A Semantic Definition Format for Data and Interactions of Things
Internet-Draft
Intended status: Standards Track
Expires: 5 June 2026

R. Mohan
B. Brinckman
Cisco Systems
L. Corneo
Ericsson

2 December 2025

Protocol Mapping for SDF
draft-ietf-asdf-sdf-protocol-mapping-02

Abstract

This document defines protocol mapping extensions for the Semantic Definition Format (SDF) to enable mapping of protocol-agnostic SDF affordances to protocol-specific operations. The protocol mapping mechanism allows SDF models to specify how properties, actions, and events should be accessed using specific IP and non-IP protocols such as Bluetooth Low Energy, Zigbee or HTTP and CoAP.

About This Document

This note is to be removed before publishing as an RFC.

Status information for this document may be found at https://datatracker.ietf.org/doc/draft-ietf-asdf-sdf-protocol-mapping/.

Discussion of this document takes place on the A Semantic Definition Format for Data and Interactions of Things Working Group mailing list (mailto:asdf@ietf.org), which is archived at https://mailarchive.ietf.org/arch/browse/asdf/. Subscribe at https://www.ietf.org/mailman/listinfo/asdf/.

Source for this draft and an issue tracker can be found at https://github.com/ietf-wg-asdf/sdf-protocol-mapping.

Status of This Memo

Copyright Notice

Table of Contents

1. Introduction

The Semantic Definition Format (SDF) [I-D.ietf-asdf-sdf] provides a protocol-agnostic way to describe IoT devices and their capabilities through properties, actions, and events (collectively called affordances). However, when implementing these affordances on actual devices using specific communication protocols, there needs to be a mechanism to map the protocol-agnostic SDF definitions to protocol-specific operations.

These protocols can be non-IP protocols that are commonly used in IoT environments, such as [BLE53] and [Zigbee22], or IP-based protocols, such as HTTP [RFC2616] or CoAP [RFC7252].

To leverage an SDF model to perform protocol-specific operations on an instance of a device, a mapping of the SDF affordance to a protocol-specific attribute is required. This document defines the protocol mapping mechanism using the sdfProtocolMap keyword, which allows SDF models to include protocol-specific mapping information alongside the protocol-agnostic definitions.

2. Conventions and Definitions

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.

3. Structure

Protocol mapping is required to map a protocol-agnostic affordance to a protocol-specific operation, as implementations of the same affordance will differ between protocols. For example, BLE will address a property as a service characteristic, while a property in Zigbee is addressed as an attribute in a cluster of an endpoint.

A protocol mapping object is a JSON object identified by the sdfProtocolMap keyword. Protocol-specific properties are embedded within this object, organized by protocol name, e.g., "ble" or "zigbee". The protocol name MUST be specified in the IANA registry requested in Section 8.1.

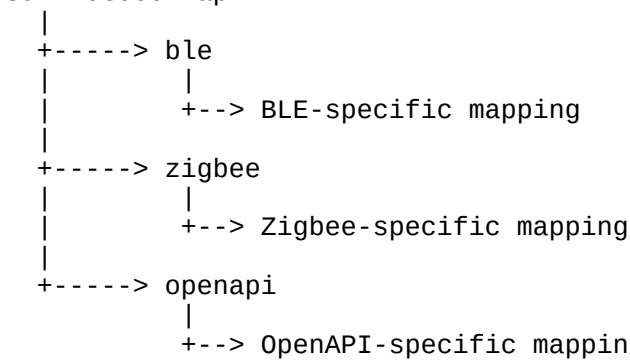

Figure 1: Property Mapping

As shown in Figure 1, protocol-specific properties must be embedded in an sdfProtocolMap object, for example a "ble" or a "zigbee" object.

Attribute	Type	Example
ble	object	an object with BLE-specific attributes
zigbee	object	an object with Zigbee-specific attributes
openapi	object	an object with OpenAPI-specific attributes

Table 1: Protocol objects

where-

* "ble" is an object containing properties that are specific to the BLE protocol.

* "zigbee" is an object containing properties that are specific to the Zigbee protocol.

