

Workgroup: Network Working Group  
Internet-Draft:  
draft-shahzad-scim-device-model-03  
Published: 13 March 2023  
Intended Status: Standards Track  
Expires: 14 September 2023  
Authors: M. Shahzad  
          North Carolina State University  
          H. Iqbal                          E. Lear  
          North Carolina State University  Cisco Systems

## Device Schema Extensions to the SCIM model

### Abstract

The initial core schema for SCIM (System for Cross Identity Management) was designed for provisioning users. This memo specifies schema extensions that enables provisioning of devices, using various underlying bootstrapping systems, such as Wifi EasyConnect, RFC 8366 vouchers, and BLE passcodes.

### Status of This Memo

This Internet-Draft is submitted in full conformance with the provisions of BCP 78 and BCP 79.

Internet-Drafts are working documents of the Internet Engineering Task Force (IETF). Note that other groups may also distribute working documents as Internet-Drafts. The list of current Internet-Drafts is at <https://datatracker.ietf.org/drafts/current/>.

Internet-Drafts are draft documents valid for a maximum of six months and may be updated, replaced, or obsoleted by other documents at any time. It is inappropriate to use Internet-Drafts as reference material or to cite them other than as "work in progress."

This Internet-Draft will expire on 14 September 2023.

### Copyright Notice

Copyright (c) 2023 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

## Table of Contents

- [1. Introduction](#)
  - [1.1. Why SCIM for devices?](#)
  - [1.2. Protocol Participants](#)
  - [1.3. Schema Description](#)
  - [1.4. Schema Representation](#)
  - [1.5. Terminology](#)
- [2. ResourceType Device](#)
  - [2.1. Common Attributes](#)
- [3. SCIM Core Device Schema](#)
  - [3.1. Singular Attributes](#)
- [4. Device Groups](#)
- [5. SCIM Device Extensions](#)
  - [5.1. BLE Extension](#)
    - [5.1.1. Singular Attributes](#)
    - [5.1.2. Multivalued Attributes](#)
    - [5.1.3. BLE Pairing Method Extensions](#)
  - [5.2. DPP EasyConnect Extension](#)
    - [5.2.1. Singular Attributes](#)
    - [5.2.2. Multivalued Attributes](#)
  - [5.3. Zigbee Extension](#)
    - [5.3.1. Singular Attribute](#)
    - [5.3.2. Multivalued Attribute](#)
  - [5.4. The Endpoint Applications Extension Schema](#)
    - [5.4.1. "deviceControl" Complex Attribute](#)
    - [5.4.2. "dataReceiver" Complex Attribute](#)
- [6. Schema JSON Representation](#)
  - [6.1. Device Resource Schema](#)
  - [6.2. Device Core Schema JSON](#)
  - [6.3. BLE Extension Schema JSON](#)
  - [6.4. DPP Extension Schema JSON](#)
  - [6.5. Zigbee Extension Schema JSON](#)
  - [6.6. EndpointApps Extension Schema JSON](#)
- [7. Security Considerations](#)
- [8. IANA Considerations](#)
- [9. Changes from Earlier Versions](#)
- [10. APPENDIX](#)
  - [10.1. Device Core Schema OpenAPI Representation](#)
  - [10.2. BLE Extension Schema OpenAPI Representation](#)
  - [10.3. DPP Extension Schema OpenAPI Representation](#)
  - [10.4. Zigbee Extension Schema OpenAPI Representation](#)
  - [10.5. EndpointApps Extension Schema OpenAPI Representation](#)
- [11. References](#)
  - [11.1. Normative References](#)
  - [11.2. Informative References](#)
- [Authors' Addresses](#)

### 1. Introduction

The Internet of Things presents a management challenge in many dimensions. One of them is the ability to onboard and manage large number of devices. There are many models for bootstrapping trust

between devices and network deployments. Indeed it is expected that different manufacturers will make use of different methods.

SCIM (System for Cross Identity Management) [[RFC7643](#)] [[RFC7644](#)] defines a protocol and a schema for provisioning of users. However, it can easily be extended to provision devices. The protocol and core schema were designed to permit just such extensions. Bulk operations are supported. This is good because often devices are procured in bulk.

### **1.1. Why SCIM for devices?**

Some might ask why SCIM is well suited for this purpose and not, for example, NETCONF or RESTCONF with YANG. After all, there are all sorts of existing models available. The answer is that the only information being passed about the device is neither state nor device configuration information, but only information necessary to bootstrap trust so that the device may establish connectivity.

### **1.2. Protocol Participants**

In the normal SCIM model, it was presumed that large federated deployments would be SCIM clients who provision and remove employees and contractors as they enter and depart those deployments, and federated services such as sales, payment, or conferencing services would be the servers.

In the device model, the roles are reversed, and may be somewhat more varied. A deployment network management system gateway (NMS gateway) plays the role of the server, receiving information about devices that are expected to be connected to its network. That server will apply appropriate local policies regarding whether/how the device should be connected.

The client may be one of a number of entities:

- \*A vendor who is authorized to add devices to a network as part of a sales transaction. This is similar to the sales integration sometimes envisioned by Bootstrapping Remote Key Infrastructure (BRSKI) [[RFC8995](#)].

- \*A client application that administrators or employees use to add, remove, or get information about devices. An example might be an tablet or phone app that scans Easyconnect QR codes.

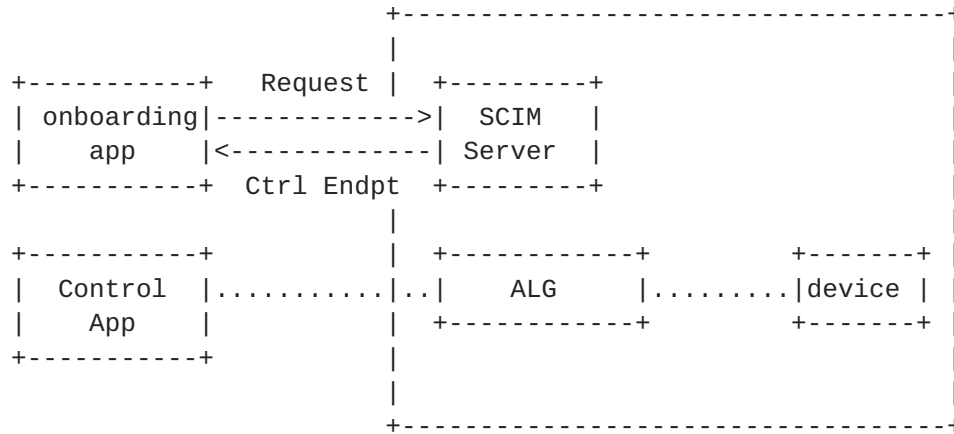


Figure 1: Basic Architecture

In [Figure 1](#), the onboarding app provides the device particulars. As part of the response, the SCIM server might provide additional information, especially in the case of non-IP devices, where an application-layer gateway may need to be used to communicate with the device. The control endpoint is one among a number of objects that may be returned.

### 1.3. Schema Description

RFC 7643 does not prescribe a language to describe a schema. We have chosen the JSON schema language [[I-D.bhutton-json-schema](#)] for this purpose. This implies that use of XML for this device extension is not supported.

Several additional schemas specify specific onboarding mechanisms, such as BLE and Wifi Easy Connect.

### 1.4. Schema Representation

Attributes defined in the device core schema and extensions comprise characteristics and SCIM datatypes defined in Sections 2.2 and 2.3 of the [[RFC7643](#)]. This RFC does not define new characteristics and datatypes for the SCIM attributes.

### 1.5. Terminology

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [[RFC2119](#)] [[RFC8174](#)] when, and only when, they appear in all capitals, as shown here.

