service-unregister HTTP/TLS/JSON

Interface Design Description

Abstract
This document describes a HTTP protocol with TLS payload security and JSON payload encoding variant of the **service-unregister** service.

Contents

1 Overview

This document describes the **service-unregister** service interface, which is enables autonomous service unregistration. It's implemented using protocol, encoding as stated in the following table:

Profile ype	Type	Version
Transfer protocol	HTTP	1.1
Data encryption	TLS	1.3
Encoding	URL	RFC 1738
Compression	N/A	-

Table 1: Communication and sematics details used for the **service-unregister** service interface

This document provides the Interface Design Description IDD to the *service-unregister – 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 service-unregister 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 Service Registry is unavailable.

```
1 DELETE /serviceregistry/unregister?service_definition={serviceDefinition}&system_name={providerName
      }&address={providerAddress}&port={providerPort}&service_uri={serviceUri} HTTP/1.1
```

Listing 1: A service-unregister invocation.

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

3.1 struct QueryParams

This structure is used to unregister a service from Service Registry.

Field	Type	Mandatory	Description
serviceDefinition	Name	yes	Identifier of the service.
providerName	Name	yes	Identifier of the provider system.
providerAddress	Address	no	Network address.
providerPort	PortNumber	yes	Port of the system.
serviceUri	URI	yes	URI of the service.

3.2 Primitives

Type	Description
Address	A string representation of the address
Name	A string identifier that is intended to be both human and machine-readable.
Number	Decimal number
PortNumber	Decimal number in the range of 0-65535
String	An arbitrary UTF-8 string.

With these primitives now available, we proceed to define all the types specified in the **service-uregister** SD document without a direct equivalent among the JSON types. Concretely, we define the **service-uregister** 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.

3.2.1 alias Name = String

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

3.2.2 alias PortNumber = Number

Decimal Number in the range of 0-65535.

3.2.3 alias URI = String

A String that represents the URL subpath where the offered service is reachable, starting with a slash (”/”). An example of a valid URI is ”/temperature”.

4 References

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