orchestration-service-by-proxy 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-service-by-proxy** service.

Contents

1 Overview

This document describes the **orchestration-service-by-proxy** service interface, that provides runtime (late) binding between application systems. 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-service-by-proxy** service interface

This document provides the Interface Design Description IDD to the *orchestration-service-by-proxy – 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-service-by-proxy** service HTTP/TLS/JSON interface in details.

2 Interface Description

The service responses with the status code `200 Ok` 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 Orchestrator is unavailable.

```
POST /orchestrator/orchestration-by-proxy HTTP/1.1

{
  "requesterSystem": {
    "systemName": "string",
    "address": "string",
    "port": 0,
    "authenticationInfo": "string"
  },
  "requestedService": {
    "serviceDefinitionRequirement": "string",
    "interfaceRequirements": [
      "string"
    ],
    "securityRequirements": [
      "CERTIFICATE", "TOKEN"
    ],
    "metadataRequirements": {
      "additionalProp1": "string",
      "additionalProp2": "string",
      "additionalProp3": "string"
    },
    "versionRequirement": 0,
    "maxVersionRequirement": 0,
    "minVersionRequirement": 0
  },
  "preferredProviders": [
    {
      "providerCloud": {
        "operator": "string",
        "name": "string"
      },
      "providerSystem": {
        "systemName": "string",
        "address": "string",
        "port": 0
      }
    }
  ],
  "orchestrationFlags": {
    "overrideStore": true,
    "matchmaking": true,
    "enableQoS": true
  },
  "qosRequirements": {
    "qosMaxRespTimeThreshold": "1000"
  }
}
```

Listing 1: An orchestration-service-by-proxy invocation.

```
{
  "response": [
    {
      "provider": {
        "id": 0,
        "systemName": "string",
        "address": "string",
        "port": 0,
        "authenticationInfo": "string",
        "metadata": {
          "additionalProp1": "string",
          "additionalProp2": "string",
          "additionalProp3": "string"
        },
        "createdAt": "string",
        "updatedAt": "string"
      },
      "service": {
        "id": 0,
        "serviceDefinition": "string",
        "createdAt": "string",
        "updatedAt": "string"
      },
      "serviceUri": "string",
      "secure": "TOKEN",
      "metadata": {
        "additionalProp1": "string",
        "additionalProp2": "string",
        "additionalProp3": "string"
      },
      "interfaces": [
        {
          "id": 0,
          "createdAt": "string",
          "interfaceName": "string",
          "updatedAt": "string"
        }
      ],
      "version": 0,
      "authorizationTokens": {
        "interfaceName1": "token1",
        "interfaceName2": "token2"
      },
      "warnings": [
        "FROM_OTHER_CLOUD", "TTL_UNKNOWN"
      ]
    }
  ]
}
```


Listing 2: An orchestration-service-by-proxy 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 3.13, which are used to represent things like hashes, identifiers and texts.

3.1 struct OrchestrationForm

Field	Type	Mandatory	Description
commands	Metadata	no	Additional commands to the Orchestrator, the only available command now is `qosExclusivity` (see in the Service Description document).
orchestrationFlags	OrchestrationFlags	no	A map of flags that changes the behaviour of the service. See details in the Service Description document.
preferredProviders	List<PreferredProvider>	no	A list of providers that takes precedence in matchmaking if they are available; if `onlyPreferred` flag is set, then the result can only be a subset of this list.
qosRequirements	Metadata	no	Quality-of-Service requirement map. See details in the Service Description document.
requestedService	ServiceQueryForm	no (yes)	Information about the requested service; mandatory in case of dynamic or flexible store orchestration.
requesterCloud	Cloud	no	Information about the cloud from which the request comes. Only specified when the request comes from an other cloud.
requesterSystem	System	yes	Information about the executor system that will consume the service.

3.2 struct Metadata

An Object which maps String key-value pairs.

3.3 struct OrchestrationFlags

An Object which maps String keys to Boolean values.

3.4 struct PreferredProvider

Field	Type	Mandatory	Description
providerCloud	Cloud	no	Information about the cloud of the preferred system. Need only specified when the system is in an other cloud.
providerSystem	System	yes	Information about the preferred system.

3.5 struct Cloud

Field	Type	Mandatory	Description
name	Name	yes	Name of the cloud.
operator	Name	yes	Operator of the cloud.

3.6 struct System

Field	Type	Mandatory	Description
address	Address	yes	Network address of the system.
authenticationInfo	String	no	X.509 public key of the system.
metadata	Metadata	no	Additional information about the system.
port	PortNumber	yes	Port of the system.
systemName	Name	yes	Name of the system.