## 2. ResourceType Device

This section defines a new resource type, 'Device'. The "ResourceType" schema specifies the metadata about a resource type

(see section 6 of [[RFC7643](#)]). The resource "Devices" comprises a core device schema and several extension schemas. The core schema provides a minimal resource representation, whereas extension schemas extend the core schema depending on the device's capability. The JSON schema for Device resource type is in [Section 6.1](#).

## 2.1. Common Attributes

The Device schema contains three common attributes as defined in the [[RFC7643](#)].

id

An id is a required and unique attribute of the device core schema (see section 3.1 of [[RFC7643](#)]).

externalID

An externalID is an optional attribute (see section 3.1 of [[RFC7643](#)]).

meta

Meta is a complex attribute and is required (see section 3.1 of [[RFC7643](#)]).

## 3. SCIM Core Device Schema

The core device schema provides the minimal representation of a resource "Device". It contains only those attributes that any device may need. Not all attributes are optional. The core schema for "Device" is identified using the schema URI:

"urn:ietf:params:scim:schemas:core:2.0:Device". The following attributes are defined in the device core schema.

### 3.1. Singular Attributes

deviceDisplayName

This attribute is of type "string" and provides a human-readable name for a device. It is intended to be displayed to end-users and should be suitable for that purpose. The attribute is not required, and is not case-sensitive. The attribute may be modified and should be returned by default. No uniqueness constraints are imposed on this attribute.

adminState

The "adminState" attribute is of type "boolean" and is a mutable attribute. If set to TRUE, the commands such as connect, disconnect, subscribe that control app sends to the controller for the devices will be processed by the controller. If set to FALSE, any command coming from the control app for the device will be rejected by the controller. This attribute is required and mutable. The attribute

should be returned by default and there is uniqueness constraint on the attribute.

mudUrl

The mudUrl attribute represents the URL to the MUD file associated with this device. This attribute is optional and mutable. The mudUrl value is case sensitive and not unique. When present, this attribute may be used as described in [RFC8520]. This attribute is case sensitive and returned by default.

Attribute	Multi Value	Req	Case Exact	Mutable	Return	Unique
deviceDisplayName	F	F	F	RW	Def	None
adminState	F	T	F	RW	Def	None
mudUrl	F	F	T	RW	Def	None

Figure 2: Characteristics of device schema attributes. (Req = Required, T = True, F = False, RW = ReadWrite, and Def = Default)

An example of a device SCIM object is as follows:

```
<CODE BEGINS>
{
  "schemas": ["urn:ietf:params:scim:schemas:core:2.0:Device"],
  "id": "e9e30dba-f08f-4109-8486-d5c6a3316111",
  "deviceDisplayName": "BLE Heart Monitor",
  "adminState": true,
  "meta": {
    "resourceType": "Device",
    "created": "2022-01-23T04:56:22Z",
    "lastModified": "2022-05-13T04:42:34Z",
    "version": "W\\\\"a330bc54f0671c9\\\"",
    "location": "https://example.com/v2/Device/e9e30dba-f08f-4109-
  }
}
<CODE ENDS>
```

The schema for the device is presented in JSON format in Section [Section 6.2](#), while the openAPI representation is provided in Section [Section 10.1](#).

#### 4. Device Groups

Device groups are created using the SCIM groups as defined in [RFC7643] Section 4.2.

## 5. SCIM Device Extensions

SCIM provides various extension schemas, their attributes, JSON representation, and example object. These schemas extend the core device schema based on the device's capability (communication stack). This RFC presents an additional hierarchical level by introducing extensions within an extension. See below for more details.

[ [ DISCUSS: Is this okay with the working group? ] ]

### 5.1. BLE Extension

This schema extends the device schema to represent the devices supporting BLE. The extension is identified using the following schema URI:

```
urn:ietf:params:scim:schemas:extension:ble:2.0:Device
```

The attributes are as follows:

#### 5.1.1. Singular Attributes

deviceMacAddress

A string value that represents a public MAC address assigned by the manufacturer. It is a unique 48-bit value. It is required, case insensitive, and it is mutable and returns as default. The regex pattern is the following:

```
^[0-9A-Fa-f]{2}(:[0-9A-Fa-f]{2}){5}
```

addressType

A boolean flag taken from the BLE core specification, 5.3. If FALSE, the device is using a public MAC address. If TRUE, the device uses a Random address resolved using IRK. This attribute is required, it is mutable, and returns by default.

irk

A string value, Identity resolving key, which is unique for every device. It is used to resolve the random address. It is required when addressType is TRUE. It is mutable and returns by default.

#### 5.1.2. Multivalued Attributes

versionSupport

A multivalued attribute that provides all the BLE versions supported by the device in the form of an array. For example, [4.1, 4.2, 5.0, 5.1, 5.2, 5.3]. It is required, mutable, and returns as default.

pairingMethods

An array of pairing methods associated with the BLE device. The pairing methods may require sub-attributes, such as key/password, for the device pairing process. To enable the scalability of pairing methods in the future, they are represented as extensions to incorporate various attributes that are part of the respective pairing process. Pairing method extensions are nested inside the BLE extension. It is required, case sensitive, mutable, and returned by default.

### 5.1.3. BLE Pairing Method Extensions

The details on pairing methods and their associated attributes are in section 2.3 of [BLE53]. This memo defines extensions for four pairing methods that are nested inside the BLE extension schema. Each extension contains the common attributes [Section 2.1](#). These extensions are as follows.

(i) pairingNull extension is identified using the following schema URI:

```
urn:ietf:params:scim:schemas:extension:pairingNull:2.0:Device
```

pairingNull does not have any attribute. It allows pairing for BLE devices that do not require a pairing method.

(ii) pairingJustWorks extension is identified using the following schema URI:

```
urn:ietf:params:scim:schemas:extension:pairingJustWorks:2.0:Device
```

Just works pairing method does not require a key to pair devices. For completeness, the key attribute is included and is set to 'null'. Key attribute is required, immutable, and return by default.

(iii) pairingPassKey extension is identified using the following schema URI:

```
urn:ietf:params:scim:schemas:extension:pairingPassKey:2.0:Device
```

The pass key pairing method requires a 6-digit key to pair devices. This extension has one singular integer attribute, "key", which is required, mutable and returned by default. The key pattern is as follows:

```
^[0-9]{6}$
```

(iv) pairingOOB extension is identified using the following schema URI:

```
urn:ietf:params:scim:schemas:extension:pairingOOB:2.0:Device
```

The out-of-band pairing method includes three singular attributes, i.e., key, randomNumber, and confirmationNumber.



key The key is string value, required and received from out-of-bond sources such as NFC. It is case sensitive, mutable, and returned by default.

randomNumber It represents a nonce added to the key. It is an integer value that is required attribute. It is mutable and returned by default.

confirmationNumber An integer which some solutions require in RESTful message exchange. It is not required. It is mutable and returned by default if it exists.

Attribute	Multi Value	Req	Case Exact	Mutable	Return	Unique
deviceMacAddress	F	T	F	RW	Def	Manuf
addressType	F	T	F	RW	Def	None
irk	F	F	F	RW	Def	Manuf
versionSupport	T	T	F	RW	Def	None
pairingMethods	T	T	T	RW	Def	None

Figure 3: Characteristics of BLE extension schema attributes. (Req = Required, T = True, F = False, RW = ReadWrite, Def = Default, and Manuf = Manufacturer).

