AuthorizationControl HTTP/TLS/JSON

Interface Design Description

Service ID: *"token-generation"*

Abstract
This document describes a HTTP/TLS/JSON variant of the TokenGeneration 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 HTTP/TLS/JSON variant of the TokenGeneration Eclipse Arrowhead service, which enables access token generation for clients. Examples of this interaction is the Orchestrator during the orchestration process needs an authorization token for the consumer to be able to use the requested service.

This document exists as a complement to the *TokenGeneration – 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 TokenGeneration 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 Authorization Control 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 TokenGeneration 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/token

Interface: TokenGeneration
Input: TokenRule
Output: TokenData

Called to generate a token for the client with the given properties, as exemplified in Listing 1.

```
POST /authorization/token HTTP/1.1

{
  "consumer": {
    "address": "string",
    "authenticationInfo": "string",
    "port": 0,
    "systemName": "string"
  },
  "consumerCloud": {
    "authenticationInfo": "string",
    "gatekeeperRelayIds": [
      0
    ],
    "gatewayRelayIds": [
      0
    ],
    "name": "string",
    "neighbor": true,
    "operator": "string",
    "secure": true
  },
  "duration": 0,
  "providers": [
    {
      "provider": {
        "address": "string",
        "authenticationInfo": "string",
        "port": 0,
        "systemName": "string"
      },
      "serviceInterfaces": [
        "string"
      ]
    }
  ],
  "service": "string"
}
```

Listing 1: An Token invocation response.

Response of the call above:

```

{
  "tokenData": [
    {
```

```
      "providerAddress": "string",
      "providerName": "string",
      "providerPort": 0,
      "tokens": {
        "additionalProp1": "string",
        "additionalProp2": "string",
        "additionalProp3": "string"
      }
    }
  ]
}
```


Listing 2: A Check an InterCloud Rule InterCloudRresponse

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

3.1 struct TokenData

This structure is the response of the TokenRule

Object Field	Value Type	Description
”tokenData”	Array<TokenInfo>	Array of Token related information.

3.2 struct TokenInfo

This structure is used to describe a TokenInfo Object.

Object Field	Value Type	Description
”providerAddress”	String	Provider Address.
”providerName”	Name	Provider Name.
”providerPort”	Number	Provider Port.
”tokens”	Object	Token key value pairs.

3.3 struct TokenRule

This structure is used to describe a TokenData Object.

Object Field	Value Type	Description
”consumer”	Consumer	Consumer.
”consumerCloud”	Cloud	Neighboring Cloud.
”duration”	Number	Validity of the token in seconds.
”providers”	Array<Provider>	List of Providers and interfaces.
”service”	Name	Service Definition

3.4 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.

With these primitives now available, we proceed to define all the types specified in the Service Discovery Register SD document without a direct equivalent among the JSON types. Concretely, we define the Service Discovery Register SD primitives either as *aliases* or *structs*. An *alias* is a renaming of an existing type, but with some further details about how it is intended to be used. Structs are described in the beginning of the parent section. The types are listed by name in alphabetical order.

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 [4], except for *+Inf*, *-Inf* and *NaN*.
Boolean	One out of `true` or `false`.
Null	Must be `null`.

3.4.1 alias Name = String

A String that is meant to be short (less than a few tens of characters) and both human and machine-readable.

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] 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	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