* Other protocol mapping objects can be added by creating a new protocol object

Example protocol mapping:

```
{
  "sdfObject": {
    "healthsensor": {
      "sdfProperty": {
        "heartrate": {
          "description": "The current measured heart rate",
          "type": "number",
          "unit": "beat/min",
          "observable": false,
          "writable": false,
          "sdfProtocolMap": {
            "ble": {
              "serviceID": "12345678-1234-5678-1234-56789abcdef4",
              "characteristicID":
                "12345678-1234-5678-1234-56789abcdef4"
            }
          }
        }
      }
    }
  }
}
```

Figure 2: Example property mapping

For properties that have a different protocol mapping for read and write operations, the protocol mapping can be specified as such:

```
{
  "sdfObject": {
    "healthsensor": {
      "sdfProperty": {
        "heartrate": {
          "description": "The current measured heart rate",
          "type": "number",
          "unit": "beat/min",
          "observable": false,
          "sdfProtocolMap": {
            "ble": {
              "read": {
                "serviceID": "12345678-1234-5678-1234-56789abcdef4",
                "characteristicID":
                  "12345678-1234-5678-1234-56789abcdef5"
              },
              "write": {
                "serviceID": "12345678-1234-5678-1234-56789abcdef4",
                "characteristicID":
                  "12345678-1234-5678-1234-56789abcdef6"
              }
            }
          }
        }
      }
    }
  }
}
```

Figure 3: Example property mapping

4. Usage

A protocol map MAY be provided as part of the SDF model, specifically in the SDF affordance definition. The extension points in the SDF affordance definition defined in [I-D.ietf-asdf-sdf] are used to specify the protocol mapping information as a part of the SDF model.

For SDF properties, the protocol mapping is specified as an extension to a named property quality using the sdfProtocolMap keyword. For SDF actions and events, the protocol mapping can be specified as an extension to the named quality or as part of the sdfInputData or sdfOutputData objects.

5. Examples

5.1. BLE Protocol Mapping

The BLE protocol mapping allows SDF models to specify how properties, actions, and events should be accessed using Bluetooth Low Energy (BLE) protocol. The mapping includes details such as service IDs and characteristic IDs that are used to access the corresponding SDF affordances.

5.1.1. BLE Protocol Mapping Structure

For SDF properties and actions, the BLE protocol mapping structure is defined as follows:

```
$$SDF-PROTOCOL-MAP //= (
  ble: ble-protocol-map
)

ble-protocol-map = {
  ? ble-property,
  ? read: { ble-property },
  ? write: { ble-property }
}

ble-property = (
  serviceID: text,
  characteristicID: text
)
```

Figure 4: CDDL definition for BLE Protocol Mapping for properties and actions

Where:

* serviceID is the BLE service ID that corresponds to the SDF property or action.

* characteristicID is the BLE characteristic ID that corresponds to the SDF property or action.

For example, a BLE protocol mapping for a temperature property might look like:

```
{
  "sdfProperty": {
    "temperature": {
      "sdfProtocolMap": {
        "ble": {
          "serviceID": "12345678-1234-5678-1234-56789abcdef4",
          "characteristicID": "12345678-1234-5678-1234-56789abcdef5"
        }
      }
    }
  }
}
```

For a temperature property that has different mappings for read and write operations, the BLE protocol mapping might look like:

```
{
  "sdfProperty": {
    "temperature": {
      "sdfProtocolMap": {
        "ble": {
          "read": {
            "serviceID": "12345678-1234-5678-1234-56789abcdef4",
            "characteristicID": "12345678-1234-5678-1234-56789abcdef5"
          },
          "write": {
            "serviceID": "12345678-1234-5678-1234-56789abcdef4",
            "characteristicID": "12345678-1234-5678-1234-56789abcdef6"
          }
        }
      }
    }
  }
}
```

For SDF events, the BLE protocol mapping structure is similar, but it may include additional attributes such as the type of the event.

```
$$SDF-PROTOCOL-MAP //= (
  ble: ble-event-map
)

ble-event-map = {
  type: "gatt" / "advertisements" / "connection_events",
  ? serviceID: text,
  ? characteristicID: text
}
```

Figure 5: BLE Protocol Mapping for events

Where:

* type specifies the type of BLE event, such as "gatt" for GATT events, "advertisements" for advertisement events, or "connection_events" for connection-related events.

* serviceID and characteristicID are optional attributes that are specified if the type is "gatt".