An example of a device object with BLE extension is as follows:

```

<CODE BEGINS>
{
  "schemas": ["urn:ietf:params:scim:schemas:core:2.0:Device",
    "urn:ietf:params:scim:schemas:extension:ble:2.0:Device"],

  "id": "e9e30dba-f08f-4109-8486-d5c6a3316111",
  "deviceDisplayName": "BLE Heart Monitor",
  "adminState": true,
  "urn:ietf:params:scim:schemas:extension:ble:2.0:Device" : {
    "versionSupport": ["5.3"],
    "deviceMacAddress": "2C:54:91:88:C9:E2",
    "addressType": false,
    "pairingMethods": ["urn:ietf:params:scim:schemas:extension:pairingNu
      "urn:ietf:params:scim:schemas:extension:pairingJustWorks:2.0:Dev
      "urn:ietf:params:scim:schemas:extension:pairingPassKey:2.0:Devic
      "urn:ietf:params:scim:schemas:extension:pairingOOB:2.0:Device"],
    "urn:ietf:params:scim:schemas:extension:pairingNull:2.0:Device": nul
    "urn:ietf:params:scim:schemas:extension:pairingJustWorks:2.0:Device"
      "key": null
    },
    "urn:ietf:params:scim:schemas:extension:pairingPassKey:2.0:Device" :
      "key": 123456
    },
    "urn:ietf:params:scim:schemas:extension:pairingOOB:2.0:Device": {
      "key": "TheKeyvalueRetrievedFromOOB",
      "randNumber": 238796813516896
    }
  },
  "meta": {
    "resourceType": "Device",
    "created": "2022-01-23T04:56:22Z",
    "lastModified": "2022-05-13T04:42:34Z",
    "version": "W\\"a330bc54f0671c9\"",
    "location": "https://example.com/v2/Device/e9e30dba-f08f-4109-8486-d
  }
}
<CODE ENDS>

```

The schema for the BLE extension is presented in JSON format in Section [Section 6.3](#), while the openAPI representation is provided in Section [Section 10.2](#).

## 5.2. DPP EasyConnect Extension

A schema that extends the device schema to enable WiFi EasyConnect (otherwise known as Device Provisioning Protocol). The extension is identified using the following schema URI:

```
urn:ietf:params:scim:schemas:extension:dpp:2.0:Device
```

The attributes in this extension are adopted from [[DPP2](#)]. The attributes are as follows:

### 5.2.1. Singular Attributes

dppVersion

An integer that represents the version of DPP the device supports. This attribute is required, case insensitive, mutable, and returned by default.

bootstrapKey

A string value representing Elliptic-Curve Diffie-Hellman (ECDH) public key. The base64 encoded lengths for P-256, P-384, and P-521 are 80, 96, and 120 characters. This attribute is required, case-sensitive, mutable, and returned by default.

deviceMacAddress

The manufacturer assigns the MAC address stored as string. It is a unique 48-bit value. This attribute is optional, case insensitive, mutable, and returned by default. The regex pattern is as follows:

```
^[0-9A-Fa-f]{2}(:[0-9A-Fa-f]{2}){5}.
```

serialNumber

An alphanumeric serial number, stored as string, may also be passed as bootstrapping information. This attribute is optional, case insensitive, mutable, and returned by default.

### 5.2.2. Multivalued Attributes

bootstrappingMethod

It is the array of strings of all the bootstrapping methods available on the enrollee device. For example, [QR, NFC]. This attribute is optional, case insensitive, mutable, and returned by default.

classChannel

This attribute is an array of strings of global operating class and channel shared as bootstrapping information. It is formatted as class/channel. For example, ['81/1', '115/36']. This attribute is optional, case insensitive, mutable, and returned by default.

Attribute	Multi Value	Req	Case Exact	Mutable	Return	Unique
dppVersion	F	T	F	RW	Def	None
bootstrapKey	F	T	T	RW	Def	None
deviceMacAddress	F	F	F	RW	Def	Manuf
serialNumber	F	F	F	RW	Def	None
bootstrappingMethod	T	F	F	RW	Def	None
classChannel	T	F	F	RW	Def	None

Figure 4: Characteristics of DPP extension schema attributes. (Req = Required, T = True, F = False, RW = ReadWrite, Def = Default, and Manuf = Manufacturer).

An example of a device object with DPP extension is below:

```
<CODE BEGINS>
{
  "schemas": ["urn:ietf:params:scim:schemas:core:2.0:Device",
             "urn:ietf:params:scim:schemas:extension:dpp:2.0:Device"

  "id": "e9e30dba-f08f-4109-8486-d5c6a3316111",
  "displayName": "WiFi Heart Monitor",
  "adminState": true,
  "urn:ietf:params:scim:schemas:extension:dpp:2.0:Device" : {
    "versionSupport": ["802.11ax"],
    "bootstrappingMethod": ["QR"],
    "bootstrapKey": "MDkwEwYHKoZIZj0CAQYIKoZIZj0DAQcDIgADURz",
    "deviceMacAddress": "2C:54:91:88:C9:F2",
    "classChannel": ["81/1", "115/36"],
    "serialNumber": "4774LH2b4044"
  },

  "meta": {
    "resourceType": "Device",
    "created": "2022-01-23T04:56:22Z",
    "lastModified": "2022-05-13T04:42:34Z",
    "version": "w\\"a330bc54f0671c9\"",
    "location": "https://example.com/v2/Device/e9e30dba-f08f-4109-
  }
}
<CODE ENDS>
```

The schema for the DPP extension is presented in JSON format in Section [Section 6.4](#), while the openAPI representation is provided in Section [Section 10.3](#).

### 5.3. Zigbee Extension

A schema that extends the device schema to enable the provisioning of Zigbee devices. The extension is identified using the following schema URI:

```
urn:ietf:params:scim:schemas:extension:zigbee:2.0:Device
```

It has one singular attribute and one multivalued attribute. The attributes are as follows:

#### 5.3.1. Singular Attribute

deviceEui64Address

An EUI-64 (Extended Unique Identifier) device address stored as string. This attribute is required, case insensitive, mutable, and returned by default. The regex pattern is as follows:

```
^[0-9A-Fa-f]{16}$
```

#### 5.3.2. Multivalued Attribute

versionSupport

An array of strings of all the Zigbee versions supported by the device. For example, [3.0]. This attribute is required, case insensitive, mutable, and returned by default.

Attribute	Multi Value	Req	Case Exact	Mutable	Return	Unique
deviceEui64Address	F	T	F	RW	Def	None
versionSupport	T	T	F	RW	Def	None

Figure 5: Characteristics of Zigbee extension schema attributes. (Req = Required, T = True, F = False, RW = ReadWrite, and Def = Default)

An example of a device object with Zigbee extension is shown below:

```

<CODE BEGINS>
{
  "schemas": ["urn:ietf:params:scim:schemas:core:2.0:Device",
    "urn:ietf:params:scim:schemas:extension:zigbee:2.0:Device"],

  "id": "e9e30dba-f08f-4109-8486-d5c6a3316111",
  "displayName": "Zigbee Heart Monitor",
  "adminState": true,
  "urn:ietf:params:scim:schemas:extension:zigbee:2.0:Device" : {
    "versionSupport": ["3.0"],
    "deviceEui64Address": "50325FFFFEE76728"
  },

  "meta": {
    "resourceType": "Device",
    "created": "2022-01-23T04:56:22Z",
    "lastModified": "2022-05-13T04:42:34Z",
    "version": "W\\"a330bc54f0671c9\"",
    "location": "https://example.com/v2/Device/e9e30dba-f08f-4109-8486-d
  }
}
<CODE ENDS>

```

The schema for the Zigbee extension is presented in JSON format in Section [Section 6.5](#), while the openAPI representation is provided in Section [Section 10.4](#).

