Document title
create-digital-twin HTTP/TLS/FORM
Date
2024-01-04
Author
Jesper Frisk
Contact
jesfri-8@student.ltu.se

Document type
IDD
Version
4.6.1
Status
RELEASE
Page
1 (7)

create-digital-twin HTTP/TLS/FORM

Interface Design Description

Abstract

This document describes a HTTP protocol with TLS payload security and Form-data payload encoding variant of the **create-digital-twin** service.

Contents

1 Overview

This document describes the **create-digital-twin** service interface, that provides the ability for system operators to create a new digital twin system. 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	-
Method	POST	-

Table 1: Communication and semantics details used for the **create-digital-twin** service interface

This document provides the Interface Design Description IDD to the *create-digital-twin – 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 **create-digital-twin** 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.

```
POST /create-digital-twin HTTP/1.1
{
  "controlCommands": [
    {
      "serviceDefinition": "string",
      "serviceUri": "string"
    }
  ],
  "physicalTwinConnection": {
    "connectionModel": {
      "address": "string",
      "port": 0
    },
    "connectionType": "string"
  },
  "sensedProperties": [
    {
      "intervalTime": 0,
      "sensorEndpointMode": "string",
      "serviceDefinition": "string",
      "serviceUri": "string"
    }
  ]
}
```

Listing 1: A create-digital-twin-service invocation.

```
{
  "address": "string",
  "authenticationInfo": "string",
  "port": 0,
  "systemName": "string"
}
```

Listing 2: A create-digital-twin-service 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 3.7, which are used to represent things like hashes, identifiers and texts.

3.1 struct DigitalTwinRequest

Field	Type	Mandatory	Description
controlCommands	List<ControlCommand>	no	Define control endpoints
physicalTwinConnection	PhysicalTwinConnection	yes	Connection settings to the physical twin.
sensedProperties	List<SensedProperty>	no	Define sensor endpoints

3.2 struct PhysicalTwinConnection

Field	Type	Mandatory	Description
connectionModel	ConnectionModel	yes	Defines parameters with how the digital twin connects to the physical twin.
connectionType	String	yes	Defines the type of connection that the digital twin uses to connect to the digital twin

3.3 struct ConnectionModel

Field	Type	Mandatory	Description
address	Address	yes	Network address.
port	PortNumber	yes	Network port.

3.4 struct ControlCommand

Field	Type	Mandatory	Description
serviceDefinition	String	yes	Definition of the service that will be registered that correspondes to the created endpoint.
serviceUri	String	yes	Uri path for the created endpoint.

3.5 struct SensedProperty

Field	Type	Mandatory	Description
intervalTime	String	no	If the ”INTERVAL_RETRIEVAL” sensor type is used then this defines the interval in seconds.
sensorEndpointMode	String	yes	Defines the type of sensor endpoint that will be used by the digital twin.
serviceDefinition	String	yes	Definition of the service that will be registered that correspondes to the created endpoint.
serviceUri	String	yes	Uri path for the created endpoint.

3.6 struct DigitalTwinResponse

Field	Type	Description
address	Address	he created digital twins network address.
authenticationInfo	String	X.509 public key of the digital twin system.
port	PortNumber	The created digital twins network port.
systemName	String	Name of the created digital twin system.

3.7 Primitives

As all messages are encoded using the JSON format [**?**], 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.

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 [**?**], 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 **create-digital-twin** SD document without a direct equivalent among the JSON types. Concretely, we define the **create-digital-twin** 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.

3.7.1 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).

3.7.2 alias PortNumber = Number

Decimal Number in the range of 0-65535.

3.7.3 alias List <A> = Array<A>

There is no difference.

4 Revision History

4.1 Amendments

No.	Date	Version	Subject of Amendments	Author
1	YYYY-MM-DD	4.6.1		Xxx Yyy

4.2 Quality Assurance

No.	Date	Version	Approved by
1	YYYY-MM-DD	4.6.1	Xxx Yyy