For example, a BLE event mapping for a heart rate measurement event might look like:

```
{
  "sdfEvent": {
    "heartRate": {
      "sdfOutputData": {
        "sdfProtocolMap": {
          "ble": {
            "type": "gatt",
            "serviceID": "12345678-1234-5678-1234-56789abcdef4",
            "characteristicID": "12345678-1234-5678-1234-56789abcdef5"
          }
        }
      }
    }
  }
}
```

Another example of an isPresent event using BLE advertisements:

```
{
  "sdfEvent": {
    "isPresent": {
      "sdfOutputData": {
        "sdfProtocolMap": {
          "ble": {
            "type": "advertisements"
          }
        }
      }
    }
  }
}
```

5.2. Zigbee Protocol Mapping

The Zigbee protocol mapping allows SDF models to specify how properties, actions, and events should be accessed using the Zigbee protocol. The mapping includes details such as cluster IDs and attribute IDs that are used to access the corresponding SDF affordances.

5.2.1. Zigbee Protocol Mapping Structure

For SDF properties and events, the Zigbee protocol mapping structure is defined as follows:

```
$$SDF-PROTOCOL-MAP //= (
  zigbee: zigbee-protocol-map
)

zigbee-protocol-map = {
  ? zigbee-property,
  ? read: { zigbee-property },
  ? write: { zigbee-property }
}

zigbee-property = (
  endpointID: uint,
  manufacturerCode: uint,
  clusterID: uint,
  attributeID: uint,
  type: uint
)
```

Figure 6: CDDL definition for Zigbee Protocol Mapping for properties and actions

Where:

* endpointID is the Zigbee endpoint ID that corresponds to the SDF affordance.

* clusterID is the Zigbee cluster ID that corresponds to the SDF affordance.

* attributeID is the Zigbee attribute ID that corresponds to the SDF affordance.

* type is the Zigbee data type of the attribute.

SDF properties are mapped to Zigbee cluster attributes and events are mapped to Zigbee cluster attribute reporting.

For example, a Zigbee protocol mapping for a temperature property might look like:

```
{
  "sdfProperty": {
    "temperature": {
      "sdfProtocolMap": {
        "zigbee": {
          "endpointID": 1,
          "clusterID": 1026, // 0x0402
          "attributeID": 0, // 0x0000
          "type": 41 // 0x29
        }
      }
    }
  }
}
```

SDF actions are mapped to Zigbee cluster commands. The Zigbee protocol mapping structure for actions is defined as follows:

```
$$SDF-PROTOCOL-MAP //= (
  zigbee: zigbee-action-map
)

zigbee-action-map = {
  endpointID: uint,
  manufacturerCode: uint,
  clusterID: uint,
  commandID: uint,
}
```

Figure 7: CDDL definition for Zigbee Protocol Mapping for actions

Where:

* endpointID is the Zigbee endpoint ID that corresponds to the SDF action.

* clusterID is the Zigbee cluster ID that corresponds to the SDF action.

* commandID is the Zigbee command ID that corresponds to the SDF action.

For example, a Zigbee protocol mapping to set a temperature might look like:

```
{
  "sdfAction": {
    "setTemperature": {
      "sdfProtocolMap": {
        "zigbee": {
          "endpointID": 1,
          "clusterID": 1026, // 0x0402
          "commandID": 0 // 0x0000
        }
      }
    }
  }
}
```

5.3. IP based Protocol Mapping

The protocol mapping mechanism can potentially also be used for IP-based protocols such as HTTP or CoAP.

In the case of HTTP, SDF protocol mappings towards an SDF quality MAY be expressed by directly pointing to OpenAPI schema and/or component.

```
$$SDF-PROTOCOL-MAP //= (
  openapi: openapi-protocol-map
)

openapi-protocol-map = {
  ? openapi-property,
  ? read: { openapi-property },
  ? write: { openapi-property }
}

openapi-property = (
  operationRef: text,
  $ref: text
)
```

An example of a protocol mapping for a property using HTTP might look like:

```
================ NOTE: '\' line wrapping per RFC 8792 ================

{
  "sdfProperty": {
    "heartrate": {
      "sdfProtocolMap": {
        "openapi": {
            "operationRef": "https://example.com/openapi.json#/paths\
/~1heartrate~1{id}~1current",
            "$ref": "https://example.com/openapi.json#/components/sc\
hema/HeartRate/properties/pulse"
        }
      }
    }
  }
}
```