#### 5.4. The Endpoint Applications Extension Schema

Sometimes non-IP devices such as those using BLE or Zigbee require an application gateway interface to manage them. SCIM clients MUST NOT specify this to describe native IP-based devices.

The device schema is a base schema upon which specific onboarding technology schemas are built. This extension provides two complex attributes deviceControl, and dataReceiver to represent two types of partner endpoint applications, device control, and data receiver applications, respectively. The extension is identified using the following schema URI:

```
urn:ietf:params:scim:schemas:extension:endpointApps:2.0:Device
```

##### 5.4.1. "deviceControl" Complex Attribute

A complex attribute representing the application responsible for controlling the device. Device control applications provide access control for the device. It determines which other applications can access the data from the device. Depending on the use case, there can be more than one device control application. deviceControl has two sub-attributes: (i) a multivalued client-tokens attribute and (ii) a singular attribute called enterpriseEndpoint.

client-tokens

This attribute contains a list of tokens that the client will use to authenticate itself. Each token may be a string up to 500 characters in length. This attribute is a multivalued string which is required, case-sensitive, mutable, and returned by default.

[[ We probably need something better here. ]]

enterpriseEndpoint

Device control apps use the URL of the enterprise endpoint to reach the network gateway. When the enterprise receives the SCIM object from the onboarding app, it adds this attribute to it and sends it back as a response to the onboarding app. This attribute is required, case-sensitive, mutable, and returned by default.

#### 5.4.2. "dataReceiver" Complex Attribute

A complex attribute representing the application(s) that receive data from the device. Depending on the use case, there can be more than one data receiver application. "dataReceiver" has two sub-attributes: (i) a multi-valued attribute called client-tokens, and (ii) a singular attribute called enterpriseEndpoint.

client-tokens

This attribute contains a list of tokens that the client will use to authenticate itself. Each token may be a string up to 500 characters in length. This attribute is a multivalued string which is required, case-sensitive, mutable, and returned by default.

[[ We probably need something better here. ]]

enterpriseEndpoint

Data receiver apps use the URL of the enterprise endpoint to reach the network gateway. When the enterprise receives the SCIM object from the onboarding app, it adds this attribute to it and sends it back as a response to the onboarding app. This attribute is required, case-sensitive, mutable, and returned by default.

Attribute	Multi Value	Req	Case Exact	Mutable	Return	Unique
client-tokens	T	T	T	RW	Def	None
enterpriseEndpoint	F	T	T	RW	Def	None
deviceControl	F	T	F	RW	Def	Manuf
dataReceiver	F	T	F	RW	Def	None

Figure 6: Characteristics of EndpointApps extension schema attributes.  
(Req = Required, T = True, F = False, RW = ReadWrite, Def = Default,  
and Manuf = Manufacturer).

An example of a device object with endpointApps extension is below:



<CODE BEGINS>

```
{
  "schemas": ["urn:ietf:params:scim:schemas:core:2.0:Device",
    "urn:ietf:params:scim:schemas:extension:ble:2.0:Device",
    "urn:ietf:params:scim:schemas:extension:endpointapps:2.0:Device"],

  "id": "e9e30dba-f08f-4109-8486-d5c6a3316111",
  "displayName": "BLE Heart Monitor",
  "adminState": true,
  "urn:ietf:params:scim:schemas:extension:ble:2.0:Device" :
  {
    "versionSupport": ["5.3"],
    "deviceMacAddress": "2C:54:91:88:C9:E2",
    "addressType": false,
    "pairingMethods": [
      "urn:ietf:params:scim:schemas:extension:pairingNull:2.0:Device",
      "urn:ietf:params:scim:schemas:extension:pairingJustWorks:2.0:Dev
      "urn:ietf:params:scim:schemas:extension:pairingPassKey:2.0:Devic
      "urn:ietf:params:scim:schemas:extension:pairingOOB:2.0:Device"],
    "urn:ietf:params:scim:schemas:extension:pairingNull:2.0:Device": nul
    "urn:ietf:params:scim:schemas:extension:pairingJustWorks:2.0:Device"
      "key": null
    },
    "urn:ietf:params:scim:schemas:extension:pairingPassKey:2.0:Device" :
      "key": 123456
    },
    "urn:ietf:params:scim:schemas:extension:pairingOOB:2.0:Device": {
      "key": "TheKeyvalueRetrievedFromOOB",
      "randNumber": 238796813516896
    }
  },
  "urn:ietf:params:scim:schemas:extension:endpointapps:2.0:Device": {

    "deviceControl": {
      "client-tokens": [
        "1293847109832471203984789383XXX"
      ],
      "enterpriseEndpoint": "https://enterprise.com/device_control_endpoi
    },

    "dataReceiver": {
      "client-tokens":[
        "dlku23234sdui232393"
      ],
      "enterpriseEndpoint": "https://enterprise.com/data_receiver_endpoint
    }
  },

  "meta": {
    "resourceType": "Device",
    "created": "2022-01-23T04:56:22Z",
    "lastModified": "2022-05-13T04:42:34Z",
```

```
"version": "W\\\\"a330bc54f0671c9\\",  
"location": "https://example.com/v2/Device/e9e30dba-f08f-4109-8486-d  
}  
}  
<CODE ENDS>
```

The schema for the endpointApps extension along with BLE extension is presented in JSON format in [Section 6.6](#), while the openAPI representation is provided in [Section 10.5](#).

## 6. Schema JSON Representation

## 6.1. Device Resource Schema

<CODE BEGINS>

```
[
  {
    "schemas": ["urn:ietf:params:scim:schemas:core:2.0:ResourceType"],
    "id": "Device",
    "name": "Device",
    "endpoint": "/Device",
    "description": "Device Account",
    "schema": "urn:ietf:params:scim:schemas:core:2.0:Device",
    "schemaExtensions": [
      {
        "schema": "urn:ietf:params:scim:schemas:extension:ble:2.0:Device"
        "required": false
      },
      {
        "schema": "urn:ietf:params:scim:schemas:extension:dpp:2.0:Device"
        "required": false
      },
      {
        "schema": "urn:ietf:params:scim:schemas:extension:zigbee:2.0:Dev"
        "required": false
      },
      {
        "schema": "urn:ietf:params:scim:schemas:extension:endpointApps:2"
        "required": false
      },
      {
        "schema": "urn:ietf:params:scim:schemas:extension:pairingNull:2."
        "required": false
      },
      {
        "schema": "urn:ietf:params:scim:schemas:extension:pairingJustWor"
        "required": false
      },
      {
        "schema": "urn:ietf:params:scim:schemas:extension:pairingPassKey"
        "required": false
      },
      {
        "schema": "urn:ietf:params:scim:schemas:extension:pairing00B:2.0"
        "required": false
      }
    ],
    "meta": {
      "location": "https://example.com/v2/ResourceTypes/Device",
      "resourceType": "ResourceType"
    }
  }
]
```

]

