Network Working Group
Internet-Draft
Intended status: Standards Track
Expires: 7 March 2026

M. Shahzad
H. Iqbal
North Carolina State University
E. Lear
Cisco Systems

3 September 2025

Device Schema Extensions to the SCIM model
draft-ietf-scim-device-model-18

Abstract

The initial core schema for SCIM (System for Cross-domain Identity Management) was designed for provisioning users. This memo specifies schema extensions that enables provisioning of devices, using various underlying bootstrapping systems, such as Wi-fi Easy Connect, FIDO device onboarding vouchers, BLE passcodes, and MAC authenticated bypass.

Table of Contents

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-domain Identity Management) [RFC7643] [RFC7644] defines a protocol and a schema for provisioning of users. However, it can easily be extended to provision device credentials

and other attributes into a network. 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.

A primary purpose of this specification is to provision the network for onboarding and communications access to and from devices within a local deployment based on the underlying capabilities of those devices.

The underlying security mechanisms of some devices range from non-existent such as the Bluetooth Low Energy (BLE) "Just Works" pairing method to a robust FIDO Device Onboard (FDO) mechanism. Information from the SCIM server is dispatched to control functions based on selected schema extensions to enable these communications within a network. The SCIM database is therefore essentially equivalent to a network's Authentication, Authorization, and Accounting (AAA) database, and should be carefully managed as such.

1.1. Why SCIM for devices?

There are a number of existing models that might provide the basis for a scheme for provisioning devices onto a network, including two standardised by the IETF: NETCONF [RFC6241] or RESTCONF [RFC8040] with YANG [RFC7950]. SCIM was chosen for the following reasons:

* NETCONF and RESTCONF focus on *configuration* rather than provisioning.

* SCIM is designed with inter-domain provisioning in mind. The use of HTTP as a substrate permits both user-based authentication for local provisioning applications, as well as OAUTH or certificate-based authentication. The inter-domain nature of these operations does not expose local policy, which itself must be (and often is) configured with other APIs, many of which are not standardized.

* SCIM is also a familiar tool within the enterprise enviroment, used extensively to configure federated user accounts.

* Finally, once one chooses a vehicle such as SCIM, one is beholden to its data model. The SCIM data model is more targeted to provisioning as articulated in [RFC7643].

This taken together with the fact that end devices are not intended to be *directly* configured leave us with SCIM as the best standard option.

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. The SCIM server resides within a deployment and is used for 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 Wi-fi Easy Connect QR codes.

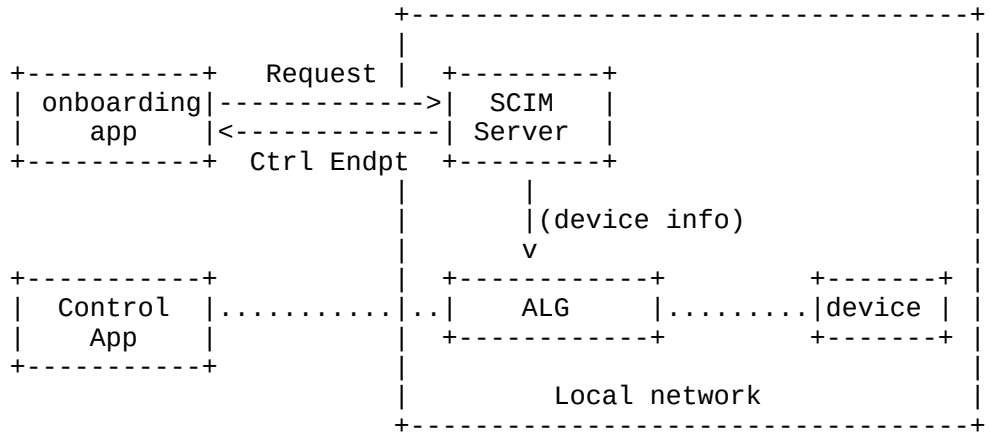


Figure 1: Basic Architecture - non-IP example

In Figure 1, the onboarding application (app) provides the device particulars, which will vary based on the type of device, as indicated by the selection of schema extensions. 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 (c.f., [I-D.ietf-asdf-nipc]). The control endpoint is one among a number of objects that may be returned. That control endpoint will then communicate with the application layer gateway (ALG) to reach the device.

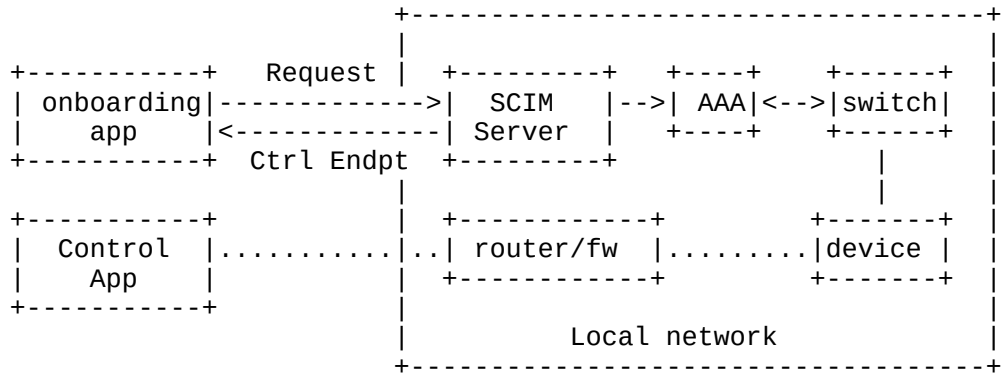


Figure 2: Interaction with AAA

Figure 2 shows how IP-based endpoints can be provisioned. In this case, the onboarding application provisions a device via SCIM. The necessary information is passed to the Authentication, Authorization, and Accounting (AAA) subsystem, such that the device is permitted to connect. Once it is online, since the device is based on IP, it will not need an ALG, but will use the normal IP infrastructure to communicate with its control application.

1.3. Schema Description

RFC 7643 does not prescribe a language to describe a schema, but instead uses narrative description with examples. We follow that approach. In addition, we provide non-normative JSON Schema [JSONSchema] and OpenAPI [OpenAPI] versions in the appendices for ease of implementation, neither of which existed when SCIM was originally developed. The only difference the authors note between the normative schema representations is that JSON Schema and OpenAPI do not have a means to express case sensitivity, and thus attributes that are not case sensitive must be manually validated.

Several additional schemas specify specific onboarding mechanisms, such as Bluetooth Low energy (BLE) [BLE54], Wi-fi Easy Connect [DPP2], and FIDO Device Onboard [FDO11].

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 [RFC7643]. This specification 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.

The reader is also expected to be familiar with the narrative schema language used in [RFC7643].

2. ResourceType Device

A new resource type ’Device’ is specified. The "ResourceType" schema specifies the metadata about a resource type (see Section 6 of [RFC7643]). It comprises a core device schema and several extension schemas. This schema provides a minimal resource representation, whereas extension schemas extend it depending on the device’s capability.

2.1. Common Attributes

The Device schema contains three common attributes as defined in Section 3.1 of [RFC7643]. No semantic or syntax changes are made here, but the attributes are listed merely for completeness.

id: A required and unique attribute of the core device schema (see section 3.1 of [RFC7643]).

externalId: An optional attribute (see section 3.1 of [RFC7643]).

meta: 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, and only one attribute is required. It is identified using the schema URI:

"urn:ietf:params:scim:schemas:core:2.0:Device".

The following attributes are defined in the core device schema.

3.1. Singular Attributes

displayName: A string that 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. It may be modified and SHOULD be returned by default. No uniqueness constraints are imposed on this attribute.

active: A mutable boolean that is required. If set to TRUE, it means that this device is intended to be operational. Attempts to control or access a device where this value is set to FALSE may fail. For example, when used in conjunction with NIPC [I-D.brinckman-nipc], commands such as connect, disconnect, subscribe that control application sends to the controller for the devices any command will be rejected by the controller.

mudUrl: A string that represents the URL to the Manufacturer Usage Description (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.

groups: An optional read-only complex object that indicates group membership. Its form is precisely the same as that defined in Section 4.1.2 of [RFC7643].

Attribute	Multi Value	Req	Case Exact	Mutable	Return	Unique
displayName	F	F	F	RW	Def	None
active	F	T	F	RW	Def	None
mudUrl	F	F	T	RW	Def	None
groups	T	F	T	RO	Def	n/a

Table 1: Characteristics of device schema attributes. (Req = Required, T = True, F = False, RO = ReadOnly, RW = ReadWrite, and Def = Default)

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

Figure 3: Core Device Example Entries

4. Groups

Device and EndpointApp groups are created using the SCIM groups as defined in Section 4.2 of [RFC7643]. If set, the "type" subattribute of the "members" attribute MUST be set to "Device" for devices and "EndpointApp" for endpoint applications.

5. Resource Type EndpointApp

This section defines the ’EndpointApp’ resource type. The "ResourceType" schema specifies the metadata about a resource type (see Section 6 of [RFC7643]). The resource "EndpointApp" represents client applications that can control and/or receive data from the devices.

6. SCIM EndpointApp Schema

The EndpointApp schema is used to authorize control or telemetry services for clients. The schema identifies the application and how clients are to authenticate to the various services.

The schema for "EndpointApp" is identified using the schema URI: "urn:ietf:params:scim:schemas:core:2.0:EndpointApp". The following attributes are defined in this schema.

6.1. Common Attributes

Like Section 2.1 The EndpointApp schema contains the three common attributes specified in Section 3.1 [RFC7643].

