register-system

Service Description

Abstract
This document provides service description for the **register-system** service.

Contents

1 Overview

This document describes the **register-system** service, which enables autonomous system registration, therefore it is an integral part of the implementation of service discovery requirements in Service Registry Mandatory Core System. Examples of this interaction is a system that has the purpose of consuming some kind of service. To enable the system being authorized to consume, the system must be a registered member of the local cloud.

The rest of this document is organized as follows. In Section 2, we describe the abstract message functions provided by the service. In Section 3, we end the document by presenting the data types used by the mentioned functions.

1.1 How This Service Is Meant to Be Used

The given service consumer application system is required to use the **register-system** service at its startup or at least before looking for services.

1.2 Important Delimitations

The registration data must meet the following criteria:

- System name can't be what is reserved for core systems.
- System name can contain maximum 63 character of letters (english alphabet), numbers and dash (-), and have to start with a letter (also cannot end with dash).

1.3 Access policy

Available for anyone within the local cloud, but in case of secure mode the system is allowed to register only when its system name is matching to the system name part of its certificate common name.

2 Service Interface

This section describes the interfaces to the service. The **register-system** service is used to register system. A system could contain various metadata as well as a physical address. The various parameters are representing the necessary system input information. In particular, each subsection names an interface, an input type and an output type, in that order. The input type is named inside parentheses, while the output type is preceded by a colon. Input and output types are only denoted when accepted or returned, respectively, by the interface in question. All abstract data types named in this section are defined in Section 3.

The following interfaces are available.

2.1 interface HTTP/TLS/JSON (SystemRequest) : SystemResponse

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

Table 1: HTTP/TLS/JSON communication details.

3 Information Model

Here, all data objects that can be part of the **register-system** service provides to the hosting System are listed in alphabetic order. Note that each subsection, which describes one type of object, begins with the *struct* keyword, which is used to denote a collection of named fields, each with its own data type. As a complement to the explicitly defined types in this section, there is also a list of implicit primitive types in Section 3.3, which are used to represent things like hashes and identifiers.

3.1 struct SystemRequest

Field	Type	Mandatory	Description
address	Address	yes	Network address.
authenticationInfo	String	no	Public key of the client certificate.
metadata	Metadata	no	Metadata
port	PortNumber	yes	Port of the system.
systemName	Name	yes	Name of the system.

3.1.1 struct Metadata

A JSON Object which maps String key-value pairs.

3.2 struct SystemResponse

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

3.2.2 struct Metadata

A JSON Object which maps String key-value pairs.

3.3 Primitives

Types and structures mentioned throughout this document that are assumed to be available to implementations of this service. The concrete interpretations of each of these types and structures must be provided by any IDD document claiming to implement this service.

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 [2], except for *+Inf*, *-Inf* and *NaN*.
Boolean	One out of `true` or `false`.
Null	Must be `null`.

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] 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	