<CODE ENDS>

## 6.2. Device Core Schema JSON

<CODE BEGINS>

```
{
  "id": "urn:ietf:params:scim:schemas:core:2.0:Device",
  "name": "Device",
  "description": "Device account",
  "attributes" : [
    {
      "name": "deviceDisplayName",
      "type": "string",
      "description": "Human readable name of the device, suitable for
        displaying to end-users. For example, 'BLE Heart Monitor'
        etc.",
      "multivalues": false,
      "required": false,
      "caseExact": false,
      "mutability": "readWrite",
      "returned": "default",
      "uniqueness": "none"
    },
    {
      "name": "adminState",
      "type": "boolean",
      "description": "A mutable boolean value indicating the device
        administrative status. If set TRUE, the commands (such as
        connect, disconnect, subscribe) that control app sends to the
        controller for the devices will be processed by the
        controller. If set FALSE, any command coming from the control
        app for the device will be rejected by the controller.",
      "multivalues": false,
      "required": true,
      "caseExact": false,
      "mutability": "readWrite",
      "returned": "default",
      "uniqueness": "none"
    },
    {
      "name": "mudUrl",
      "type": "reference",
      "description": "A URL to MUD file of the device (RFC 8520).",
      "multivalues": false,
      "required": false,
      "caseExact": true,
      "mutability": "readWrite",
      "returned": "default",
      "uniqueness": "none"
    }
  ],
  "meta" : {
    "resourceType" : "Schema",
    "location" :
      "/v2/Schemas/urn:ietf:params:scim:schemas:core:2.0:Device"
  }
}
```



<CODE ENDS>

### 6.3. BLE Extension Schema JSON

<CODE BEGINS>

```
[
  {
    "id": "urn:ietf:params:scim:schemas:extension:ble:2.0:Device",
    "name": "bleExtension",
    "description": "Ble extension for device account",
    "attributes" : [
      {
        "name": "versionSupport",
        "type": "string",
        "description": "Provides a list of all the BLE versions
          supported by the device. For example, [4.1, 4.2, 5.0, 5.1,
          5.2, 5.3].",
        "multivalues": true,
        "required": true,
        "caseExact": false,
        "mutability": "readWrite",
        "returned": "default",
        "uniqueness": "none"
      },
      {
        "name": "deviceMacAddress",
        "type": "string",
        "description": "It is the public MAC address assigned by the
          manufacturer. It is unique 48 bit value. The regex pattern
          is ^[0-9A-Fa-f]{2}(:[0-9A-Fa-f]{2}){5}.",
        "multivalues": false,
        "required": true,
        "caseExact": false,
        "mutability": "readWrite",
        "returned": "default",
        "uniqueness": "Manufacturer"
      },
      {
        "name": "addressType",
        "type": "boolean",
        "description": "AddressType flag is taken from the BLE core
          specifications 5.3. If FALSE, the device is using public MAC
          address. If TRUE, device is using Random address which is
          resolved using IRK.",
        "multivalues": false,
        "required": true,
        "caseExact": false,
        "mutability": "readWrite",
        "returned": "default",
        "uniqueness": "none"
      },
      {
        "name": "irk",
        "type": "string",
        "description": "Identity resolving key, which is unique for
          every device. It is used to resolve random address.",
        "multivalues": false,
```

```

    "required": false,
    "caseExact": false,
    "mutability": "readWrite",
    "returned": "default",
    "uniqueness": "Manufacturer"
  },
  {
    "name": "pairingMethods",
    "type": "string",
    "description": "List of pairing methods associated with the
      ble device, stored as schema URI.",
    "multivalues": true,
    "required": true,
    "caseExact": true,
    "mutability": "readWrite",
    "returned": "default",
    "uniqueness": "none"
  }
],
"meta" : {
  "resourceType" : "Schema",
  "location" : "/v2/Schemas/urn:ietf:params:scim:
    schemas:extension:ble:2.0:Device"
}
},
{
  "id": "urn:ietf:params:scim:schemas:extension:pairingNull:2.0:Device",
  "name": "nullPairing",
  "description": "Null pairing method for ble. It is included for
    the devices that do not have a pairing method.",
  "meta" : {
    "resourceType" : "Schema",
    "location" : "/v2/Schemas/urn:ietf:params:scim:schemas:
      extension:pairingNull:2.0:Device"
  }
},
{
  "id": "urn:ietf:params:scim:schemas:extension:
    pairingJustWorks:2.0:Device",
  "name": "pairingJustWorks",
  "description": "Just works pairing method for ble.",
  "attributes" : [
    {
      "name": "key",
      "type": "integer",
      "description": "Just works does not have any key value. For
        completeness, it is added with a key value 'null'.",
      "multivalues": false,
      "required": true,
      "caseExact": false,
      "mutability": "immutable",
      "returned": "default",
      "uniqueness": "none"
    }
  ]
}

```

```

    }
  ],
  "meta" : {
    "resourceType" : "Schema",
    "location" : "/v2/Schemas/urn:ietf:params:scim:schemas:extension:pairingJustWorks:2.0:Device"
  }
},
{
  "id": "urn:ietf:params:scim:schemas:
    extension:pairingPassKey:2.0:Device",
  "name": "pairingPassKey",
  "description": "Pass key pairing method for ble.",
  "attributes" : [
    {
      "name": "key",
      "type": "integer",
      "description": "A six digit passkey for ble device. The
        pattern of key is ^[0-9]{6}$.",
      "multivalued": false,
      "required": true,
      "caseExact": false,
      "mutability": "readWrite",
      "returned": "default",
      "uniqueness": "none"
    }
  ],
  "meta" : {
    "resourceType" : "Schema",
    "location" : "/v2/Schemas/urn:ietf:params:scim:schemas:
      extension:pairingPassKey:2.0:Device"
  }
},
{
  "id": "urn:ietf:params:scim:schemas:extension:pairingOOB:2.0:Device"
  "name": "pairingOOB",
  "description": "Pass key pairing method for ble.",
  "attributes" : [
    {
      "name": "key",
      "type": "string",
      "description": "A key value retrieved from out of band source
        such as NFC.",
      "multivalued": false,
      "required": true,
      "caseExact": true,
      "mutability": "readWrite",
      "returned": "default",
      "uniqueness": "none"
    },
    {
      "name": "randomNumber",
      "type": "integer",

```

```
    "description": "Nonce added to the key.",
    "multivalued": false,
    "required": true,
    "caseExact": false,
    "mutability": "readWrite",
    "returned": "default",
    "uniqueness": "none"
  },
  {
    "name": "confirmationNumber",
    "type": "integer",
    "description": "Some solutions require confirmation number in
      RESTful message exchange.",
    "multivalued": false,
    "required": false,
    "caseExact": false,
    "mutability": "readWrite",
    "returned": "default",
    "uniqueness": "none"
  }
],
"meta" : {
  "resourceType" : "Schema",
  "location" : "/v2/Schemas/urn:ietf:params:scim:schemas:
    extension:pairing00B:2.0:Device"
}
}
]
```