6.2. Singular Attributes

applicationType: A string that represents the type of application. It will only contain two values; 'deviceControl' or 'telemetry'. 'deviceControl' is the application that sends commands to control the device. 'telemetry' is the application that receives data from the device. The attribute is required, and is not case-sensitive. The attribute is readOnly and should be returned by default. No uniqueness constraints are imposed on this attribute.

applicationName: a string that represents a human readable name for the application. This attribute is required and mutable. The attribute should be returned by default and there is no uniqueness contraint on the attribute.

clientToken: A string contains a token that the client will use to authenticate itself. Each token may be a string up to 500 characters in length. It is not mutable, read-only, generated if no certificateInfo object is provisioned, case sensitive and returned by default if it exists. The SCIM server should expect that client tokens will be shared by the SCIM client with other components within the client's infrastructure. groups:

An optional read-only complex object that indicates group membership. Its form is precisely the same as that defined in Section 4.1.2 of [RFC7643].

6.3. Complex Attributes

6.3.1. certificateInfo

certificateInfo is a complex attribute that contains x509 certificate's subject name and root CA information associated with application clients that will connect for purposes of device control or telemetry.

rootCA: A base64-encoded string as described in [RFC4648] Section 4 a trust anchor certificate. This trust anchor is applicable for certificates used for client application access. The object is not required, singular, case sensitive, and read/write. If not present, a set of trust anchors MUST be configured out of band.

subjectName: when present, a string taht contains one of two one of two names:

* a distinguished name as that will be present in the certificate subject field, as described in Section 4.1.2.4 of [RFC5280]; or

* or a dnsName as part of a subjectAlternateName as described in Section 4.2.1.6 of [RFC5280].

In the latter case, servers validating such certificates SHALL reject connections when name of the peer as resolved by a DNS reverse lookup does not match the dnsName in the certificate. If multiple dnsNames are present, it is left to server implementations to address any authorization conflicts associated with those names. This attribute is not required, mutable, singular and NOT case sensitive.

Attribute	Multi Value	Req	Case Exact	Mutable	Return	Unique
applicationType	F	T	F	R	Def	None
applicationName	F	T	F	RW	Def	None
clientToken	F	F	T	R	N	None
certificateInfo	F	F	F	RW	Def	None
rootCA	F	F	T	RW	Def	None
subjectName	F	T	T	RW	Def	None

Table 2: Characteristics of EndpointApp schema attributes. (Req = Required, T = True, F = False, R = ReadOnly, RW = ReadWrite, Manuf = Manufacturer, N = No, and Def = Default)

Note that either clientToken or certificateInfo are used for the authentication of the application. If certificateInfo is NOT present when an endpointApp is object created, then the server SHOULD return a clientToken. Otherwise, if the server accepts the certificateInfo object for authentication, it SHOULD NOT return a clientToken. If the server accepts and produces a clientToken, then control and telemetry servers MUST validate both. The SCIM client will know that this is the case based on the SCIM object that is returned.

certificateInfo is preferred in situations where client functions are federated such that different clients may connect for different purposes.

```
<CODE BEGINS>
{
  "schemas": ["urn:ietf:params:scim:schemas:core:2.0:EndpointApp"],
  "id": "e9e30dba-f08f-4109-8486-d5c6a3316212",
  "applicationType": "deviceControl",
  "applicationName": "Device Control App 1",
  "certificateInfo": {
      "rootCA" : "MIIBIjAN...",
      "subjectName": "www.example.com"
  },
  "meta": {
    "resourceType": "EndpointApp",
    "created": "2022-01-23T04:56:22Z",
    "lastModified": "2022-05-13T04:42:34Z",
    "version": "W\/\"a330bc54f0671c9\"",
    "location": "https://example.com/v2/EndpointApps/e9e30dba-f08f
       -4109-8486-d5c6a3316212"
  }
}
<CODE ENDS>
```

Figure 4: Endpoint App Example

7. SCIM Device Extensions

SCIM provides various extension schemas, their attributes, JSON representation, and example object. The core schema is extended with a new resource type, Device. No schemaExtensions list is specified in that definition. Instead, IANA registry entries are created, where all values for "required" are set to false. All extensions to the Device schema MUST be registered via IANA, as described in Section 9.2. The schemas below demonstrate how this model is to work. All the SCIM Server related Schema URIs are valid only with Device resource types.

7.1. Bluetooth Low Energy (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:

7.1.1. Singular Attributes

deviceMacAddress: A string value that represent a public MAC address

assigned by the manufacturer. It is a unique 48-bit value. It is required, case insensitive, is mutable, and is returned by default. The ECMA regular expression pattern [ECMA] is the following:

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

isRandom: A boolean flag taken from [BLE54]. If FALSE, the device is using a public MAC address. If TRUE, the device uses a random address. If an Idenifying Resolving Key (IRK) is present, the address represents a resolvable private address. Otherwise, the address is assumed to be a random static address. Non-resolvable private addresses are not supported by this specification. This attribute is not required. It is mutable, and is returned by default. The default value is FALSE.

separateBroadcastAddress: When present, this string represents an address used for broadcasts/advertisements. This value MUST NOT be set when an IRK is provided. Its form is the same as deviceMacAddress. It is not required, multivalued, mutable, and returned by default.

irk: A string value that specifies the identity resolving key (IRK), which is unique to each device. It is used to resolve private random address. It should only be provisioned when isRandom is TRUE. It is mutable and never returned. For more information about the use of the IRK, see Section 5.4.5 of [BLE54].

mobility: A boolean attribute to enable BLE device mobility. If set to TRUE, the device could be expected to move within a network of APs. For example, BLE device is connected with AP-1 and moves out of range but comes in range of AP-2, it will be disconnected with AP-1 and connects with AP-2. It is returned by default and mutable.

7.1.2. Multivalued Attributes

versionSupport: A multivalued set of strings that specifies 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", "5.4"]. It is required, mutable, and return as default.

pairingMethods: An multivalued set of strings that specifies 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.

7.1.3. BLE Pairing Method Extensions

The details on pairing methods and their associated attributes are in section 5.2.4 of [BLE54]. This memo defines extensions for four pairing methods that are nested insided the BLE extension schema. Each extension contains the common attributes Section 6.1. These extension 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 returned by default.

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

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

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

randomNumber: An integer that represents a nonce added to the key. It is a 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
isRandom	F	T	F	RW	Def	None
sepBroadcastAdd	T	F	F	RW	Def	None
irk	F	F	F	WO	Nev	Manuf
versionSupport	T	T	F	RW	Def	None
mobility	F	F	F	RW	Def	None
pairingMethods	T	T	T	RW	Def	None

Table 3: Characteristics of BLE extension schema attributes. sepBroadcastAdd is short for separateBroadcastAddress. (Req = Required, T = True, F = False, RW = ReadWrite, WO=Write Only, Def = Default, Nev = Never, and Manuf = Manufacturer).

```
<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",
  "displayName": "BLE Heart Monitor",
  "active": true,
  "urn:ietf:params:scim:schemas:extension:ble:2.0:Device" : {
    "versionSupport": ["5.3"],
    "deviceMacAddress": "2C:54:91:88:C9:E2",
    "isRandom": false,
    "separateBroadcastAddress": ["AA:BB:88:77:22:11", "AA:BB:88:77
       :22:12"],
    "mobility": true,
    "pairingMethods": ["urn:ietf:params:scim:schemas:extension
       :pairingPassKey:2.0:Device"],
    "urn:ietf:params:scim:schemas:extension:pairingPassKey:2.0
       :Device" : {
      "key": 123456
    }
  },
  "meta": {
    "resourceType": "Device",
      "created": "2022-01-23T04:56:22Z",
    "lastModified": "2022-05-13T04:42:34Z",
    "version": "W\/\"a330bc54f0671c9\"",
    "location": "https://example.com/v2/Devices/e9e30dba-f08f-4109
       -8486-d5c6a3316111"
  }
}
<CODE ENDS>
```

Figure 5: BLE Example

In the above example, the pairing method is "pairingPassKey", which implies that this BLE device pairs using only a passkey. In another example below, the pairing method is "pairingOOB", denoting that this BLE device uses the out-of-band pairing method.

```
<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",
  "displayName": "BLE Heart Monitor",
  "active": true,
  "urn:ietf:params:scim:schemas:extension:ble:2.0:Device" : {
    "versionSupport": ["5.3"],
    "deviceMacAddress": "2C:54:91:88:C9:E2",
    "isRandom": false,
    "separateBroadcastAddress": ["AA:BB:88:77:22:11", "AA:BB:88:77
       :22:12"],
    "mobility": true,
    "pairingMethods": ["urn:ietf:params:scim:schemas:extension
       :pairingOOB:2.0:Device"],
    "urn:ietf:params:scim:schemas:extension:pairingOOB:2.0:Device":
        {
      "key": "TheKeyvalueRetrievedFromOOB",
      "randomNumber": 238796813516896
    }
  },
  "meta": {
    "resourceType": "Device",
      "created": "2022-01-23T04:56:22Z",
    "lastModified": "2022-05-13T04:42:34Z",
    "version": "W\/\"a330bc54f0671c9\"",
    "location": "https://example.com/v2/Devices/e9e30dba-f08f-4109
       -8486-d5c6a3316111"
  }
}
<CODE ENDS>
```

Figure 6: BLE with pairingOOB

However, a device can have more than one pairing method. Support for multiple pairing methods is also provided by the multi-valued attribute pairingMethods. In the example below, the BLE device can pair with both passkey and OOB pairing methods.

```
<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",
  "displayName": "BLE Heart Monitor",
  "active": true,
  "urn:ietf:params:scim:schemas:extension:ble:2.0:Device" : {
    "versionSupport": ["5.3"],
    "deviceMacAddress": "2C:54:91:88:C9:E2",
    "isRandom": false,
    "separateBroadcastAddress": ["AA:BB:88:77:22:11", "AA:BB:88:77
       :22:12"],
    "mobility": true,
    "pairingMethods": ["urn:ietf:params:scim:schemas:extension
       :pairingPassKey:2.0:Device",
        "urn:ietf:params:scim:schemas:extension:pairingOOB:2.0
           :Device"],
    "urn:ietf:params:scim:schemas:extension:pairingPassKey:2.0
       :Device" : {
      "key": 123456
    },
    "urn:ietf:params:scim:schemas:extension:pairingOOB:2.0:Device":
        {
      "key": "TheKeyvalueRetrievedFromOOB",
      "randomNumber": 238796813516896
    }
  },
  "meta": {
    "resourceType": "Device",
      "created": "2022-01-23T04:56:22Z",
    "lastModified": "2022-05-13T04:42:34Z",
    "version": "W\/\"a330bc54f0671c9\"",
    "location": "https://example.com/v2/Devices/e9e30dba-f08f-4109
       -8486-d5c6a3316111"
  }
}
<CODE ENDS>
```

Figure 7: BLE Pairing with both passkey and OOB

7.2. Wi-Fi Easy Connect Extension

A schema that extends the device schema to enable Wi-Fi Easy Connect (otherwise known as Device Provisioning Protocol or DPP). Throughout this specification we use the term DPP. 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:

7.2.1. 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 an 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: A MAC address stored as string. It is a unique 48-bit value. This attribut is optional, case insensitive, mutable, and returned by default. Its form is identical to that of the deviceMacAddress for BLE devices.