The operationRef points to the OpenAPI operation that retrieves the current heart rate, and the $ref points to the OpenAPI schema that defines the heart rate property. An example of the OpenAPI schema might look like:

```
paths:
  /heartrate/{id}/current:
    get:
      summary: Get current heart rate
      description: |-
        Retrieve the current heart rate for a specific user
        identified by {id}.
      parameters:
        - name: id
          in: path
          required: true
          description: |-
            The ID of the user whose heart rate is being queried.
          schema:
            type: string
      responses:
        "200":
          description: |-
            Successful response with current heart rate data.
          content:
            application/json:
              schema:
                $ref: "#/components/schemas/HeartRate"
    put:
      summary: Set current heart rate
      description: |-
```

```
          Set the current heart rate for a specific user
          identified by {id}.
        parameters:
          - name: id
            in: path
            required: true
            description: |-
              The ID of the user whose heart rate is being set.
            schema:
              type: string
        requestBody:
          required: true
          content:
            application/json:
              schema:
                $ref: "#/components/schemas/HeartRate"

  components:
    schemas:
      HeartRate:
        type: object
        properties:
          pulse:
            type: integer
            description: The current heart rate in beats per minute.
          spo2:
            type: number
            format: float
            description: |-
              The current body temperature in degrees Celsius.
```

We assume that the readable properties will map to GET operations and the writable properties will map to PUT operations. If this is not the case, or if the API is different, the protocol mapping can be specified as such:

```
================ NOTE: '\' line wrapping per RFC 8792 ================

{
  "sdfProperty": {
    "heartrate": {
      "sdfProtocolMap": {
        "openapi": {
          "read": {
            "operationRef": "https://example.com/openapi.json#/paths\
/~1heartrate~1{id}~1current/get",
            "$ref": "https://example.com/openapi.json#/components/sc\
hema/HeartRate/properties/pulse"
          },
          "write": {
            "operationRef": "https://example.com/openapi.json#/paths\
/~1heartrate~1{id}~1current/put",
            "$ref": "https://example.com/openapi.json#/components/sc\
hema/HeartRate/properties/pulse"
          }
        }
      }
    }
  }
}
```

6. SCIM SDF Extension

While SDF provides a way to describe a device, a method is needed to associate a mapping between an instance of a device and its associated SDF models. To accomplish this, we define a SCIM extension that can be used in conjunction with [I-D.ietf-scim-device-model] in Figure 8. Implementation of this SCIM extension is OPTIONAL and independent of the protocol mapping functionality defined in the rest of this document. The SCIM schema attributes used here are described in Section 7 of [RFC7643].

```
================ NOTE: '\' line wrapping per RFC 8792 ================

{
    "id": "urn:ietf:params:scim:schemas:extension:sdf:2.0:Device",
    "name": "SDFExtension",
    "description": "Device extension schema for SDF.",
    "attributes": [
        {
            "name": "sdf",
            "type": "string",
            "description": "SDF models supported by the device.",
            "multiValued": true,
            "required": true,
            "caseExact": true,
            "mutability": "readWrite",
            "returned": "default",
            "uniqueness": "none"
        }
    ],
    "meta": {
        "resourceType": "Schema",
        "location": "/v2/Schemas/urn:ietf:params:scim:schemas:extens\
ion:sdf:2.0:Device"
    }
}
```

Figure 8: SCIM SDF Extension Schema

An example SCIM device schema extension might look like:

```
{
    "schemas": [
        "urn:ietf:params:scim:schemas:core:2.0:Device",
        "urn:ietf:params:scim:schemas:extension:sdf:2.0:Device"
    ],
    "id": "e9e30dba-f08f-4109-8486-d5c6a3316111",
    "displayName": "Heart Monitor",
    "active": true,
    "urn:ietf:params:scim:schemas:extension:sdf:2.0:Device": {
        "sdf": [
            "https://example.com/thermometer#/sdfThing/thermometer",
            "https://example.com/heartrate#/sdfObject/healthsensor"
        ]
    }
}
```

7. Security Considerations

TODO Security

8. IANA Considerations

This section provides guidance to the Internet Assigned Numbers Authority (IANA) regarding registration of values related to this document, in accordance with [RFC8126].

8.1. Protocol Mapping

IANA is requested to create a new registry called "SDF Protocol Mapping".

The registry must contain the following attributes:

* Protocol map name
* Protocol name
* Description
* Reference of the specification describing the protocol mapping. This specification must be reviewed by an expert.

