHORIZED	No valid authorization
500	INTERNAL SERVER ERROR	Database error etc

Table 1: ListSystems responses

2.2 GET /datamanager/historian/\$systemName

Interface: [ListSystemServices](#)

Called to get a list of all service endpoints that a specific system has pushed data to using [Store](#). An example is given in Listing 2.

```
1 GET /datamanager/historian/examplesystem1 HTTP/1.1
2
3 [
4   "service1", "service1", "service2", "serviceN"
5 ]
```

Listing 2: A [GetSystemService](#) invocation.

Code	Type	Description
200	OK	No error
400	BAD REQUEST	Bad input
401	UNAUTHORIZED	No valid authorization
500	INTERNAL SERVER ERROR	Database error etc

Table 2: GetSystemService responses

2.3 PUT /datamanager/historian/\$systemName/\$serviceName

Interface: **DataStore**
Input: **SensorData**

Called to store sensor data in a service endpoint's database. An example is given in Listing 3.

```
1 PUT /datamanager/historian/examplesystem1/service1 HTTP/1.1
2
3 [
4   {"bn": "examplesystem1", "bt": 1.276020076001e+09, "bu": "Cel", "bver": 5,
5     {"n": "temperature", "t": 0, "v": 22.2},
6     {"n": "temperature", "t": -15, "v": 21.9}
7 ]
```

Listing 3: A **DataStore** invocation with a SenML-encoded message.

Code	Type	Description
200	OK	No error
400	BAD REQUEST	Bad input
401	UNAUTHORIZED	No valid authorization
500	INTERNAL SERVER ERROR	Database error etc

Table 3: DataStore responses

2.4 GET /datamanager/historian/\$systemName/\$serviceName

Interface: **DataFetch**
Output: **SensorData**

Called to retrieve already stored sensor data at a service endpoint's database. An example is given in Listing 4.

```
1 GET /datamanager/historian/examplesystem1/service1 HTTP/1.1
2
3 [
4   {"bn": "examplesystem1", "bt": 1.276020076001e+09, "bu": "Cel", "bver": 5,
5     {"n": "temperature", "t": 0, "v": 22.2},
6     {"n": "temperature", "t": -15, "v": 21.9}
7 ]
```

Listing 4: A **DataFetch** Response with a SenML-encoded message.

An set of valid input parameters are given in Table 4.

Name	Type	Description
to	Unix timestamp	End time for time series
from	Unix timestamp	Start time for time series
count	Integer	Maximum number of points in the response
sigX	String	Restrict points belonging to specific signal, i.e. sig0=temperature&sig=humidity
sigXcount	Integer	Limit number of points per signal, i.e. sig0count=10&sig1count=5

Table 4: DataFetch input parameters

Code	Type	Description
200	OK	No error
400	BAD REQUEST	Bad input
401	UNAUTHORIZED	No valid authorization
404	NOT FOUND	No such system/service combination
500	INTERNAL SERVER ERROR	Database error etc

Table 5: DataFetch responses

3 Information Model

The SensorData request payload contains generic sensor data. SensorData normally contains information about the unit, source system, timestamp and metadata. The default payload type is JSON-encoded SenML (RFC 8428). The response to a Store/ Retrieve request is a simple HTTP status code (Created/OK – request was a success, No Content – request had no effect). For the Push interface the content-type must be set to 'application/json'. For Fetch, the response content-type is 'application/json'.

3.1 struct **SensorData**

All messages must be encoded using SenML [4] with JSON.

Object Field	Value Type	Description
"bn"	Base name	Base name of a message.
"bt"	Base time	Base time.
"bu"	Base unit	Base unit.
"bver"	Base version	Base version (not used).
"n"	Signal name	Name of sensor signal measurement.
"t"	Time	Time of measurement.
"u"	Unit	Unit of measurement.

4 References

- [1] R. Fielding and J. Reschke, "Hypertext Transfer Protocol (HTTP/1.1): Message Syntax and Routing," RFC 7230, 2018, RFC Editor. [Online]. Available: <https://doi.org/10.17487/RFC7230>
- [2] E. Rescorla, "The Transport Layer Security (TLS) Protocol Version 1.3," RFC 8446, 2018, RFC Editor. [Online]. Available: <https://doi.org/10.17487/RFC8446>
- [3] T. Bray, "The JavaScript Object Notation (JSON) Data Interchange Format," RFC 7159, 2014, RFC Editor. [Online]. Available: <https://doi.org/10.17487/RFC7159>
- [4] C. Jennings and Z. Shelby, "Sensor Measurement Lists (SenML)," RFC 8428, 2018, RFC Editor. [Online]. Available: <https://doi.org/10.17487/RFC8428>



ARROWHEAD

Document title
Historian HTTP/TLS/JSON
Date
2020-12-09

Version
0.2
Status
DRAFT
Page
8 (8)

5 Revision History

5.1 Amendments

No.	Date	Version	Subject of Amendments	Author
1	2018-09-17	4.0	Initial	Jens Eliasson
2	2018-10-30	4.0	Text update	Jens Eliasson
3	2019-03-20	4.0	Updated data model to RFC 8428	Jens Eliasson
4	2020-05-07	4.1.3	Updated text	Jens Eliasson
5	2020-11-30	4.1.3	New template	Jens Eliasson

5.2 Quality Assurance

No.	Date	Version	Approved by
1			