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.

7.2.2. Multivalued Attributes

bootstrappingMethod: One or more 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: One or more strings representing the 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	MultiValue	Req	CaseExact	Mutable	Return	Unique
dppVersion	F	T	F	RW	Def	None
bootstrapKey	F	T	T	WO	Nev	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

Table 4: Characteristics of DPP extension schema attributes. (Req = Required, T = True, F = False, RW = ReadWrite, WO = Write Only, Def = Default, Nev = Never, and Manuf = Manufacturer).

```
<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",
        "active": true,
        "urn:ietf:params:scim:schemas:extension:dpp:2.0:Device" : {
                "dppVersion": 2,
                "bootstrappingMethod": ["QR"],
                "bootstrapKey":
                    "MDkwEwYHKoZIzj0CAQYIKoZIzj0DAQcDIgADURzxmt
                   tZoIRIPWGoQMV00XHWCAQIhXruVWOz0NjlkIA=",
                "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/Devices/e9e30dba-f08f
             -4109-8486-d5c6a3316111"
        }
}
<CODE ENDS>
```

Figure 8: DPP Example

7.3. Ethernet MAB Extension

This extension enables a legacy means of (very) weak authentication, known as MAC Authenticated Bypass (MAB), that is supported in many wired ethernet solutions. If the MAC address is known, then the device may be permitted (perhaps limited) access. The extension is identified by the following URI:

urn:ietf:params:scim:schemas:extension:ethernet-mab:2.0:Device

Note that this method is not likely to work properly with MAC address randomization.

7.3.1. Single Attribute

This extension has a singular attribute:

deviceMacAddress: This is the Ethernet address to be provisioned onto the network. It takes the identical form as found in the BLE extension.

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

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

```
<CODE BEGINS>
{
  "schemas": ["urn:ietf:params:scim:schemas:core:2.0:Device",
     "urn:ietf:params:scim:schemas:extension:ethernet-mab:2.0
        :Device"],

  "id": "e9e30dba-f08f-4109-8486-d5c6a3316111",
  "displayName": "Some random Ethernet Device",
  "active": true,
  "urn:ietf:params:scim:schemas:extension:ethernet-mab:2.0:Device"
     : {
    "deviceMacAddress": "2C:54:91:88:C9:E2"
  },

  "meta": {
    "resourceType": "Device",
      "created": "2022-01-23T04:56:22Z",
    "lastModified": "2022-05-13T04:42:34Z",
    "version": "W\/\"a330bc54f0671c9\"",
    "location": "https://example.com/v2/Devices/e9e30dba-f08f-4109
       -8486-d5c6a3316111"
  }
}
<CODE ENDS>
```

Figure 9: MAB Example

7.4. FIDO Device Onboard Extension

This extension specifies a voucher to be used by the FDO Device Onboard (FDO) protocols [FDO11] to complete a trusted transfer of ownership and control of the device to the environment. The SCIM server MUST know how to process the voucher, either directly or by forwarding it along to an owner process as defined in the FDO specification.

urn:ietf:params:scim:schemas:extension:fido-device-onboard:2.0:Device

7.4.1. Single Attribute

This extension has a singular attribute:

fdoVoucher: The voucher is formated as a PEM-encoded object in accordance with [FDO11].

Attribute	Multi Value	Req	Case Exact	Mutable	Return	Unique
fdoVoucher	F	T	F	WO	Nev	None

Table 6: Characteristics of FDO extension schema attributes (Req = Required, T = True, F = False, WO = WriteOnly, and Nev = Never)

```
<CODE BEGINS>
{
  "schemas": ["urn:ietf:params:scim:schemas:core:2.0:Devices",
     "urn:ietf:params:scim:schemas:extension:fido-device-onboard
        :2.0:Devices"],

  "id": "e9e30dba-f08f-4109-8486-d5c6a3316111",
  "displayName": "Some random Ethernet Device",
  "active": true,
  "urn:ietf:params:scim:schemas:extension:fido-device-onboard:2.0
     :Devices" : {
     "fdoVoucher": "{... voucher ...}"
  },

  "meta": {
    "resourceType": "Device",
      "created": "2022-01-23T04:56:22Z",
    "lastModified": "2022-05-13T04:42:34Z",
    "version": "W\/\"a330bc54f0671c9\"",
    "location": "https://example.com/v2/Devices/e9e30dba-f08f-4109
       -8486-d5c6a3316111"
  }
}
<CODE ENDS>
```

Figure 10: FDO Example

7.5. Zigbee Extension

A schema that extends the device schema to enable the provisioning of Zigbee devices [Zigbee]. 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:

7.5.1. 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. It takes the same form as the deviceMACaddress in the BLE extension.

7.5.2. Multivalued Attribute

versionSupport: One or more 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

Table 7: Characteristics of Zigbee extension schema attributes. (Req = Required, T = True, F = False, RW = ReadWrite, and Def = Default)

```
<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",
  "active": true,
  "urn:ietf:params:scim:schemas:extension:zigbee:2.0:Device" : {
    "versionSupport": ["3.0"],
    "deviceEui64Address": "50:32:5F:FF:FE:E7:67:28"
  },

  "meta": {
    "resourceType": "Device",
      "created": "2022-01-23T04:56:22Z",
    "lastModified": "2022-05-13T04:42:34Z",
    "version": "W\/\"a330bc54f0671c9\"",
    "location": "https://example.com/v2/Devices/e9e30dba-f08f-4109
       -8486-d5c6a3316111"
  }
}
<CODE ENDS>
```

Figure 11: Zigbee Example

7.6. 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.

endpointAppsExt provides the list of applications that connect to enterprise gateway. The endpointAppsExt has one multivalued attribute and two singular attributes. The extension is identified using the following schema URI:

urn:ietf:params:scim:schemas:extension:endpointAppsExt:2.0:Device

7.6.1. Singular Attributes

deviceControlEnterpriseEndpoint: A string representing the URL of the enterprise endpoint to reach the enterprise gateway. When the enterprise receives the SCIM object from the onboarding application, it adds this attribute to it and sends it back as a response to the onboarding application. This attribute is required, case-sensitive, mutable, and returned by default. The uniqueness is enforced by the enterprise.

telemetryEnterpriseEndpoint: A string representing a URL of the enterprise endpoint to reach the an enterprise gateway for telemetry. When the enterprise receives the SCIM object from the onboarding application, it adds this attribute to it and sends it back as a response to the onboarding application. This attribute is optional, case-sensitive, mutable, and returned by default. The uniqueness is enforced by the enterprise. An implementation MUST generate an exception if telemetryEnterpriseEndpoint is not returned and telemetry is required for the proper functioning of a device.

7.6.2. Multivalued Attribute

applications: A multivalued attribute of one or more complex attributes that represent a list of endpoint applications i.e., deviceControl and telemetry. Each entry in the list comprises two attributes including "value" and "$ref".

value: A string containingthe identifier of the endpoint application formated as UUID. It is same as the common attribute "$id" of the resource "endpointApp". It is read/write, required, case insensitive and returned by default.

$ref: A reference to the respective endpointApp resource object stored in the SCIM server. It is readOnly, required, case sensitive and returned by default.

Attribute	Multi Value	Req	Case Exact	Mutable	Return	Unique
devContEntEndpoint	F	T	T	R	Def	Ent
telEntEndpoint	F	F	T	R	Def	Ent
applications	T	T	F	RW	Def	None
value	F	T	F	RW	Def	None
$ref	F	T	F	R	Def	None

Table 8: Characteristics of EndpointAppsExt extension schema attributes. DevContEntEndpoint represents attribute deviceControlEnterpriseEndpoint and telEntEndpoint represents telemetryEnterpriseEndpoint. (Req = Required, T = True, F = False, R = ReadOnly, RW = ReadWrite, Ent = Enterprise, and Def = Default).