<CODE ENDS>

#### 6.4. DPP Extension Schema JSON

<CODE BEGINS>

```
{
  "id": "urn:ietf:params:scim:schemas:extension:dpp:2.0:Device",
  "name": "dppExtension",
  "description": "Device extension schema for DPP",
  "attributes" : [
    {
      "name": "dppVersion",
      "type": "integer",
      "description": "Version of DPP this device supports.",
      "multivalues": false,
      "required": true,
      "caseExact": false,
      "mutability": "readWrite",
      "returned": "default",
      "uniqueness": "none"
    },
    {
      "name": "bootstrappingMethod",
      "type": "string",
      "description": "The list of all the bootstrapping methods
        available on the enrollee device. For example, [QR, NFC].",
      "multivalues": true,
      "required": false,
      "caseExact": false,
      "mutability": "readWrite",
      "returned": "default",
      "uniqueness": "none"
    },
    {
      "name": "bootstrapKey",
      "type": "string",
      "description": "This key is Elliptic-Curve Diffie-Hellman
        (ECDH) public key. The base64 encoded length for P-256,
        P-384, and P-521 is 80, 96, and 120 characters.",
      "multivalues": false,
      "required": true,
      "caseExact": true,
      "mutability": "readWrite",
      "returned": "default",
      "uniqueness": "none"
    },
    {
      "name": "deviceMacAddress",
      "type": "string",
      "description": "The MAC address assigned by the
        manufacturer. It is unique 48 bit value. The regex pattern is
        ^[0-9A-Fa-f]{2}(:[0-9A-Fa-f]{2}){5}.",
      "multivalues": false,
      "required": false,
      "caseExact": false,
      "mutability": "readWrite",
      "returned": "default",
    }
  ]
}
```



```

    "uniqueness": "Manufacturer"
  },
  {
    "name": "classChannel",
    "type": "string",
    "description": "A list of global operating class and channel
      shared as bootstrapping information. It is formatted as
      class/channel. For example, '81/1', '115/36'.",
    "multivalues": true,
    "required": false,
    "caseExact": false,
    "mutability": "readWrite",
    "returned": "default",
    "uniqueness": "none"
  },
  {
    "name": "serialNumber",
    "type": "string",
    "description": "An alphanumeric serial number that may also be
      passed as bootstrapping information.",
    "multivalues": false,
    "required": false,
    "caseExact": false,
    "mutability": "readWrite",
    "returned": "default",
    "uniqueness": "none"
  }
],
"meta" : {
  "resourceType" : "Schema",
  "location" : "/v2/Schemas/urn:ietf:params:
    scim:schemas:extension:dpp:2.0:Device"
}
}

```

<CODE ENDS>

## 6.5. Zigbee Extension Schema JSON

<CODE BEGINS>

```
{
  "id": "urn:ietf:params:scim:schemas:extension:zigbee:2.0:Device",
  "name": "zigbeeExtension",
  "description": "Device extension schema for zigbee.",
  "attributes" : [
    {
      "name": "versionSupport",
      "type": "string",
      "description": "Provides a list of all the zigbee versions
        supported by the device. For example, [3.0].",
      "multivalued": true,
      "required": true,
      "caseExact": false,
      "mutability": "readWrite",
      "returned": "default",
      "uniqueness": "none"
    },
    {
      "name": "deviceEui64Address",
      "type": "string",
      "description": "The EUI-64 (Extended Unique Identifier) device
        address. The regex pattern is ^[0-9A-Fa-f]{16}$.",
      "multivalued": false,
      "required": true,
      "caseExact": false,
      "mutability": "readWrite",
      "returned": "default",
      "uniqueness": "none"
    }
  ],
  "meta" : {
    "resourceType" : "Schema",
    "location" :
      "/v2/Schemas/urn:ietf:params:scim:schemas:
        extension:zigbee:2.0:Device"
  }
}
```

<CODE ENDS>

## 6.6. EndpointApps Extension Schema JSON

<CODE BEGINS>

```
{
  "id": "urn:ietf:params:scim:schemas:extension:endpointApps:2.0:
    Device",
  "name": "endpointApps",
  "description": "Extension for partner endpoint applications that
    can onboard, control, and communicate with the device.",
  "attributes" : [
    {
      "name": "deviceControl",
      "type": "complex",
      "description": "Device control application attributes.",
      "multivalued": false,
      "required": true,
      "caseExact": false,
      "mutability": "readWrite",
      "returned": "default",
      "uniqueness": "none",
      "subAttributes" : [
        {
          "name": "client-tokens",
          "type": "string",
          "description": "This attribute contains a list of tokens
            that the client will used to authenticate itself. Each
            token may be a string up to 500 characters in length.",
          "multivalued": true,
          "required": true,
          "caseExact": true,
          "mutability": "readWrite",
          "returned": "default",
          "uniqueness": "none"
        },
        {
          "name": "enterpriseEndpoint",
          "type": "reference",
          "description": "The URL of the enterprise endpoint which
            device control apps uses to reach enterprise network
            gateway. When enterprise receives the SCIM object from
            onboarding app, it adds this attribute to the object and
            sends back the object as response back to the onboarding
            app.",
          "multivalued": false,
          "required": true,
          "caseExact": true,
          "mutability": "readWrite",
          "returned": "default",
          "uniqueness": "none"
        }
      ]
    },
    {
      "name": "dataReceiver",
      "type": "complex",
```

```

"description": "Data receiver application attributes.",
"multivalues": false,
"required": true,
"caseExact": false,
"mutability": "readWrite",
"returned": "default",
"uniqueness": "none",
"subAttributes" : [
  {
    "name": "client-tokens",
    "type": "string",
    "description": "This attribute contains a list of tokens
that the client will used to authenticate itself. Each
token may be a string up to 500 characters in length.",
    "multivalues": true,
    "required": true,
    "caseExact": true,
    "mutability": "readWrite",
    "returned": "default",
    "uniqueness": "none"
  },
  {
    "name": "enterpriseEndpoint",
    "type": "reference",
    "description": "The URL of the enterprise endpoint which
device control apps uses to reach enterprise network
gateway. When enterprise receives the SCIM object from
onboarding app, it adds this attribute to the object and
sends back the object as response back to the onboarding
app.",
    "multivalues": false,
    "required": true,
    "caseExact": true,
    "mutability": "readWrite",
    "returned": "default",
    "uniqueness": "none"
  }
]
}
],
"meta" : {
  "resourceType" : "Schema",
  "location" :
"/v2/Schemas/urn:ietf:params:scim:schemas:
extension:endpointApps:2.0:Device"
}
}

```

<CODE ENDS>

The following sections provide representations of schemas for both SCIM resources and service provider schemas. Note that the JSON representation has been modified for readability and to fit the specification format.

#### 8.7.1. Resource Schema Representation

The following is intended as an example of the SCIM schema representation in JSON format for SCIM resources. Where permitted, individual values and schema MAY change. This example includes schema representations for "User", "Group", and "EnterpriseUser"; other schema representations are possible.

### 7. Security Considerations

Because provisioning operations are sensitive, each client must be appropriately authenticated. Certain objects may be read-only or not visible based on who is connected.

[ More to be added here. ]

### 8. IANA Considerations

TBD

### 9. Changes from Earlier Versions

Draft -01:

\*Doh! We forgot the core device schema!

Draft -00:

\*Initial revision

### 10. APPENDIX

#### 10.1. Device Core Schema OpenAPI Representation

OpenAPI representation of device core schema is as follows:

```
<CODE BEGINS>
openapi: 3.0.3
info:
  title: Device Schema
  version: 1.0.0
  description: API for device core schema

components:
  schemas:
    Device:
      title: Device
      description: Device account
      type: object
      properties:
        deviceDisplayName:
          type: string
          description: "Human readable name of the device, suitable
            for displaying to end-users. For example,
            'BLE Heart Monitor' etc."
          nullable: true
          readOnly: false
          writeOnly: false
        adminState:
          type: boolean
          description: A mutable boolean value indicating the device
            administrative status. If set TRUE, the
            commands (such as connect, disconnect,
            subscribe) that control app sends to the
            controller for the devices will be processed
            by the controller. If set FALSE, any command
            coming from the control app for the device
            will be rejected by the controller.
          nullable: false
          readOnly: false
          writeOnly: false
        mudUrl:
          type: string
          format: uri
          description: A URL to MUD file of the device (RFC 8520). It
            is added for future use. Current usage is not
            defined yet.
          nullable: true
          readOnly: false
          writeOnly: false
      required:
        - adminState
      additionalProperties: false
      allOf:
        - $ref: '#/components/schemas/CommonAttributes'
    CommonAttributes:
      type: object
      properties:
        id:
```

```
type: string
format: uri
description: The unique identifier for a resource.
nullable: false
readOnly: true
writeOnly: false
externalId:
  type: string
  description: An identifier for the resource that is defined
    by the provisioning client.
  nullable: true
  readOnly: false
  writeOnly: false
meta:
  type: object
  properties:
    resourceType:
      type: string
      description: The name of the resource type of the
        resource.
      nullable: false
      readOnly: true
      writeOnly: false
    location:
      type: string
      format: uri
      description: The URI of the resource being returned.
      nullable: false
      readOnly: false
      writeOnly: false
    created:
      type: string
      format: date-time
      description: The date and time the resource was added
        to the service provider.
      nullable: false
      readOnly: true
      writeOnly: false
    lastModified:
      type: string
      format: date-time
      description: The most recent date and time that the
        details of this resource were updated at
        the service provider.
      nullable: false
      readOnly: true
      writeOnly: false
    version:
      type: string
      description: The version of the resource.
      nullable: true
      readOnly: false
      writeOnly: false
```



```

        additionalProperties: false

paths:
  /devices:
    post:
      summary: Create a new device
      description: Creates a new device.
      operationId: createDevice
      requestBody:
        description: The device to create
        required: true
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/Device'
      responses:
        '201':
          description: Created
          headers:
            Location:
              schema:
                type: string
                format: uri
              description: The URI of the newly created device.
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/Device'

  /devices/{deviceId}:
    get:
      summary: Get a device by ID
      description: Returns a single device by its ID.
      operationId: getDeviceById
      parameters:
        - name: deviceId
          in: path
          description: The ID of the device to return.
          required: true
          schema:
            type: string
      responses:
        '200':
          description: OK
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/Device'