The registrant of an existing entry may request updates to that entry, subject to the same expert review. They should verify that updates preserve backward compatibility with deployed implementations, or if breaking changes are necessary, consider whether a new registry entry is more appropriate.

Following protocol mappings are described in this document:

Protocol map	Protocol Name	Description	Reference
ble	Bluetooth Low Energy (BLE)	Protocol mapping for BLE devices	This document
zigbee	Zigbee	Protocol mapping for Zigbee devices	This document
openapi	OpenAPI	Protocol mapping for OpenAPI	This document

Table 2: Protocol Mapping Registry

8.2. SCIM Device Schema SDF Extension

IANA is requested to create the following extensions in the SCIM Server-Related Schema URIs registry as described in Section 6:

URN	Description	Resource Type	Reference
urn:ietf:params:scim: schemas:extension: sdf:2.0:Device	SDF Extension	Device	This memo, Section 6

Table 3

9. References

9.1. Normative References

[I-D.ietf-asdf-sdf]
Koster, M., Bormann, C., and A. Ker辰nen, "Semantic Definition Format (SDF) for Data and Interactions of Things", Work in Progress, Internet-Draft, draft-ietf-asdf-sdf-25, 13 October 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-asdf-sdf-25>.

[I-D.ietf-scim-device-model]
Shahzad, M., Iqbal, H., and E. Lear, "Device Schema Extensions to the SCIM model", Work in Progress, Internet-Draft, draft-ietf-scim-device-model-18, 3 September 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-scim-device-model-18>.

[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/rfc/rfc2119>.

[RFC7643] Hunt, P., Ed., Grizzle, K., Wahlstroem, E., and C. Mortimore, "System for Cross-domain Identity Management: Core Schema", RFC 7643, DOI 10.17487/RFC7643, September 2015, <https://www.rfc-editor.org/rfc/rfc7643>.

[RFC8126] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/rfc/rfc8126>.

[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/rfc/rfc8174>.

9.2. Informative References

[BLE53] Bluetooth SIG, "Bluetooth Core Specification Version 5.3", 13 July 2021, <https://www.bluetooth.com/specifications/specs/core-specification-5-3/>.

[RFC2616] Fielding, R., Gettys, J., Mogul, J., Frystyk, H., Masinter, L., Leach, P., and T. Berners-Lee, "Hypertext Transfer Protocol -- HTTP/1.1", RFC 2616, DOI 10.17487/RFC2616, June 1999, <https://www.rfc-editor.org/rfc/rfc2616>.

[RFC7252] Shelby, Z., Hartke, K., and C. Bormann, "The Constrained Application Protocol (CoAP)", RFC 7252, DOI 10.17487/RFC7252, June 2014, <https://www.rfc-editor.org/rfc/rfc7252>.

[Zigbee22] Zigbee Alliance, "Zigbee 3.0 Specification", 2022, <https://zigbeealliance.org/solution/zigbee/>.

Appendix A. CDDL Definition

This appendix contains the combined CDDL definitions for the SDF protocol mappings.

```
<CODE BEGINS> file "sdf-protocol-map.cddl"
$$SDF-EXTENSION-DATA //= (
  sdfProtocolMap: {
    * $$SDF-PROTOCOL-MAP
  }
)

$$SDF-PROTOCOL-MAP //= (
  ble: ble-protocol-map
)

ble-protocol-map = {
  ? ble-property,
```

```
    ? read: { ble-property },
    ? write: { ble-property }
  }

  ble-property = (
    serviceID: text,
    characteristicID: text
  )

  $$SDF-PROTOCOL-MAP //= (
    ble: ble-event-map
  )

  ble-event-map = {
    type: "gatt" / "advertisements" / "connection_events",
    ? serviceID: text,
    ? characteristicID: text
  }

  $$SDF-PROTOCOL-MAP //= (
    zigbee: zigbee-protocol-map
  )

  zigbee-protocol-map = {
    ? zigbee-property,
    ? read: { zigbee-property },
    ? write: { zigbee-property }
  }

  zigbee-property = (
    endpointID: uint,
    manufacturerCode: uint,
    clusterID: uint,
    attributeID: uint,
    type: uint
  )

  $$SDF-PROTOCOL-MAP //= (
    zigbee: zigbee-action-map
  )

  zigbee-action-map = {
    endpointID: uint,
    manufacturerCode: uint,
    clusterID: uint,
    commandID: uint,
  }
  <CODE ENDS>
```

Appendix B. OpenAPI Definition

The following non-normative model is provided for convenience of the implementor.