<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:endpointAppsExt:2.0
        :Device"],
  "id": "e9e30dba-f08f-4109-8486-d5c6a3316111",
  "displayName": "BLE Heart Monitor",
  "active": true,
  "urn:ietf:params:scim:schemas:extension:ble:2.0:Device" : {
    "versionSupport": ["5.3"],
    "deviceMacAddress": "2C:54:91:88:C9:E2",
    "isRandom": false,
    "separateBroadcastAddress": ["AA:BB:88:77:22:11", "AA:BB:88:77
       :22:12"],
    "mobility": false,
    "pairingMethods": [
        "urn:ietf:params:scim:schemas:extension:pairingPassKey:2.0
           :Device"],
    "urn:ietf:params:scim:schemas:extension:pairingPassKey:2.0
       :Device" : {
      "key": 123456
    }
  },
  "urn:ietf:params:scim:schemas:extension:endpointAppsExt:2.0
     :Device": {
```

```
      "applications": [
        {
          "value" : "e9e30dba-f08f-4109-8486-d5c6a3316212",
          "$ref" : "https://example.com/v2/EndpointApps/e9e30dba-f08f
             -4109-8486-d5c6a3316212"
        },
        {
          "value" : "e9e30dba-f08f-4109-8486-d5c6a3316333",
          "$ref" : "https://example.com/v2/EndpointApps/e9e30dba-f08f
             -4109-8486-d5c6a3316333"
        }
      ],
        "deviceControlEnterpriseEndpoint": "https
           ://example.com/device_control_app_endpoint/",
        "telemetryEnterpriseEndpoint": "https
           ://example.com/telemetry_app_endpoint/"
    },
    "meta": {
      "resourceType": "Device",
        "created": "2022-01-23T04:56:22Z",
      "lastModified": "2022-05-13T04:42:34Z",
      "version": "W\/\"a330bc54f0671c9\"",
      "location": "https://example.com/v2/Devices/e9e30dba-f08f-4109
         -8486-d5c6a3316111"
    }
  }
  <CODE ENDS>
```

Figure 12: Endpoint Applications Extension Example

The schema for the endpointAppsExt extension along with BLE extension is presented in JSON format in Appendix B.9, while the openAPI representation is provided in Appendix C.8.

8. Security Considerations

Because provisioning operations permit device access to a network, each SCIM client MUST be appropriately authenticated.

8.1. SCIM operations

An attacker that has authenticated to a trusted SCIM client could manipulate portions of the SCIM database. To be clear on the risks, we specify each operation below:

8.1.1. Unauthorized Object Creation

An attacker that is authenticated could attempt to add elements that the enterprise would not normally permit on a network. For instance, an enterprise may not wish specific devices that have well-known vulnerabilities to be introduced to their environment. To mitigate the attack, network administrators should layer additional policies regarding what devices are permitted on the network.

An attacker that gains access to SCIM could attempt to add an IP-based device that itself attempts unauthorized access, effectively acting as a Bot. Network administrators SHOULD establish appropriate access-control policies that follow the principle of least privilege to mitigate this attack.

8.2. Object Deletion

Once granted, even if the object is removed, the server may or may not act on that removal. The deletion of the object is a signal of intent by the application that it no longer expects the device to be on the network. It is strictly up to the SCIM server and its back end policy to decide whether or not to revoke access to the infrastructure. It is RECOMMENDED that SCIM delete operations trigger a workflow in accordance with local network policy.

8.3. Read operations

Read operations are necessary in order for an application to sync its state to know what devices it is expected to manage. An attacker with access to SCIM objects may gain access to the devices themselves. To prevent one SCIM client from interfering with devices that it has no business managing, only clients that have created objects or those they authorize SHOULD have the ability to read those objects.

8.4. Update Operations

Update operations may be necessary if a device has been modified in some way. Attackers with update access may be able to disable network access to devices or device access to networks. To avoid this, the same access control policy for read operations is RECOMMENDED here.

8.5. Higher level protection for certain systems

Devices provisioned with this model may be completely controlled by the administrator of the SCIM server, depending on how those systems are defined. For instance, if BLE passkeys are provided, the device can be connected to, and perhaps paired with. If the administrator of the SCIM client does not wish the network to have complete access to the device, the device itself MUST support finer levels of access control and additional authentication mechanisms. Any additional security must be provided at higher application layers. For example, if client applications wish to keep private information to and from the device, they should encrypt that information over-the-top.

8.6. Logging

An attacker could learn what devices are on a network by examining SCIM logs. Due to the sensitive nature of SCIM operations, logs SHOULD be encrypted both on the disk and in transit.

9. IANA Considerations

9.1. New Schemas

The IANA is requested to add the following additions to the "SCIM Schema URIs for Data Resources" registry as follows:

URN	Name	Reference
urn:ietf:params:scim:schemas:core: 2.0:Device	Core Device Schema	This memo, Section 3
urn:ietf:params:scim:schemas:core: 2.0:EndpointApp	Endpoint Application	This memo, Section 6

Table 9

Note that the line break in URNs should be removed, as should this comment.

9.2. Device Schema Extensions

IANA is requested to create the following extensions in the SCIM Server-Related Schema URIs registry as described in Section 7:

URN	Description	Resource Type	Reference
urn:ietf:params:scim:schemas:extension:ble:2.0:Device	BLE Extension	Device	This memo, Section 7.1
urn:ietf:params:scim:schemas:extension: ethernet-mab:2.0:Device	Ethernet MAB	Device	This memo, Section 7.3
urn:ietf:params:scim:schemas:extension: fido-device-onboard:2.0:Device	FIDO Device Onboard	Device	This memo, Section 7.4
urn:ietf:params:scim:schemas:extension:dpp:2.0:Device	Wi-fi Easy Connect	Device	This memo, Section 7.2
urn:ietf:params:scim:schemas:extension:endpointAppsExt:2.0:Device	Application Endpoint Extension	Device	This memo, Section 7.1.3
urn:ietf:params:scim:schemas:extension:pairingJustWorks:2.0:Device	Just Works Auth BLE	Device	This memo, Section 7.1.3
urn:ietf:params:scim:schemas:extension:pairingOOB:2.0:Device	Out of Band Pairing for BLE	Device	This memo, Section 7.1.3
urn:ietf:params:scim:schemas:extension:pairingPassKey:2.0:Device	Passkey Pairing for BLE	Device	This memo, Section 7.1.3

Table 10

10. Acknowledgments

The authors would like to thank Bart Brinckman, Rohit Mohan, Lars Streubesand, Christian Ams端ss, Jason Livingwood, Mike Ounsworth, Monty Wiseman, Geoffrey Cooper, Paulo Jorge N. Correia, Phil Hunt, and Elwyn Davies for their reviews, and Nick Ross for his contribution to the Appendix.

11. References

11.1. Normative References

[BLE54] Bluetooth SIG, "Bluetooth Core Specification, Version 5.4", 2023, <https://www.bluetooth.org/DocMan/handlers/DownloadDoc.ashx?doc_id=587177>.

[DPP2] Wi-Fi Alliance, "Wi-Fi Easy Connect Specification, Version 2.0", 2020.

[ECMA] ECMA International, "ECMA-262, 16th Edition", June 2025, <https://ecma-international.org/publications-and-standards/standards/ecma-262/>.

[FDO11] FIDO Alliance, "FIDO Device Onboard Specification 1.1", April 2022.

[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>.

[RFC4648] Josefsson, S., "The Base16, Base32, and Base64 Data Encodings", RFC 4648, DOI 10.17487/RFC4648, October 2006, <https://www.rfc-editor.org/info/rfc4648>.

[RFC5280] Cooper, D., Santesson, S., Farrell, S., Boeyen, S., Housley, R., and W. Polk, "Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile", RFC 5280, DOI 10.17487/RFC5280, May 2008, <https://www.rfc-editor.org/info/rfc5280>.

[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>.

[Zigbee] Zigbee Alliance, "Zigbee Specification", August 2015, <https://zigbeealliance.org/wp-content/uploads/2019/11/docs-05-3474-21-0csg-zigbee-specification.pdf>.

11.2. Informative References

[I-D.brinckman-nipc]
Brinckman, B., Mohan, R., and B. Sanford, "An Application Layer Interface for Non-IP device control (NIPC)", Work in Progress, Internet-Draft, draft-brinckman-nipc-01, 21 April 2024, <https://datatracker.ietf.org/doc/html/draft-brinckman-nipc-01>.

[I-D.ietf-asdf-nipc]
Brinckman, B., Mohan, R., and B. Sanford, "An Application Layer Interface for Non-IP device control (NIPC)", Work in Progress, Internet-Draft, draft-ietf-asdf-nipc-12, 19 August 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-asdf-nipc-12>.

[JSONSchema]
Wright, A., Ed., Andrews, H. A., Ed., Hutton, B., Ed., and G. Dennis, "JSON Schema- A Media Type for Describing JSON Documents", December 2022, <https://json-schema.org/draft/2020-12/json-schema-core>.

[OpenAPI] swagger.io, "OpenAPI Specification, Version 3.1.1", October 2024, <https://swagger.io/specification/>.

[RFC6241] Enns, R., Ed., Bjorklund, M., Ed., Schoenwaelder, J., Ed., and A. Bierman, Ed., "Network Configuration Protocol (NETCONF)", RFC 6241, DOI 10.17487/RFC6241, June 2011, <https://www.rfc-editor.org/info/rfc6241>.

[RFC7950] Bjorklund, M., Ed., "The YANG 1.1 Data Modeling Language", RFC 7950, DOI 10.17487/RFC7950, August 2016, <https://www.rfc-editor.org/info/rfc7950>.

[RFC8040] Bierman, A., Bjorklund, M., and K. Watsen, "RESTCONF Protocol", RFC 8040, DOI 10.17487/RFC8040, January 2017, <https://www.rfc-editor.org/info/rfc8040>.

[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>.

Appendix A. Changes from Earlier Versions

[RFC Editor to remove this section.]

Draft 17:

* Fix example.

Draft 16:

* More DISCUSS resolution: make clear that JSON Schema is not normative

* Add reference for ECMA for regex

* lots of typo/spelling error cleanup

* Add figure labels for examples

* fix an aasvg rendering problem

* add some reference targets.

* Elwyn Davies review suggestions.

Drafts 17: * Post DISCUSS hiccup with groups. * Add OpenAPI header * multivalues->multivalued * externalID->externalId * remove nullable (wasn’t doing anything) * Update appropriate json schema and openapi accordingly.

Drafts 14, 15, 16: * Resolve DISCUSSes

Draft 13: * post IANA and IETF LC

Drafts 10-12: * additional WGLC and shepherd comments

Draft -09: * last call comments, bump BLE version, add acknowledgments. * Also, recapture Rohit comments and those of Christian.

Drafts 04-08: * Lots of cleanup * Security review responses * Removal of a tab * Dealing with certificate stuff

Draft -03: * Add MAB, FDO * Some grammar improvements * fold OpenAPI * IANA considerations

Draft -02: * Clean up examples * Move openapi to appendix Draft -01:

* Doh! We forgot the core device scheme!

Draft -00:

* Initial revision

Appendix B. JSON Schema Representation

B.1. Resource Schema

<CODE BEGINS>