```

<CODE ENDS>

## **10.2. BLE Extension Schema OpenAPI Representation**

OpenAPI representation of BLE extension schema is as follows:

```
<CODE BEGINS>
openapi: 3.0.1
info:
  version: 1.0.0
  title: DPP Extension Schema
  description: API for DPP device extension schema

components:
  schemas:
    BleDeviceExtension:
      type: object
      properties:
        versionSupport:
          type: array
          items:
            type: string
          description: Provides a list of all the BLE versions
            supported by the device. For example,
            [4.1, 4.2, 5.0, 5.1, 5.2, 5.3].
          nullable: false
          readOnly: false
          writeOnly: false
        deviceMacAddress:
          type: string
          description: It is the public MAC address assigned by the
            manufacturer. It is unique 48 bit value. The
            regex pattern is
            ^[0-9A-Fa-f]{2}(:[0-9A-Fa-f]{2}){5}.
          nullable: false
          readOnly: false
          writeOnly: false

        addressType:
          type: boolean
          description: AddressType flag is taken from the BLE core
            specifications 5.3. If FALSE, the device is
            using public MAC address. If TRUE, device is
            using Random address which is resolved using
            the IRK.
          nullable: false
          readOnly: false
          writeOnly: false
        irk:
          type: string
          description: Identity resolving key, which is unique for
            every device. It is used to resolve random
            address.
          nullable: true
          readOnly: false
          writeOnly: false
      pairingMethods:
        type: array
        items:
```

```
    type: string
    description: List of pairing methods associated with the
                 ble device, stored as schema URI.
    nullable: true
    readOnly: false
    writeOnly: false
required:
- versionSupport
- deviceMacAddress
- AddressType
- pairingMethods
additionalProperties: false
```

NullPairing:

```
type: object
properties:
  id:
    type: string
    description: The id of the null pairing schema.
    nullable: false
    readOnly: true
    writeOnly: false
```

PairingJustWorks:

```
type: object
description: Just works pairing method for ble
properties:
  key:
    type: integer
    description: Just works does not have any key value. For
                 completeness, it is added with a key value
                 'null'.
    nullable: false
    readOnly: false
    writeOnly: false
required:
- key
```

PairingPassKey:

```
type: object
description: Pass key pairing method for ble
properties:
  key:
    type: integer
    description: A six digit passkey for ble device.
                 The pattern of key is  $^{[0-9]\{6\}}\$$ .
    nullable: false
    readOnly: false
    writeOnly: false
required:
- key
```

PairingOOB:

```
type: object
description: Out-of-band pairing method for BLE
properties:
  key:
    type: string
    description: The OOB key value for ble device.
    nullable: false
    readOnly: false
    writeOnly: false
  randomNumber:
    type: integer
    description: Nonce added to the key
    nullable: false
    readOnly: false
    writeOnly: false
  confirmationNumber:
    type: integer
    description: Some solutions require a confirmation number
      in the RESTful message exchange.
    nullable: true
    readOnly: false
    writeOnly: false
required:
  - key
  - randomNumber
```

paths:

/devices:

post:

```
summary: Create a new device
operationId: createDevice
requestBody:
  required: true
  content:
    application/json:
      schema:
        $ref: '#/components/schemas/BleDeviceExtension'
      anyof:
        - $ref: '#/components/schemas/NullPairing'
        - $ref: '#/components/schemas/PairingJustWorks'
        - $ref: '#/components/schemas/PairingPassKey'
        - $ref: '#/components/schemas/PairingOOB'
```

responses:

```
'201':
  description: Created
  content:
    application/json:
      schema:
        $ref: '#/components/schemas/BleDeviceExtension'
      anyof:
        - $ref: '#/components/schemas/NullPairing'
        - $ref: '#/components/schemas/PairingJustWorks'
        - $ref: '#/components/schemas/PairingPassKey'
```

```
        - $ref: '#/components/schemas/PairingOOB'
/devices/{deviceId}:
  get:
    summary: Get a Dpp device by ID
    description: Returns a single Dpp device by its ID.
    operationId: getDeviceById
    parameters:
      - name: id
        in: path
        description: The ID of the device to return.
        required: true
        schema:
          type: string
    responses:
      '200':
        description: OK
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/BleDeviceExtension'
            anyof:
              - $ref: '#/components/schemas/NullPairing'
              - $ref: '#/components/schemas/PairingJustWorks'
              - $ref: '#/components/schemas/PairingPassKey'
              - $ref: '#/components/schemas/PairingOOB'
```

<CODE ENDS>

### **10.3. DPP Extension Schema OpenAPI Representation**

OpenAPI representation of DPP extension schema is as follows:

<CODE BEGINS>

openapi: 3.0.1

info:

version: 1.0.0

title: DPP Extension Schema

description: API for DPP device extension schema

components:

schemas:

DppDeviceExtension:

type: object

properties:

dppVersion:

type: integer

description: Version of DPP this device supports.

nullable: false

readOnly: false

writeOnly: false

bootstrappingMethod:

type: array

items:

type: string

description: The list of all the bootstrapping methods available on the enrollee device. For example, [QR, NFC].

nullable: true

readOnly: false

writeOnly: false

bootstrapKey:

type: string

description: This key is Elliptic-Curve Diffie-Hellman (ECDH) public key. The base64 encoded length for P-256, P-384, and P-521 is 80, 96, and 120 characters.

nullable: false

readOnly: false

writeOnly: false

deviceMacAddress:

type: string

description: The MAC address assigned by the manufacturer.

