Config HTTP/TLS/JSON

Interface Design Description

Service ID: *"config"*

Abstract
This document describes the Config 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]

Contents

1 Overview

This document describes the HTTP/TLS/JSON variant of the Config Eclipse Arrowhead service. The Config service is used to store and manage configurations and settings for application systems in a local cloud. The Config service also enables cloud operators (sysop) to create new, delete and list configurations.

This document exists as a complement to the *Config – Service Description* (Config 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 Config 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 the Config service 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 Config service's SD document, as well as input and output types. All interfaces in this section respond with the HTTP status code `200 OK` if called successfully, unless otherwise is stated.

2.1 GET /configuration/config/$systemName

Interface: GetConfig
Output: ConfigurationResponse

Called to fetch the active configuration for a specific system. The response is a exemplified in Listing 1.

```
GET /configuration/config/examplesystem1 HTTP/1.1

{
  "id": 31,
  "systemName": "examplesystem1",
  "contentType": "text/plain",
  "data": "cG9ydD04ODAxCnBhdGg9ImV4YW1wbGUvcGF0aCIKaXA9IjEwLjAuMC4yMyI=",
  "createdAt": "2021-01-26 22:23:13",
  "updatedAt": "2021-01-26 22:23:13"
}
```

Listing 1: A GetConfig invocation.

Code	Type	Description
200	OK	No error
401	UNAUTHORIZED	No valid authorization
404	NOT FOUND	No valid configuration found
500	INTERNAL SERVER ERROR	Database error etc

Table 1: GetConfig status code responses

2.2 GET /configuration/config/raw/$systemName

Interface: GetRawConfig

Called to get a black-box or binary raw configuration file that has been stored using Store. The output is Base64-decoded data (without the encapsulating JSON object). An example of a plain text configuration file is given in Listing 2.

```
GET /configuration/config/raw//examplesystem1 HTTP/1.1

port=8801
path="example/path"
ip="10.0.0.23"
```

Listing 2: A GetRawConfig invocation.

Code	Type	Description
200	OK	No error
400	BAD REQUEST	Bad input
401	UNAUTHORIZED	No valid authorization
404	NOT FOUND	No valid configuration found
500	INTERNAL SERVER ERROR	Database error etc

Table 2: GetRawConfig status code responses

2.3 GET /configuration/mgmt/config

Interface: ManagementListConfigurations
Output: ConfigurationListResponse

Called to get a complete list of all configuration objects in the database. An example is given in Listing 4.

```
GET /configuration/mgmt/config HTTP/1.1

RESPONSE:
{
  "count": 1,
  "data": [
    {
      "id": 1,
      "systemName": "examplesystem1",
      "contentType": "text/plain",
      "data": "cG9ydD04ODAxCnBhdGg9ImV4YW1wbGUvcGF0aCIKaXA9IjEwLjAuMC4yMyI=",
      "createdAt": "2021-01-26 22:23:13",
      "updatedAt": "2021-01-26 22:23:13"
    }
  ]
}
```

Listing 3: A ManagementListConfigurations invocation with an JSON-encoded response.

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: ManagementListConfigurations responses

2.4 PUT /configuration/mgmt/config/$systemName

Interface: ManagementStoreConfiguration
Input: ConfigurationRequest
Output: ConfigurationResponse

Called to store configuration data. Only the local cloud operator (sysop) can perform this action. An example is given in Listing 4.

```
PUT /configuration/mgmt/config/examplesystem1 HTTP/1.1

REQUEST:
{
```

```
  "systemName": "examplesystem1",
  "contentType": "text/plain",
  "data": "cG9ydD04ODAxCnBhdGg9ImV4YW1wbGUvcGF0aCIKaXA9IjEwLjAuMC4yMyI="
}

RESPONSE:
 {
  "id": 1,
  "systemName": "examplesystem1",
  "contentType": "text/plain",
  "data": "cG9ydD04ODAxCnBhdGg9ImV4YW1wbGUvcGF0aCIKaXA9IjEwLjAuMC4yMyI=",
  "createdAt": "2021-01-26 22:23:13",
  "updatedAt": "2021-01-26 22:23:13"
}
```

Listing 4: A ManagementStoreConfiguration Response with JSON-encoded data.

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 4: ManagementStoreConfiguration responses

2.5 DELETE /configuration/mgmt/config/$systemName

Interface: ManagementDeleteConfiguration
Output: ConfigurationResponse

Called to delete a configuration object in the database. An example is given in Listing 5. The deleted entry is also returned.

```
DELETE /configuration/mgmt/config/examplesystem1 HTTP/1.1

RESPONSE:
{
  "id": 1,
  "systemName": "examplesystem1",
  "contentType": "text/plain",
  "data": "cG9ydD04ODAxCnBhdGg9ImV4YW1wbGUvcGF0aCIKaXA9IjEwLjAuMC4yMyI=",
  "createdAt": "2021-01-26 22:23:13",
  "updatedAt": "2021-01-26 22:23:13"
}
```

Listing 5: A ManagementDeleteConfiguration invocation with an JSON-encoded response.

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 5: ManagementDeleteConfiguration responses

3 Information Model

The Config service does not restrict the information model. It supports JSON, plain text and black-box / binary data.

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	2021-01-15	0.1	Initial	Jens Eliasson
2	2021-01-25	0.5	Text update	Jens Eliasson
3	2021-02-06	0.9	Data models	Jens Eliasson
4	2021-02-09	1.0	Updated service interfaces	Jens Eliasson

5.2 Quality Assurance

No.	Date	Version	Approved by
1			