```
[
  {
    "schemas": ["urn:ietf:params:scim:schemas:core:2.0
       :ResourceType"],
    "id": "Device",
    "name": "Device",
    "endpoint": "/Devices",
    "description": "Device Account",
    "schema": "urn:ietf:params:scim:schemas:core:2.0:Device",
    "meta": {
      "location": "https://example.com/v2/ResourceTypes/Device",
      "resourceType": "ResourceType"
    }
  },
  {
    "schemas": ["urn:ietf:params:scim:schemas:core:2.0
       :ResourceType"],
    "id": "EndpointApp",
    "name": "EndpointApp",
    "endpoint": "/EndpointApp",
    "description": "Endpoint application such as device control and
        telemetry.",
    "schema": "urn:ietf:params:scim:schemas:core:2.0:EndpointApp",
    "meta": {
      "location": "https
         ://example.com/v2/ResourceTypes/EndpointApp",
      "resourceType": "ResourceType"
    }
  }
]
```

<CODE ENDS>

B.2. Core Device Schema

```
<CODE BEGINS>
{
  "id": "urn:ietf:params:scim:schemas:core:2.0:Device",
  "name": "Device",
  "description": "Entry containing attributes about a device",
  "attributes" : [
    {
      "name": "displayName",
      "type": "string",
      "description": "Human readable name of the device, suitable
          for displaying to end-users. For example, 'BLE Heart
          Monitor' etc.",
      "multiValued": false,
      "required": false,
      "caseExact": false,
      "mutability": "readWrite",
      "returned": "default",
      "uniqueness": "none"
    },
    {
      "name": "active",
      "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 processeed by the
          controller. If set FALSE, any command comming from the
          control app for the device will be rejected by the
          controller.",
      "multiValued": 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).",
      "multiValued": false,
      "required": false,
      "caseExact": true,
      "mutability": "readWrite",
      "returned": "default",
      "uniqueness": "none"
```

```
      },
      {
        "name": "groups",
        "type": "complex",
        "multiValued": true,
        "description": "A list of groups to which the device belongs,
            either through direct membership, through nested groups,
            or dynamically calculated.",
        "required": false,
        "subAttributes": [
          {
            "name": "value",
            "type": "string",
            "multiValued": false,
            "description": "The identifier of the Device’s group.",
            "required": false,
            "caseExact": false,
            "mutability": "readOnly",
            "returned": "default",
            "uniqueness": "none"
          },
          {
            "name": "$ref",
            "type": "reference",
            "referenceTypes": [
              "Group"
            ],
            "multiValued": false,
            "description": "The URI of the corresponding ’Group’
                resource to which the device belongs.",
            "required": false,
            "caseExact": false,
            "mutability": "readOnly",
            "returned": "default",
            "uniqueness": "none"
          },
          {
            "name": "display",
            "type": "string",
            "multiValued": false,
            "description": "A human-readable name, primarily used for
                display purposes.  READ-ONLY.",
            "required": false,
            "caseExact": false,
            "mutability": "readOnly",
            "returned": "default",
            "uniqueness": "none"
          },
```

```
          {
            "name": "type",
            "type": "string",
            "multiValued": false,
            "description": "A label indicating the attribute's
                function, e.g., 'direct' or 'indirect'.",
            "required": false,
            "caseExact": false,
            "canonicalValues": [
              "direct",
              "indirect"
            ],
            "mutability": "readOnly",
            "returned": "default",
            "uniqueness": "none"
          }
        ],
        "mutability": "readOnly",
        "returned": "default"
      }
  ],
    "meta" : {
      "resourceType" : "Schema",
      "location" :
        "/v2/Schemas/urn:ietf:params:scim:schemas:core:2.0:Device"
    }
  }
  <CODE ENDS>
```

B.3. EndpointApp Schema

```
  <CODE BEGINS>
  {
    "id": "urn:ietf:params:scim:schemas:core:2.0:EndpointApp",
    "name": "EndpointApp",
    "description": "Endpoint application and their credentials",
    "attributes" : [
      {
        "name": "applicationType",
        "type": "string",
        "description": "This attribute will only contain two values;
            'deviceControl' or 'telemetry'.",
        "multiValued": false,
        "required": true,
        "caseExact": false,
        "mutability": "readOnly",
        "returned": "default",
        "uniqueness": "none"
```

```
      },
      {
        "name": "applicationName",
        "type": "string",
        "description": "Human readable name of the application.",
        "multiValued": false,
        "required": true,
        "caseExact": false,
        "mutability": "readWrite",
        "returned": "default",
        "uniqueness": "none"
      },
      {
        "name": "certificateInfo",
        "type": "complex",
        "description": "Contains x509 certificate's subject name and
            root CA information associated with the device control or
            telemetry app.",
        "multiValued": false,
        "required": false,
        "caseExact": false,
        "mutability": "readWrite",
        "returned": "default",
        "uniqueness": "none",
        "subAttributes" : [
          {
            "name" : "rootCA",
            "type" : "string",
            "description" : "The base64 encoding of the DER encoding
                of the CA certificate",
            "multiValued" : false,
            "required" : false,
            "caseExact" : true,
            "mutability" : "readWrite",
            "returned" : "default",
            "uniqueness" : "none"
          },
          {
            "name" : "subjectName",
            "type" : "string",
            "description" : "A Common Name (CN) of the form of CN =
                dnsName",
            "multiValued" : false,
            "required" : true,
            "caseExact" : true,
            "mutability" : "readWrite",
            "returned" : "default",
            "uniqueness" : "none"
```

```
          }
        ]
      },
      {
        "name": "clientToken",
        "type": "string",
        "description": "This attribute contains a token that the
            client will use to authenticate itself.  Each token may
            be a string up to 500 characters in length.",
        "multiValued": false,
        "required": false,
        "caseExact": true,
        "mutability": "readOnly",
        "returned": "default",
        "uniqueness": "none"
      },
      {
        "name": "groups",
        "type": "complex",
        "multiValued": true,
        "description": "A list of groups to which an endpoint
            application belongs, either through direct membership,
            through nested groups, or dynamically calculated.",
        "required": false,
        "subAttributes": [
          {
            "name": "value",
            "type": "string",
            "multiValued": false,
            "description": "The identifier of the endpoint
                application's group.",
            "required": false,
            "caseExact": false,
            "mutability": "readOnly",
            "returned": "default",
            "uniqueness": "none"
          },
          {
            "name": "$ref",
            "type": "reference",
            "referenceTypes": [
              "Group"
            ],
            "multiValued": false,
            "description": "The URI of the corresponding 'Group'
                resource to which the endpoint application belongs.",
            "required": false,
            "caseExact": false,
```

```
            "mutability": "readOnly",
            "returned": "default",
            "uniqueness": "none"
          },
          {
            "name": "display",
            "type": "string",
            "multiValued": false,
            "description": "A human-readable name, primarily used for
                display purposes.  READ-ONLY.",
            "required": false,
            "caseExact": false,
            "mutability": "readOnly",
            "returned": "default",
            "uniqueness": "none"
          },
          {
            "name": "type",
            "type": "string",
            "multiValued": false,
            "description": "A label indicating the attribute’s
                function, e.g., ’direct’ or ’indirect’.",
            "required": false,
            "caseExact": false,
            "canonicalValues": [
              "direct",
              "indirect"
            ],
            "mutability": "readOnly",
            "returned": "default",
            "uniqueness": "none"
          }
        ],
        "mutability": "readOnly",
        "returned": "default"
      }
    ],
    "meta" : {
      "resourceType" : "Schema",
      "location" :
        "/v2/Schemas/urn:ietf:params:scim:schemas:core:2.0:Device"
    }
  }
  <CODE ENDS>
```

B.4. BLE Extension Schema

<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].",
        "multiValued": true,
        "required": true,
        "caseExact": false,
        "mutability": "readWrite",
        "returned": "default",
        "uniqueness": "none"
      },
      {
        "name": "deviceMacAddress",
        "type": "string",
        "pattern": "^[0-9A-Fa-f]{2}(:[0-9A-Fa-f]{2}){5}$",
        "description": "A unique public MAC address assigned by the
            manufacturer.",
        "multiValued": false,
        "required": true,
        "caseExact": false,
        "mutability": "readWrite",
        "returned": "default",
        "uniqueness": "Manufacturer"
      },
      {
        "name": "isRandom",
        "type": "boolean",
          "description": "The isRandom flag is taken from the BLE
              core specifications 5.3. If TRUE, device is using a
              random address.  Default value is false.",
        "multiValued": false,
        "required": false,
        "caseExact": false,
        "mutability": "readWrite",
        "returned": "default",
        "uniqueness": "none"
      },
      {
        "name": "separateBroadcastAddress",
```

```
          "type": "string",
          "description": "When present, this address is used for
              broadcasts/advertisements.  This value MUST NOT be set
              when an IRK is provided.  Its form is the same as
              deviceMa‘cAddress.",
          "multiValued": true,
          "required": false,
          "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.
              This value MUST NOT be set when
              separateBroadcastAddress is set.",
          "multiValued": false,
          "required": false,
          "caseExact": false,
          "mutability": "readWrite",
          "returned": "default",
          "uniqueness": "Manufacturer"
        },
        {
          "name": "mobility",
          "type": "bool",
          "description": "If set to True, the BLE device will
              automatically connect to the closest AP. For example,
              BLE device is connected with AP-1 and moves out of
              range but comes in range of AP-2, it will be
              disconnected with AP-1 and connects with AP-2.",
          "multiValued": false,
          "required": false,
          "caseExact": false,
          "mutability": "readWrite",
          "returned": "default",
          "uniqueness": "none"
        },
        {
          "name": "pairingMethods",
          "type": "string",
          "description": "List of pairing methods associated with the
              ble device, stored as schema URI.",
          "multiValued": 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'.",
          "multiValued": 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:pairingOOB:2.0:Device"
      }
    }
  ]
  <CODE ENDS>
```

