orchestration-create-flexible-store-rules HTTP/TLS/JSON

Interface Design Description

Abstract
This document describes a HTTP protocol with TLS payload security and JSON payload encoding variant of the **orchestration-create-flexible-store-rules** service.

Contents

1 Overview

This document describes the **orchestration-create-flexible-store-rules** service interface, which enables systems to add flexible matching rules to the Orchestrator Core System (if the Orchestrator is in flexible store mode). It's implemented using protocol, encoding as stated in the following table:

Profile type	Type	Version
Transfer protocol	HTTP	1.1
Data encryption	TLS	1.3
Encoding	JSON	RFC 8259 [1]
Compression	N/A	-

Table 1: Communication and semantics details used for the **orchestration-create-flexible-store-rules** service interface

This document provides the Interface Design Description IDD to the *orchestration-create-flexible-store-rules – 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 orchestration-create-flexible-store-rules service HTTP/TLS/JSON interface in details.

2 Interface Description

The service responses with the status code `201 Created` if called successfully. The error codes are, `400 Bad Request` if request is malformed or the Orchestrator is not in flexible store mode, `401 Unauthorized` if improper client side certificate is provided, `500 Internal Server Error` if Orchestrator is unavailable.

```
POST /orchestrator/store/felxible HTTP/1.1

[{
  "consumerSystem": {
    "systemName": "testConsumer"
  },
  "providerSystem": {
    "metadata": {
      "mode": "legacy"
    }
  },
  "serviceDefinitionName": "temperature",
  "servuceInterfaceName": "HTTP-SECURE-JSON"
  "serviceMetadata": {
    "unit": "celsius"
  },
  "priority": 1
}]
```

Listing 1: An orchestration-create-flexible-store-rules invocation.

```
{
  "count": 1,
  "data": [{
    "id": 1,
    "consumerSystem": {
      "systemName": "testConsumer"
    },
    "providerSystem": {
      "metadata": {
        "mode": "legacy"
      }
    },
    "serviceDefinitionName": "temperature",
    "servuceInterfaceName": "HTTP-SECURE-JSON"
    "serviceMetadata": {
      "unit": "celsius"
    },
    "priority": 1,
    "createdAt": "2020-03-18T22:13:32.143",
    "updatedAt": "2020-03-18T22:13:32.143"
  }]
}
```

Listing 2: An orchestration-create-flexible-store-rules response.

3 Data Models

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

3.1 struct FlexibleRule

Field	Type	Mandatory	Description
consumerSystem	SystemDescriptor	yes	Requirements for a system that tells if this rule is applied to a consumer.
priority	Number	no	Priority of the rule.
providerSystem	SystemDescriptor	yes	Requirements for a system that tells if this rule is applied to a provider.
serviceDefinition	Name	yes	Identifier of a service.
serviceInterfaceName	Interface	no	Interface requirement
serviceMetadata	Metadata	no	Service instance level metadata requirements.

3.2 struct SystemDescriptor

Field	Type	Mandatory	Description
systemName	Name	no (yes)	Name of a system. Mandatory if *metadata* is not specified.
metadata	Metadata	no (yes)	System level metadata requirements (a system's metadata must contain all of its keys with the same value to match). Mandatory if *systemName* is not specified.

3.3 struct Metadata

An Object which maps String key-value pairs.

3.4 struct FlexibleRuleListResponse

Field	Type	Description
count	Number	The number of created rules.
data	List<FlexibleRuleResponse>	Created rule records.

3.5 struct FlexibleRuleResponse

Field	Type	Description
consumerSystem	SystemDescriptor	Requirements for a system that tells if this rule is applied to a consumer.
createdAt	DateTime	Rule record was created at this UTC timestamp.
id	Number	Identifier of the rule.
priority	Number	Priority of the rule.
providerSystem	SystemDescriptor	Requirements for a system that tells if this rule is applied to a provider.
serviceDefinition	Name	Identifier of a service.
serviceInterface	Interface	Interface requirement
serviceMetadata	Metadata	Service instance level metadata requirements.
updatedAt	DateTime	Rule record was modified at this UTC timestamp.

3.6 Primitives

As all messages are encoded using the JSON format [2], 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
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 [3], 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 **orchestration-create-flexible-store-rules** SD document without a direct equivalent among the JSON types. Concretely, we define the **orchestration-create-flexible-store-rules** 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.

3.6.1 alias DateTime = String

Pinpoints a moment in time in the format of ISO8601 standard ”yyyy-mm-ddThh:mm:ss”, where ”yyy” denotes year (4 digits), ”mm” denotes month starting from 01, ”dd” denotes day starting from 01, ”T” is the separator between date and time part, ”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 ”2020-12-05T12:00:00”

3.6.2 alias Interface = String

A String that describes an interface in *Protocol-SecurityType-MimeType* format. *SecurityType* can be SECURE or INSECURE. *Protocol* and *MimeType* can be anything. An example of a valid interface is: ”HTTPS-SECURE-JSON” or ”HTTP-INSECURE-SENML”.

3.6.3 alias List <A> = Array<A>

There is no difference.

3.6.4 alias Name = String

A String indentifier that is intended to be both human and machine-readable.

4 References

[1] T. Bray, “The JavaScript Object Notation (JSON) Data Interchange Format,” RFC 8259, Dec. 2017. [Online]. Available: https://rfc-editor.org/rfc/rfc8259.txt

[2] ——, “The JavaScript Object Notation (JSON) Data Interchange Format,” RFC 7159, 2014, RFC Editor. [Online]. Available: https://doi.org/10.17487/RFC7159

[3] 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	YYYY-MM-DD	4.6.0		Xxx Yyy

5.2 Quality Assurance

No.	Date	Version	Approved by
1	YYYY-MM-DD	4.6.0	Xxx Yyy