AuthorizationControl HTTP/TLS/JSON

Interface Design Description

Service ID: *"authorization-control"*

Abstract

This document describes a HTTP/TLS/JSON variant of the AuthorizationControl 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 Authorization Control Eclipse Arrowhead service, which enables enforcing intracloud and intercloud authorization rules. Examples of this interaction is to check if the requested consumer system is allowed to consume the given service by the specified provider.

This document exists as a complement to the *AuthorizationControl – 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 AuthorizationControl 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 Authorization Control 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/echo

Interface: Echo
Output: StatusCodeKind

Called to check the core systems availability, as exemplified in Listing 1.

```
GET /authorization/echo HTTP/1.1

Got it!
```

Listing 1: An Echo invocation response.

2.2 POST /authorization/intercloud/check

Interface: Check an Intercloud Rule
Input: InterCloudRule
Output: InterCloudResponse

Called to check whether the cloud is authorized to use a service. 4.

```
POST/authorization/intercloud/check HTTP/1.1

{
  "cloud": {
    "authenticationInfo": "string",
    "gatekeeperRelayIds": [
      0
    ],
    "gatewayRelayIds": [
      0
    ],
    "name": "string",
    "neighbor": true,
    "operator": "string",
    "secure": true
  },
  "providerIdsWithInterfaceIds": [
    {
      "id": 0,
      "idList": [
        0
      ]
    }
  ],
  "serviceDefinition": "string"
}
```

Listing 2: A Check an Intracloud Rule invocation with IntraCloudRule payload.

Response of the call above:

```
{
  "authorizedProviderIdsWithInterfaceIds": [
    {
      "id": 0,
      "idList": [
        0
      ]
    }
  ],
  "cloud": {
    "authenticationInfo": "string",
    "createdAt": "string",
    "id": 0,
    "name": "string",
    "neighbor": true,
    "operator": "string",
    "ownCloud": true,
    "secure": true,
    "updatedAt": "string"
  },
  "serviceDefinition": "string"
}
```

Listing 3: A Check an InterCloud Rule InterCloudRresponse

2.3 POST /authorization/intracloud/check

Interface: Check an Intracloud Rule
Input: IntraCloudRule
Output: IntraCloudResponse

Called to check whether the consumer System can use a Service from a list of provider Systems, as exemplified in Listing 4.

```
POST/authorization/intracloud/check HTTP/1.1

{
  "consumer": {
    "address": "192.168.0.101",
    "authenticationInfo": "public key",
    "port": 8080,
    "systemName": "ExampleConsumerSystem"
  },
  "providerIdsWithInterfaceIds": [
    {
      "id": 12,
      "idList": [
        3,4
      ]
    }
  ],
  "serviceDefinitionId": 15
}
```

Listing 4: A Check an Intracloud Rule invocation with IntraCloudRule payload.

Response of the call above:

```
{
  "authorizedProviderIdsWithInterfaceIds": [
    {
      "id": 12,
      "idList": [
        3,4
      ]
    }
  ],
  "consumer": {
    "address": "192.168.0.101",
    "authenticationInfo": "public key",
    "createdAt": "2020-12-05 12:00:00",
    "id": 37,
    "port": 8080,
    "systemName": "ExampleConsumerSystem",
    "updatedAt": "2020-12-05 12:00:00"
  },
  "serviceDefinitionId": 15
}
```

Listing 5: A Check an Intracloud Rule IntraClouRresponse

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

3.1 struct AuthorizedProvider

This structure is used to describe an AuthorizedProvider Object.

Object Field	Value Type	Description
"id"	Number	Provider ID.
"idList"	Array<RandomID>	List of Interface IDs.

3.2 struct InterCloudResponse

This structure is the response of the IntraCloudRule

Object Field	Value Type	Description
"authorizedProviderIdsWithInterfaceIds"	Array<AuthorizedProviders>	Authorized Providers.
"cloud"	Cloud	A cloud.
"serviceDefinitionId"	Number	Service Definition ID.

3.3 struct InterCloudRule

This structure is used to check whether the consumer system can use a service from a list of provider systems.

Object Field	Value Type	Description
"cloud"	Cloud	A cloud.
"providerIdsWithInterfaceIds"	Array<ProviderIDsWithInterfaceIDs>	Provider IDs with Interface IDs.
"serviceDefinitionId"	Number	Service Definition ID.

3.4 struct IntraCloudResponse

This structure is the response of the IntraCloudRule

Object Field	Value Type	Description
"authorizedProviderIdsWithInterfaceIds"	Array<AuthorizedProviders>	Authorized Providers.
"consumer"	Consumer	A consumer.
"serviceDefinitionId"	Number	Service Definition ID.

3.5 struct IntraCloudRule

This structure is used to check whether the consumer system can use a service from a list of provider systems.

Object Field	Value Type	Description
”consumer”	Consumer	A consumer.
”providerIdsWithInterfaceIds”	Array<ProviderIDsWithInterfaceIDs>	Provider IDs with Interface IDs.
”serviceDefinitionId”	Number	Service Definition ID.

3.6 struct Consumer

This structure is used to describe a Consumer *Object*.

Object Field	Value Type	Description
”address”	String	Consumers address.
”authenticationInfo”	String	Consumers public key.
”createdAt”	DateTime	Date of creation (contained in IntraCloudResponse).
”id”	RandomID	ID of the consumer (contained in IntraCloudResponse).
”port”	Number	Port.
”systemName”	String	System Name.
”updatedAt”	DateTime	Date of last modification (contained in IntraCloudResponse).

3.7 struct Cloud

This structure is used to describe a Cloud *Object*.

Object Field	Value Type	Description
”authenticationInfo”	String	Clouds public key.
”createdAt”	DateTime	Date of creation (contained in InterCloudResponse).
”id”	RandomID	ID of the consumer (contained in InterCloudResponse).
”name”	Name	Name of the Cloud
”neighbor”	Boolean	Neighbor or not
”operator”	Name	Operator company name
”ownCloud”	Boolean	Owncloud or not.
”secure”	Boolean	Is the cloud using security or not.
”updatedAt”	DateTime	Date of last modification (contained in InterCloudResponse).

3.8 struct ProviderIDsWithInterfaceIDs

This structure is used to describe a provider IDs with Interface IDs *Object*.

3.9 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

Object Field	Value Type	Description
"id"	RandomID	Provider ID.
"idList"	Array<Number>	List of Interface IDs.

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
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`.

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.

3.9.1 alias DateTime = String

Pinpoints a moment in time in the format of "YYYY-MM-DD HH:mm:ss", where "YYYY" denotes year (4 digits), "MM" denotes month starting from 01, "DD" denotes day starting from 01, "HH" denotes hour in the 24-hour format (00-23), "MM" denotes minute (00-59), "SS" denotes second (00-59). " " is used as separator between the date and the time. An example of a valid date/time string is "2020-12-05 12:00:00"

3.9.2 alias Name = String

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

3.9.3 alias RandomID = Number

An integer Number, originally chosen from a secure source of random numbers. When new RandomIDs are created, they must be ensured not to conflict with any relevant existing random numbers.

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