B.5. DPP Extension Schema

```
  <CODE BEGINS>
  {
      "id": "urn:ietf:params:scim:schemas:extension:dpp:2.0:Device",
      "name": "dppExtension",
      "description": "Device extension schema for Wi-Fi Easy Connect
          / Device Provisioning Protocol (DPP)",
      "attributes" : [
        {
          "name": "dppVersion",
          "type": "integer",
          "description": "Version of DPP this device supports.",
          "multiValued": 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].",
          "multiValued": true,
          "required": false,
          "caseExact": false,
          "mutability": "readWrite",
          "returned": "default",
          "uniqueness": "none"
        },
        {
          "name": "bootstrapKey",
          "type": "string",
          "description": "A base64-encoded Elliptic-Curve Diffie
             -Hellman public key (may be P-256, P-384, or P-521).",
          "multiValued": false,
          "required": true,
          "caseExact": true,
          "mutability": "readWrite",
          "returned": "default",
          "uniqueness": "none"
        },
        {
          "name": "deviceMacAddress",
          "type": "string",
          "pattern": "^[0-9A-Fa-f]{2}(:[0-9A-Fa-f]{2}){5}$",
          "description": "A unique public MAC address assigned by the
              manufacturer.",
          "multiValued": 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'.",
          "multiValued": 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.",
          "multiValued": 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>
```

B.6. Ethernet MAB Extension Schema

```
<CODE BEGINS>
{
  "id": "urn:ietf:params:scim:schemas:extension:ethernet-mab:2.0
     :Device",
  "name": "ethernetMabExtension",
  "description": "Device extension schema for MAC authentication
      Bypass.",
  "attributes" : [
      {
        "name": "deviceMacAddress",
        "type": "string",
        "pattern": "^[0-9A-Fa-f]{2}(:[0-9A-Fa-f]{2}){5}$",
        "description": "A MAC address assigned by the manufacturer",
        "multiValued": false,
        "required": true,
        "caseExact": false,
        "mutability": "readWrite",
        "returned": "default",
        "uniqueness": "Manufacturer"
      }
  ],
  "meta" : {
    "resourceType" : "Schema",
    "location" : "/v2/Schemas/urn:ietf:params:scim:schemas
       :extension:ethernet-mab:2.0:Device"
  }
}
<CODE ENDS>
```

B.7. FDO Extension Schema

```
<CODE BEGINS>
{
  "id": "urn:ietf:params:scim:schemas:extension:fido-device-onboard
     :2.0:Devices",
  "name": "FDOExtension",
  "description": "Device extension schema for FIDO Device Onboard
     (FDO).",
  "attributes" : [
      {
        "name": "fdoVoucher",
        "type": "string",
        "description": "A voucher as defined in the FDO
            specification",
        "multiValued": false,
        "required": true,
        "caseExact": false,
        "mutability": "readWrite",
        "returned": "default",
        "uniqueness": "Manufacturer"
      }
  ],
  "meta" : {
    "resourceType" : "Schema",
    "location" : "/v2/Schemas/urn:ietf:params:scim:schemas
       :extension:fido-device-onboard:2.0:Devices"
  }
}
<CODE ENDS>
```

B.8. Zigbee Extension Schema

```
<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",
      "pattern": "^[0-9A-Fa-f]{2}(:[0-9A-Fa-f]{2}){7}$",
      "description": "The EUI-64 (Extended Unique Identifier)
          device address.",
      "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>
```

B.9. EndpointAppsExt Extension Schema

```
<CODE BEGINS>
{
  "id": "urn:ietf:params:scim:schemas:extension:endpointAppsExt:2.0
     :Device",
  "name": "endpointAppsExt",
  "description": "Extension for partner endpoint applications that
```

```
        can onboard, control, and communicate with the device.",
    "attributes" : [
      {
        "name": "applications",
        "type": "complex",
        "description": "Includes references to two types of
            application that connect with entrprise, i.e.,
            deviceControl and telemetry.",
        "multiValued": true,
        "required": true,
        "caseExact": false,
        "mutability": "readWrite",
        "returned": "default",
        "uniqueness": "none",
        "subAttributes" : [
          {
            "name" : "value",
            "type" : "string",
            "description" : "The identifier of the endpointApp.",
            "multiValued" : false,
            "required" : true,
            "caseExact" : false,
            "mutability" : "readWrite",
            "returned" : "default",
            "uniqueness" : "none"
          },
          {
            "name" : "$ref",
            "type" : "reference",
            "referenceTypes" : "EndpointApps",
            "description" : "The URI of the corresponding
                'EndpointApp' resource which will control or obtain
                data from the device.",
            "multiValued" : false,
            "required" : false,
            "caseExact" : true,
            "mutability" : "readOnly",
            "returned" : "default",
            "uniqueness" : "none"
          }
        ]
      },
      {
        "name": "deviceControlEnterpriseEndpoint",
        "type": "reference",
        "description": "The URL of the enterprise endpoint which
            device control apps use to reach enterprise network
            gateway.",
```

```
        "multiValued": false,
        "required": true,
        "caseExact": true,
        "mutability": "readOnly",
        "returned": "default",
        "uniqueness": "Enterprise"
      },
      {
        "name": "telemetryEnterpriseEndpoint",
        "type": "reference",
        "description": "The URL of the enterprise endpoint which
            telemetry apps use to reach enterprise network gateway.",
        "multiValued": false,
        "required": false,
        "caseExact": true,
        "mutability": "readOnly",
        "returned": "default",
        "uniqueness": "Enterprise"
      }
    ],
    "meta" : {
      "resourceType" : "Schema",
      "location" : "/v2/Schemas/urn:ietf:params:scim:schemas
         :extension:endpointAppsExt:2.0:Device"
    }
  }
  <CODE ENDS>
```

Appendix C. OpenAPI representation

The following sections are provided for informational purposes.

C.1. Core Device Schema OpenAPI Representation

OpenAPI representation of core device schema is as follows:

```
<CODE BEGINS>
openapi: 3.1.0
info:
  title: SCIM Device Schema
  version: 1.0.0

components:
  schemas:
    Group:
      type: object
      description: A list of groups to which the device belongs,
                   either through direct membership, through nested
```

```
                groups, or dynamically calculated.
      properties:
        value:
          type: string
          description: the unique identifier of a group,
                       typically a UUID.
          readOnly: true
          writeOnly: false
        display:
          type: string
          description: a display string for the group.
          readOnly: true
          writeOnly: false
        $ref:
          type: string
          format: uri
          description: reference to the group object
          readOnly: true
          writeOnly: true
    Device:
      description: Entry containing attributes about a device
      type: object
      properties:
        displayName:
          type: string
          description: "Human readable name of the device, suitable
                        for displaying to end-users. For example,
                       'BLE Heart Monitor' etc."
          readOnly: false
          writeOnly: false
        active:
          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 processeed
                       by the controller.  If set FALSE, any command
                       comming from the control app for the device
                       will be rejected by the controller.
          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.
            readOnly: false
            writeOnly: false
          groups:
            type: array
            description: list of groups device belongs to
            items:
              $ref: '#/components/schemas/Group'

        required:
          - active
        additionalProperties: false
        allOf:
          - $ref: '#/components/schemas/CommonAttributes'
      CommonAttributes:
        type: object
        properties:
          schemas:
            type: array
            items:
              type: string
              enum:
                - urn:ietf:params:scim:schemas:core:2.0:Device
            description: The list of schemas that define the resource.
          id:
            type: string
            format: uri
            description: The unique identifier for a resource.
            readOnly: true
            writeOnly: false
          externalId:
            type: string
            description: An identifier for the resource that is
                defined
                        by the provisioning client.
            readOnly: false
            writeOnly: false
          meta:
            type: object
            readOnly: true
            properties:
              resourceType:
                type: string
                description: The name of the resource type of the
                             resource.
                readOnly: true
                writeOnly: false
              location:
```

```
                type: string
                format: uri
                description: The URI of the resource being returned.
                readOnly: true
                writeOnly: false
              created:
                type: string
                format: date-time
                description: The date and time the resource was added
                             to the service provider.
                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.
                readOnly: true
                writeOnly: false
              version:
                type: string
                description: The version of the resource.
                readOnly: true
                writeOnly: false
            additionalProperties: false
<CODE ENDS>
```