3.7 struct ServiceQueryForm

Field	Type	Mandatory	Description
interfaceRequirements	List<Interface>	no	Names of the required interfaces. If specified at least one of the interfaces must match for having result(s).
maxVersionRequirement	Number	no	Required maximum version of the service. If specified version must be equal or lower for having result(s). Ignored if `versionRequirement` is specified.
metadataRequirements	Metadata	no	Service metadata requirements. If specified the whole content of the map must match for having result(s). Only applied if the `metadataSearch` flag is set to true.
minVersionRequirement	Number	no	Required minimum version of the service. If specified version must be equal or higher for having result(s). Ignored if `versionRequirement` is specified.
pingProviders	Boolean	no	Whether or not the provider should be pinged. If true only the responding providers will comply. The orchestration flag `pingProviders` overrides this value.
securityRequirements	List<SecureType>	no	Types of the required security levels. If specified at least one of the types must match for having result(s).
serviceDefinitionRequirement	Name	yes	Identifier of the service.
versionRequirement	Number	no	Required version of the service. If specified version must match for having result(s).

3.8 struct OrchestrationResultList

Field	Type	Description
response	List<OrchestrationResult>	List of orchestration results.

3.9 struct OrchestrationResult

Field	Type	Description
authorizationTokens	Metadata	Tokens to use the service instance (one for every supported interface). Only filled if the security type is `TOKEN`.
interfaces	List<ServiceInterfaceRecord>	List of interfaces the service instance supports.
metadata	Metadata	Service instance metadata.
provider	SystemRecord	Descriptor of the provider system record.
secure	SecureType	Type of security the service instance uses.
service	ServiceDefinitionRecord	Descriptor of the service definition record.
serviceUri	String	Path of the service on the provider.
version	Version	Version of the service instance.
warnings	List<OrchestratorWarning>	List of warnings about the provider and/or its service instance.

3.10 struct ServiceInterfaceRecord

Field	Type	Description
createdAt	DateTime	Interface instance record was created at this UTC timestamp.
id	Number	Identifier of the interface instance.
interfaceName	Interface	Specified name of the interface.
updatedAt	DateTime	Interface instance record was modified at this UTC timestamp.

3.11 struct SystemRecord

Field	Type	Description
address	Address	Network address of the system.
authenticationInfo	String	X.509 public key of the system.
createdAt	DateTime	System instance record was created at this UTC timestamp.
id	Number	Identifier of the system instance.
metadata	Metadata	Additional information about the system.
port	PortNumber	Port of the system.
systemName	Name	Name of the system.
updatedAt	DateTime	System instance record was modified at this UTC timestamp.

3.12 struct ServiceDefinitionRecord

Field	Type	Description
createdAt	DateTime	Service definition instance record was created at this UTC timestamp.
id	Number	Identifier of the service definition instance.
serviceDefinition	Name	Name of the service definition.
updatedAt	DateTime	Service definition instance record was modified at this UTC timestamp.

3.13 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-service-by-proxy** SD document without a direct equivalent among the JSON types. Concretely, we define the **orchestration-service-by-proxy** 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.

3.13.1 alias Address = String

A string representation of a network address. An address can be a version 4 IP address (RFC 791), a version 6 IP address (RFC 2460) or a DNS name (RFC 1034).

3.13.2 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”

3.13.3 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: ”HTTP-SECURE-JSON” or ”HTTP-INSECURE-SENML”.

3.13.4 alias List <A> = Array<A>

There is no difference.

3.13.5 alias Name = String

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

3.13.6 alias PortNumber = Number

Decimal Number in the range of 0-65535.

3.13.7 alias OrchestratorWarning = String

A String that represents a potentially interesting information about a provider and/or its service instance. Possible values are *FROM_OTHER_CLOUD* (if the provider is in an other cloud), *TTL_EXPIRED* (the provider is no longer accessible), *TTL_EXPIRING* (the provider will be inaccessible in a matter of minutes), *TTL_UNKNOWN* (the provider does not specified expiration time), *VIA_GATEWAY* (the provider is in an other cloud and only accessible via a tunnel provided by the Gateway Core System)

3.13.8 alias SecureType = String

A String that describes an the security type. Possible values are *NOT_SECURE* or *CERTIFICATE* or *TOKEN*.

3.13.9 alias Version = Number

A Number that represents the version of the service. And example of a valid version is: 1.

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