The regex pattern is

$^{[0-9A-Fa-f]{2}}(:[0-9A-Fa-f]{2}){5}$ .

nullable: false

readOnly: false

writeOnly: false

classChannel:

type: array

items:

type: string

description: A list of global operating class and channel shared as bootstrapping information. It is formatted as class/channel. For example, '81/1', '115/36'.



```
    nullable: false
    readOnly: false
    writeOnly: false
  serialNumber:
    type: string
    description: An alphanumeric serial number that may also be
                 passed as bootstrapping information.
    nullable: false
    readOnly: false
    writeOnly: false
  required:
    - dppVersion
    - bootstrapKey
  additionalProperties: false
```

paths:

/devices:

post:

```
  summary: Create a new device
  operationId: createDevice
  requestBody:
    required: true
    content:
      application/json:
        schema:
          $ref: '#/components/schemas/DppDeviceExtension'
  responses:
    '201':
      description: Created
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/DppDeviceExtension'
```

/devices/{deviceId}:

get:

```
  summary: Get a Dpp device by ID
  description: Returns a single Dpp device by its ID.
  operationId: getDeviceById
  parameters:
    - name: id
      in: path
      description: The ID of the device to return.
      required: true
      schema:
        type: string
  responses:
    '200':
      description: OK
      content:
        application/json:
          schema:
            $ref: '#/components/schemas/DppDeviceExtension'
```

<CODE ENDS>

#### **10.4. Zigbee Extension Schema OpenAPI Representation**

OpenAPI representation of zigbee extension schema is as follows:

```
<CODE BEGINS>
openapi: 3.0.1
info:
  title: Zigbee Device Extension
  description: API for Zigbee device extensions schema
  version: 1.0.0

components:
  schemas:
    ZigbeeDeviceExtension:
      type: object
      properties:
        versionSupport:
          type: array
          items:
            type: string
          description: Provides a list of all the Zigbee versions
            supported by the device. For example, [3.0].
          nullable: false
          readOnly: false
          writeOnly: false
        deviceEui64Address:
          type: string
          description: The EUI-64 (Extended Unique Identifier) device
            address. The regex pattern is
            ^[0-9A-Fa-f]{16}$.
          nullable: false
          readOnly: false
          writeOnly: false
      required:
        - versionSupport
        - deviceEui64Address
      description: Device extension schema for Zigbee.

paths:
  /devices:
    post:
      summary: Create a new device
      operationId: createDevice
      requestBody:
        required: true
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/ZigbeeDeviceExtension'
      responses:
        '201':
          description: Created
          content:
            application/json:
              schema:
                $ref: '#/components/schemas/ZigbeeDeviceExtension'
```

```
/devices/{deviceId}:
  get:
    summary: Get a Zigbee device by ID
    description: Returns a single Zigbee device by its ID.
    operationId: getDeviceById
    parameters:
      - name: id
        in: path
        description: The ID of the device to return.
        required: true
        schema:
          type: string
    responses:
      '200':
        description: OK
        content:
          application/json:
            schema:
              $ref: '#/components/schemas/ZigbeeDeviceExtension'
```

<CODE ENDS>

## **10.5. EndpointApps Extension Schema OpenAPI Representation**

OpenAPI representation of endpoint Apps schema is as follows:

```
<CODE BEGINS>
openapi: 3.0.1
info:
  version: 1.0.0
  title: Endpoint Apps Extension
  description: Extension for partner endpoint applications that can
              onboard, control, and communicate with the device.

components:
  schemas:
    endpointApps:
      type: array
      items:
        allOf:
          - $ref: '#/components/schemas/deviceControl'
          - $ref: '#/components/schemas/dataReceiver'

    deviceControl:
      type: array
      items:
        $ref: '#/components/schemas/deviceControlAttributes'

    deviceControlAttributes:
      properties:
        client-tokens:
          type: string
          format: uri
          description: This attribute contains a list of tokens that
                      the client will used to authenticate itself.
                      Each token may be a string up to 500
                      octets in length.
          nullable: false
          readOnly: false
          writeOnly: false
        enterpriseEndpoint:
          type: string
          format: uri
          description: The URL of the enterprise endpoint which
                      device control apps uses to reach enterprise
                      network gateway. When enterprise receives the
                      SCIM object from onboarding app, it adds this
                      attribute to the object and sends back the
                      object as response back to the onboarding
                      app.
          nullable: false
          readOnly: false
          writeOnly: false
      required:
        - client-tokens
        - enterpriseEndpoint

    dataReceiver:
      type: array
```

```
items:
  $ref: '#/components/schemas/dataReceiverAttributes'

dataReceiverAttributes:
  properties:
    client-tokens:
      type: string
      format: uri
      description: This attribute contains a list of tokens that
                   the client will used to authenticate itself.
                   Each token may be a string up to 500 octets
                   in length.
      nullable: false
      readOnly: false
      writeOnly: false

    enterpriseEndpoint:
      type: string
      format: uri
      description: The URL of the enterprise endpoint which data
                   receiver apps uses to reach enterprise network
                   gateway. When enterprise receives the SCIM
                   object from onboarding app, it adds this
                   attribute to the object and sends back the
                   object as response back to the onboarding app.
      nullable: false
      readOnly: false
      writeOnly: false

paths:
  /endpointApps:
    get:
      summary: Get endpoint apps
      description: Returns all endpoint apps
      operationId: getAllEndpointApps
      responses:
        '200':
          description: OK
          content:
            application/json:
              schema:
                type: array
                items:
                  $ref: '#/components/schemas/endpointApps'
```

<CODE ENDS>



## 11. References

### 11.1. Normative References

- [BLE53] Bluetooth SIG, "Bluetooth Core Specification, Version 5.3", 2021.
- [DPP2] Wi-Fi Alliance, "Wi-Fi Easy Connect Specification, Version 2.0", 2020.
- [I-D.bhutton-json-schema] Wright, A., Andrews, H., Hutton, B., and G. Dennis, "JSON Schema: A Media Type for Describing JSON Documents", Work in Progress, Internet-Draft, draft-bhutton-json-schema-01, 10 June 2022, <<https://datatracker.ietf.org/doc/html/draft-bhutton-json-schema-01>>.
- [RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <<https://www.rfc-editor.org/info/rfc2119>>.
- [RFC7643] Hunt, P., Ed., Grizzle, K., Wahlstroem, E., and C. Mortimore, "System for Cross-domain Identity Management: Core Schema", RFC 7643, DOI 10.17487/RFC7643, September 2015, <<https://www.rfc-editor.org/info/rfc7643>>.
- [RFC7644] Hunt, P., Ed., Grizzle, K., Ansari, M., Wahlstroem, E., and C. Mortimore, "System for Cross-domain Identity Management: Protocol", RFC 7644, DOI 10.17487/RFC7644, September 2015, <<https://www.rfc-editor.org/info/rfc7644>>.
- [RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <<https://www.rfc-editor.org/info/rfc8174>>.
- [RFC8520] Lear, E., Droms, R., and D. Romascanu, "Manufacturer Usage Description Specification", RFC 8520, DOI 10.17487/RFC8520, March 2019, <<https://www.rfc-editor.org/info/rfc8520>>.

### 11.2. Informative References

- [RFC8995] Pritikin, M., Richardson, M., Eckert, T., Behringer, M., and K. Watsen, "Bootstrapping Remote Secure Key Infrastructure (BRSKI)", RFC 8995, DOI 10.17487/RFC8995, May 2021, <<https://www.rfc-editor.org/info/rfc8995>>.

### Authors' Addresses

Muhammad Shahzad  
North Carolina State University

Department of Computer Science  
890 Oval Drive  
Campus Box 8206  
Raleigh, NC, 27695-8206  
United States of America

Email: [mshahza@ncsu.edu](mailto:mshahza@ncsu.edu)

Hassan Iqbal  
North Carolina State University  
Department of Computer Science  
890 Oval Drive  
Campus Box 8206  
Raleigh, NC, 27695-8206  
United States of America

Email: [hiqbal@ncsu.edu](mailto:hiqbal@ncsu.edu)

Eliot Lear  
Cisco Systems  
Richtistrasse 7  
CH-8304 Wallisellen  
Switzerland

Phone: [+41 44 878 9200](tel:+41448789200)

Email: [lear@cisco.com](mailto:lear@cisco.com)