C.2. EndpointApp Schema OpenAPI Representation

OpenAPI representation of endpointApp schema is as follows:

```
<CODE BEGINS>
openapi: 3.1.0
info:
  title: SCIM endpoint app schema
  version: 1.0.0

components:
  schemas:
    Group:
      type: object
      description: A list of groups to which the endpoint
                   application belongs, either through
                   direct membership, through nested
                   groups, or dynamically calculated.
      properties:
        value:
```

```
            type: string
            description: the unique identifier of a group,
                         typically a UUID.
            readOnly: true
            writeOnly: false
          display:
            type: string
            description: a display string for the group.
            readOnly: true
            writeOnly: false
          $ref:
            type: string
            format: uri
            description: reference to the group object
            readOnly: true
            writeOnly: true
      EndpointApp:
        title: EndpointApp
        description: Endpoint application resource
        type: object
        properties:
          applicationType:
            type: string
            description: This attribute will only contain two values;
                         'deviceControl' or 'telemetry'.
            readOnly: false
            writeOnly: false

          applicationName:
            type: string
            description: Human readable name of the application.
            readOnly: false
            writeOnly: false
          groups:
            type: array
            description: list of groups to which the endpointApp
                         belongs.
            items:
              $ref: '#/components/schemas/Group'

        required:
          - applicationType
          - applicationName

        additionalProperties: true
        oneOf:
          - $ref: '#/components/schemas/clientToken'
          - $ref: '#/components/schemas/certificateInfo'
```

```
        allOf:
          - $ref: '#/components/schemas/CommonAttributes'

      clientToken:
        type: string
        description: "This attribute contains a token that the client
                      will use to authenticate itself. Each token may
                      be a string up to 500 characters in length."
        readOnly: true
        writeOnly: false

      certificateInfo:
        type: object
        description: "Contains x509 certificate's subject name and
                      root CA information associated with the device
                      control or telemetry app."
        properties:
          rootCA:
            type: string
            description: "The base64 encoding of a trust anchor
                          certificate,as per RFC 4648 Section 4."
            readOnly: false
            writeOnly: false

          subjectName:
            type: string
            description: "Also known as the Common Name (CN), the
                          Subject Name is a field in the X.509
                          certificate that identifies the primary
                          domain or IP address for which the
                          certificate is issued."
            readOnly: false
            writeOnly: false

        required:
        - subjectName

      CommonAttributes:
        type: object
        properties:
          schemas:
            type: array
            items:
              type: string
              enum:
                - urn:ietf:params:scim:schemas:core:2.0:EndpointApp
            description: The list of schemas that define the resource.
          id:
```

```
            type: string
            format: uri
            description: The unique identifier for a resource.
            readOnly: true
            writeOnly: false
          meta:
            type: object
            readOnly: true
            properties:
              resourceType:
                type: string
                description: The name of the resource type of the
                             resource.
                readOnly: true
                writeOnly: false
              location:
                type: string
                format: uri
                description: The URI of the resource being returned.
                readOnly: true
                writeOnly: false
              created:
                type: string
                format: date-time
                description: The date and time the resource was added
                             to the service provider.
                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.
                readOnly: true
                writeOnly: false
              version:
                type: string
                description: The version of the resource.
                readOnly: true
                writeOnly: false
            additionalProperties: false
```

<CODE ENDS>

C.3. BLE Extension Schema OpenAPI Representation

OpenAPI representation of BLE extension schema is as follows:

```
<CODE BEGINS>
openapi: 3.1.0
info:
  title: SCIM Bluetooth Extension Schema
  version: 1.0.0

components:
  schemas:
    BleDevice:
      type: object
      description: BLE Device schema.
      properties:
        schemas:
          type: array
          items:
            type: string
            enum:
              - urn:ietf:params:scim:schemas:extension:ble:2.0
                 :Device
        urn:ietf:params:scim:schemas:extension:ble:2.0:Device:
          $ref: '#/components/schemas/BleDeviceExtension'
          required: true
    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].
          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}.
          readOnly: false
          writeOnly: false

        isRandom:
          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 a random address.
      readOnly: false
      writeOnly: false

    separateBroadcastAddress:
      type: string
      description: "When present, this address is used for
                    broadcasts/advertisements.  This value MUST
                        NOT
                    be set when an IRK is provided.  Its form is
                    the same as deviceMa‘cAddress."
      readOnly: false
      writeOnly: false

    irk:
      type: string
      description: Identity resolving key, which is unique for
                   every device. It is used to resolve random
                   address.
      readOnly: false
      writeOnly: true
    mobility:
      type: boolean
      description: If set to True, the BLE device will
                   automatically connect to the closest AP. For
                   example, BLE device is connected with AP-1
                       and
                   moves out of range but comes in range of AP
                      -2,
                   it will be disconnected with AP-1 and
                       connects
                   with AP-2.
      readOnly: false
      writeOnly: false
    pairingMethods:
      type: array
      items:
        type: string
      description: List of pairing methods associated with the
                   ble device, stored as schema URI.
      readOnly: false
      writeOnly: false
    urn:ietf:params:scim:schemas:extension:pairingNull:2.0
       :Device:
      $ref: ’#/components/schemas/NullPairing’
      required: false
    urn:ietf:params:scim:schemas:extension:pairingJustWorks:2.0
```

```
            :Device:
          $ref: '#/components/schemas/PairingJustWorks'
          required: false
        urn:ietf:params:scim:schemas:extension:pairingPassKey:2.0
           :Device:
          $ref: '#/components/schemas/PairingPassKey'
          required: false
        urn:ietf:params:scim:schemas:extension:pairingOOB:2.0
           :Device:
          $ref: '#/components/schemas/PairingOOB'
          required: false
      required:
        - versionSupport
        - deviceMacAddress
        - AddressType
        - pairingMethods
      additionalProperties: false

    NullPairing:
      type: object

    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'.
          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}$.
          readOnly: false
          writeOnly: true
      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.
            readOnly: false
            writeOnly: false
          randomNumber:
            type: integer
            description: Nonce added to the key
            readOnly: false
            writeOnly: true
          confirmationNumber:
            type: integer
            description: Some solutions require a confirmation number
                         in the RESTful message exchange.
            readOnly: false
            writeOnly: true
        required:
          - key
          - randomNumber
```

<CODE ENDS>

C.4. DPP Extension Schema OpenAPI Representation

OpenAPI representation of DPP extension schema is as follows:

<CODE BEGINS>

```
openapi: 3.1.0
info:
  title: SCIM Device Provisioning Protocol Extension Schema
  version: 1.0.0

components:
  schemas:
    DppDevice:
      type: object
      description: Wi-Fi Easy Connect (DPP) device extension schema
      properties:
        schemas:
          type: array
          items:
            type: string
            enum:
              - urn:ietf:params:scim:schemas:extension:dpp:2.0
                :Device
```

```
      urn:ietf:params:scim:schemas:extension:dpp:2.0:Device:
        $ref: '#/components/schemas/DppDeviceExtension'
        required: true
    DppDeviceExtension:
      type: object
      properties:
        dppVersion:
          type: integer
          description: Version of DPP this device supports.
          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].
          readOnly: false
          writeOnly: false
        bootstrapKey:
          type: string
          description: An 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.
          readOnly: false
          writeOnly: true
        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}.
          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'.
          readOnly: false
          writeOnly: false
        serialNumber:
          type: string
```

```
            description: An alphanumeric serial number that may also
                be
                        passed as bootstrapping information.
            readOnly: false
            writeOnly: false
          required:
            - dppVersion
            - bootstrapKey
          additionalProperties: false
   <CODE ENDS>
```

C.5. Ethernet MAB Extension Schema OpenAPI Representation

OpenAPI representation of Ethernet MAB extension schema is as follows:

```
<CODE BEGINS>
openapi: 3.1.0
info:
  title: SCIM MAC Authentication Bypass Extension Schema
  version: 1.0.0

components:
  schemas:
    EthernetMABDevice:
      type: object
      description: Ethernet MAC Authenticated Bypass
      properties:
        schemas:
          type: array
          items:
            type: string
            enum:
              - urn:ietf:params:scim:schemas:extension:ethernet-mab
                 :2.0:Device
        urn:ietf:params:scim:schemas:extension:ethernet-mab:2.0
           :Device:
          $ref: '#/components/schemas/EthernetMABDeviceExtension'
          required: true
    EthernetMABDeviceExtension:
      type: object
      properties:
        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}.
          readOnly: false
          writeOnly: false
      required:
        - deviceMacAddress
      description: Device extension schema for Ethernet-MAB
<CODE ENDS>
```

C.6. FDO Extension Schema OpenAPI Representation

OpenAPI representation of FDO extension schema is as follows:

```
<CODE BEGINS>
openapi: 3.1.0
info:
  title: SCIM Fido Device Onboarding Extension Schema
  version: 1.0.0

components:
  schemas:
    FDODevice:
      type: object
      description: FIDO Device Onboarding Extension
      properties:
        schemas:
          type: array
          items:
            type: string
            enum:
              - urn:ietf:params:scim:schemas:extension:fido-device
                 -onboard:2.0:Devices
        urn:ietf:params:scim:schemas:extension:fido-device-onboard
           :2.0:Devices:
          $ref: '#/components/schemas/FDODeviceExtension'
          required: true
    FDODeviceExtension:
      type: object
      properties:
        fdoVoucher:
          type: string
          description: A FIDO Device Onboard (FDO) Voucher
          readOnly: false
          writeOnly: false
      required:
        - fdoVoucher
      description: Device Extension for a FIDO Device Onboard (FDO)
<CODE ENDS>
```

C.7. Zigbee Extension Schema OpenAPI Representation

OpenAPI representation of zigbee extension schema is as follows:

```
<CODE BEGINS>
openapi: 3.1.0
info:
  title: SCIM Zigbee Extension Schema
  version: 1.0.0

components:
  schemas:
    ZigbeeDevice:
      type: object
      description: Zigbee Device schema.
      properties:
        schemas:
          type: array
          items:
            type: string
            enum:
              - urn:ietf:params:scim:schemas:extension:zigbee:2.0
                 :Device
        urn:ietf:params:scim:schemas:extension:zigbee:2.0:Device:
          $ref: '#/components/schemas/ZigbeeDeviceExtension'
          required: true
    ZigbeeDeviceExtension:
      type: object
      properties:
        versionSupport:
          type: array
          items:
            type: string
          description: Provides a list of all the Zigbee versions
                       supported by the device. For example, [3.0].
          readOnly: false
          writeOnly: false
        deviceEui64Address:
          type: string
          description: The EUI-64 (Extended Unique Identifier)
              device
                       address. The regex pattern is
                       ^[0-9A-Fa-f]{16}$.
          readOnly: false
          writeOnly: false
      required:
        - versionSupport
        - deviceEui64Address
      description: Device extension schema for Zigbee.
<CODE ENDS>
```

C.8. EndpointAppsExt Extension Schema OpenAPI Representation

OpenAPI representation of endpoint Apps extension schema is as follows:

