Authorization Publickey HTTP/TLS/JSON

Interface Design Description

Service ID: *"publickey"*

Abstract
This document describes a HTTP/TLS/JSON variant of the Authorization Public Key service.

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 Authorization Public Key Eclipse Arrowhead service, which enables clients to get the public key of the Auhtorization Core System. Examples of this interaction is a system that wants to check the public key of the Authorization Core System..

This document exists as a complement to the *Authorization Public Key – Service Description* 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 Auhtorization Public Key service using HTTP [1], TLS [2] and JSON [3], both in terms of its functions (Section 2) and its information model (Section 3).

2 Service Functions

This section lists the functions that must be exposed by the Auhtorization Public Key service in alphabetical order. In particular, each subsection first names the HTTP method and path used to call the function, after which it names an abstract function from the Auhtorization Public Key SD document, as well as input and output types. All functions in this section respond with the HTTP status code `200 Created` if called successfully. The error codes are, `400 Bad Request` if request is malformed, `401 Unauthorized` if improper client side certificate is provided, `500 Internal Server Error` if Service Registry is unavailable.

2.1 GET /authorization/publickey

Interface: Publickey
Input: Publickey

Called to get the public key of the Authorization Core System, as exemplified in Listing 2.

```
1 GET /authorization/publickey HTTP/1.1
```

Listing 1: A Publickey invocation.

Response of the call above:

```
1 MIIBIjANBgkqhkiG9w0BAQEFAAOCAQ8AMIIBCgKCAQEAwms8AvBuIxqPjXmyGnqds1EIkvX/kjl+kW9a0SObsp1n/u567vbpYSa+
    ESZNg4KrxAHJjA8M1TvpGkq4LLrJkEUkC2WNxq3qbWQbseZrIDSpcn6C7gHObJOLjRSpGTSlRHZfncRs1h+
    MLApVhf6qf611mZNDgN5AqaMtBbB3UzArE3CgO0jiKzBgZGyT9RSKccjlsO6amBgZrLBY0+x6VXPJK71hwZ7/1
    Y2CHGsgSb20/g2P82qLYf91Eht33u01rcptsETsvGrsq6SqIKtHtmWkYMW1lWB7p2mwFpAft8llUpHewRRAU1qsKYAI6myc/
    sPmQuQul+4yESMSBu3KyQIDAQAB
```

Listing 2: A Publickey response

3 Information Model

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 3.1, which are used to represent things like hashes, identifiers and texts.

3.1 Primitives

As all messages are encoded using the JSON format [3], the following primitive constructs, part of that standard, become available. Note that the official standard is defined in terms of parsing rules, while this list only concerns syntactic information. Furthermore, the Object and Array types are given optional generic type parameters, which are used in this document to signify when pair values or elements are expected to conform to certain types.

JSON Type	Description
String	An arbitrary UTF-8 string.

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

5 Revision History

5.1 Amendments

No.	Date	Version	Subject of Amendments	Author
1	2020-12-05	1.0.0		Szvetlin Tanyi

5.2 Quality Assurance

No.	Date	Version	Approved by
1	2021-01-26	4.3.0	Jerker Delsing