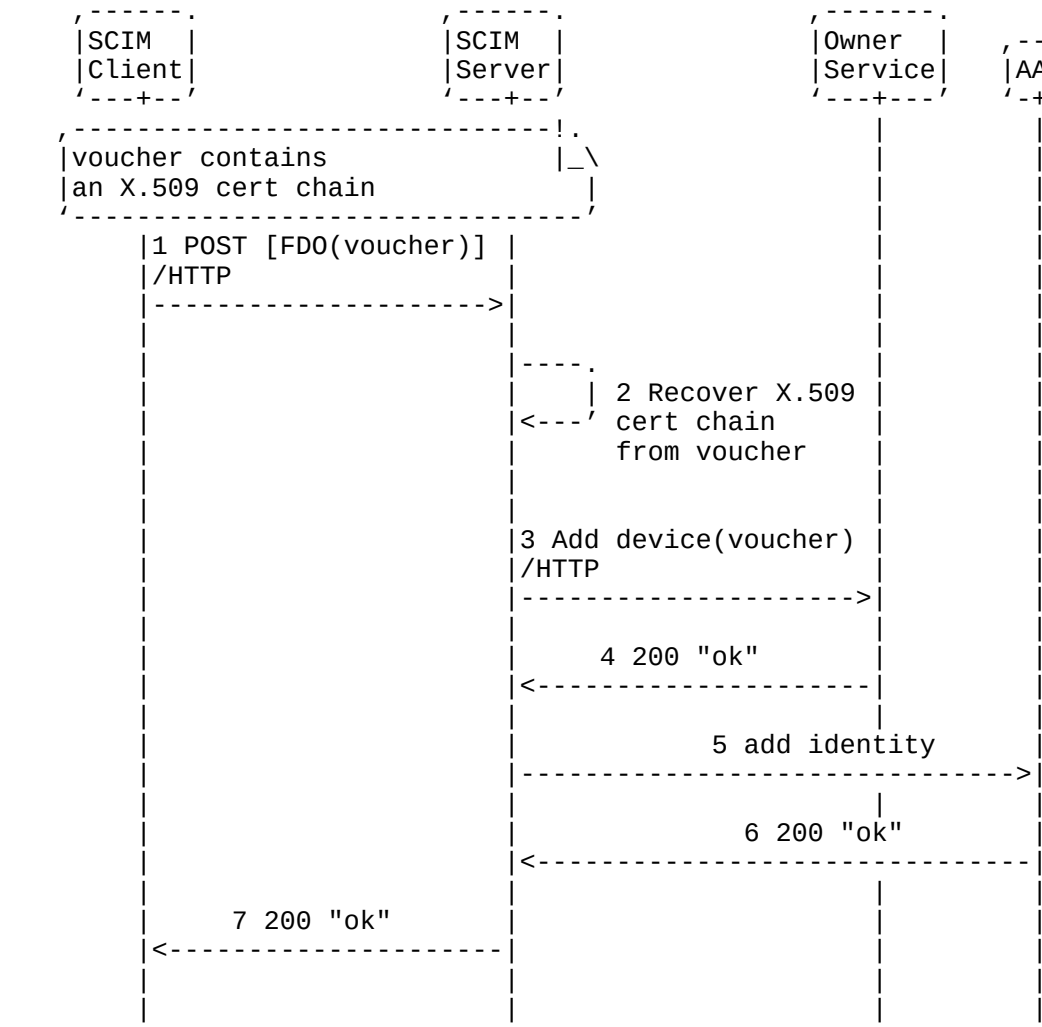
<CODE BEGINS>

```
openapi: 3.1.0
info:
  title: SCIM Endpoint extension schema
  version: 1.0.0


components:
  schemas:
    EndpointAppsExt:
      type: object
      properties:
        applications:
          $ref: '#/components/schemas/applications'

        deviceControlEnterpriseEndpoint:
          type: string
          format: url
          description: The URL of the enterprise endpoint which
              device
                       control apps use to reach enterprise network
                       gateway.
          readOnly: true
          writeOnly: false

        telemetryEnterpriseEndpoint:
          type: string
          format: url
          description: The URL of the enterprise endpoint which
                       telemetry apps use to reach enterprise
                           network
                       gateway.
          readOnly: true
          writeOnly: false

      required:
        - applications
        - deviceControlEnterpriseEndpoint

    applications:
      type: array
      items:
        value:
```

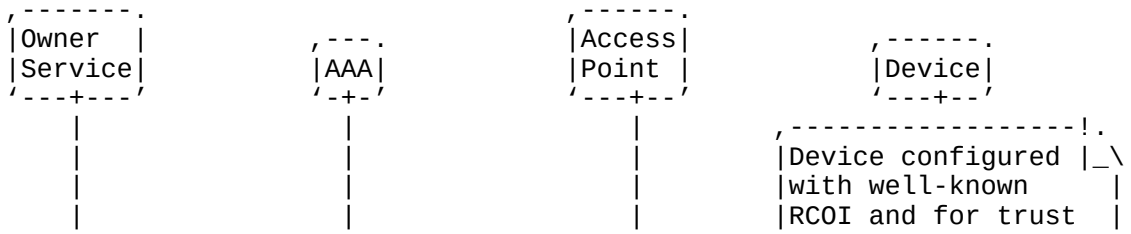
```
            type: string
            description: The identifier of the endpointApp.
            readOnly: false
            writeOnly: false
          ref:
            type: string
            format: uri
            description: The URI of the corresponding 'EndpointApp'
                         resource which will control or obtain data
                             from
                         the device.
            readOnly: true
            writeOnly: false
        required:
          - value
          - ref
   <CODE ENDS>
```

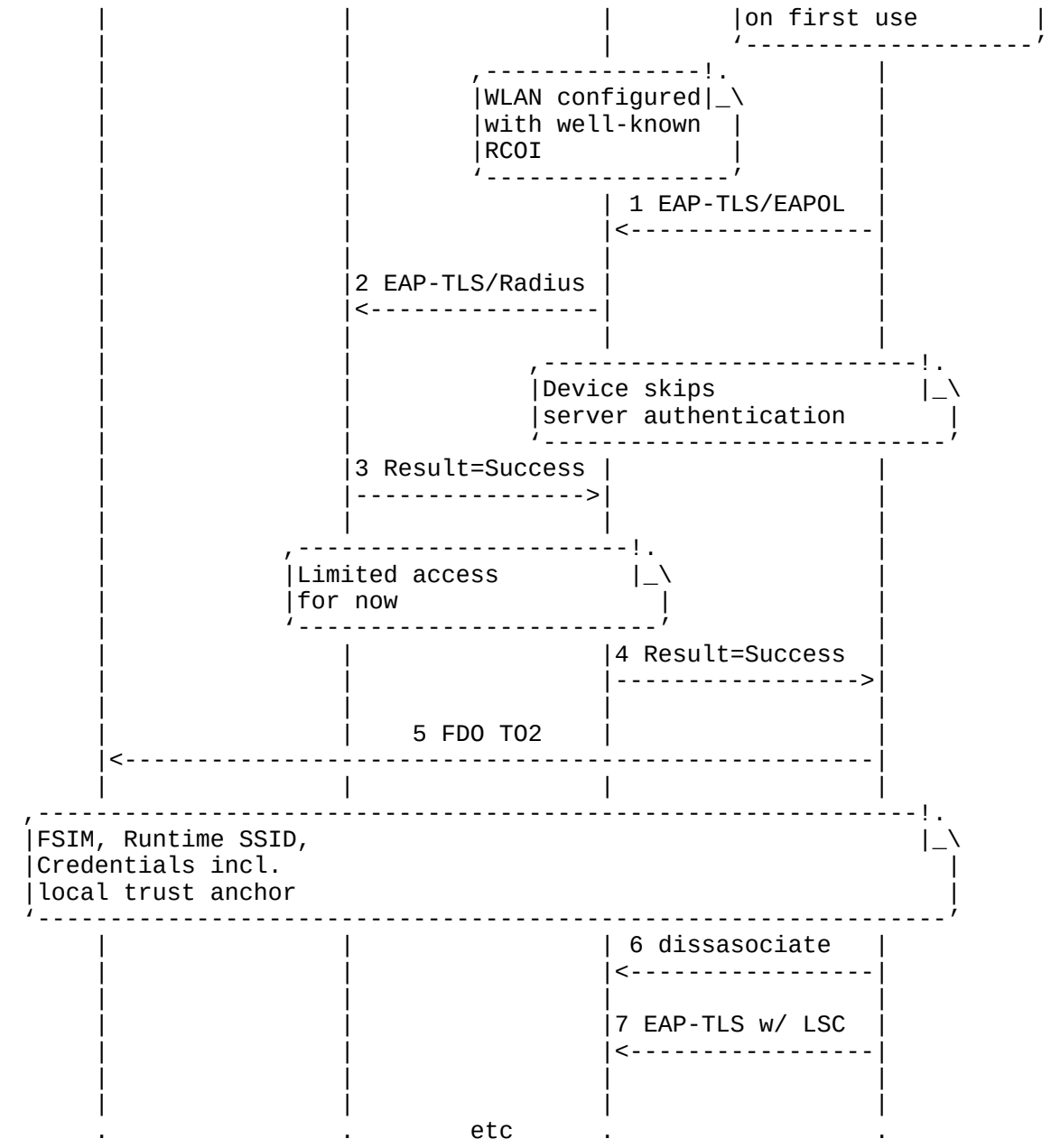
Appendix D. Fido Device Onboarding Example Flow

The following diagrams are included to demonstrate how FDO can be used. In this first diagram, a device is onboarded not only to the device owner process, but also to the AAA server for initial onboarding. The voucher contains a device certificate that is used by the AAA system for authentication.



After this flow is complete, the device can then first provisionally onboard, and then later receive a trust anchor through FDO's TO2 process. This is shown below.





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

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: hassaniqbal931@gmail.com

Eliot Lear
Cisco Systems
Richtistrasse 7
CH-8304 Wallisellen
Switzerland
Phone: +41 44 878 9200
Email: lear@cisco.com