<CODE BEGINS> file "ProtocolMap.yaml"
=============== NOTE: '\' line wrapping per RFC 8792 ================

```
openapi: 3.0.3
info:
  title: SDF Protocol Mapping
  description: |-
    SDF Protocol Mapping. When adding a
    new protocol mapping please add a reference to the protocol map
    for all the schemas in this file.
  version: 0.10.0
externalDocs:
  description: SDF Protocol Mapping IETF draft
  url: https://datatracker.ietf.org/doc/draft-ietf-asdf-sdf-protocol\
-mapping/

paths: {}

components:
  schemas:
# Protocol Mapping
## Protocol Map for Service Discovery
    ProtocolMap-ServiceList:
      type: object
      properties:
        sdfProtocolMap:
          oneOf:
            - $ref: './ProtocolMap-BLE.yaml#/components/schemas/Prot\
ocolMap-BLE-ServiceList'

## Protocol Map for Service Discovery result
    ProtocolMap-ServiceMap:
      type: object
      properties:
        sdfProtocolMap:
          oneOf:
            - $ref: './ProtocolMap-BLE.yaml#/components/schemas/Prot\
ocolMap-BLE-ServiceMap'
            - $ref: './ProtocolMap-Zigbee.yaml#/components/schemas/P\
rotocolMap-Zigbee-ServiceMap'

## Protocol Map for Broadcasts
    ProtocolMap-Broadcast:
```

```
        type: object
        properties:
          sdfProtocolMap:
            oneOf:
              - $ref: './ProtocolMap-BLE.yaml#/components/schemas/Prot\
   ocolMap-BLE-Broadcast'
              - $ref: './ProtocolMap-Zigbee.yaml#/components/schemas/P\
   rotocolMap-Zigbee-Broadcast'

   ## Protocol Map for a property
       ProtocolMap-Property:
         type: object
         properties:
           sdfProtocolMap:
             oneOf:
               - $ref: './ProtocolMap-BLE.yaml#/components/schemas/Prot\
   ocolMap-BLE-Propmap'
               - $ref: './ProtocolMap-Zigbee.yaml#/components/schemas/P\
   rotocolMap-Zigbee-Propmap'

   ## Protocol Map for an event
       ProtocolMap-Event:
         type: object
         properties:
           sdfProtocolMap:
             oneOf:
               - $ref: './ProtocolMap-BLE.yaml#/components/schemas/Prot\
   ocolMap-BLE-Event'
               - $ref: './ProtocolMap-Zigbee.yaml#/components/schemas/P\
   rotocolMap-Zigbee-Event'
   <CODE ENDS>
```

Figure 9

B.1. Protocol map for BLE

```
   <CODE BEGINS> file "ProtocolMap-BLE.yaml"
   =============== NOTE: '\' line wrapping per RFC 8792 ================

   openapi: 3.0.3
   info:
     title: SDF Protocol Mapping for BLE
     description: |-
       SDF Protocol Mapping for BLE devices.
     version: 0.10.0
   externalDocs:
     description: SDF Protocol Mapping IETF draft
     url: https://datatracker.ietf.org/doc/draft-ietf-asdf-sdf-protocol\
```

```
   -mapping/

   paths: {}

   components:
     schemas:
   # BLE Protocol Mapping
   ## A Service is a device with optional service IDs
       ProtocolMap-BLE-ServiceList:
         type: object
         properties:
           ble:
             type: object
             properties:
               services:
                 type: array
                 items:
                   type: object
                   properties:
                     serviceID:
                       type: string
                       format: uuid
                       example: 00001809-0000-1000-8000-00805f9b34fb
               cached:
                 description: |-
                   If we can cache information, then device doesn’t need
                   to be rediscovered before every connected.
                 type: boolean
                 default: false
               cacheIdlePurge:
                 description: cache expiry period, when device allows
                 type: integer
                 example: 3600 # default 1 hour
               autoUpdate:
                 description: |-
                   autoupdate services if device supports it (default)
                 type: boolean
                 example: true
               bonding: #optional, by default defined in SCIM object
                 type: string
                 example: default
                 enum:
                   - default
                   - none
                   - justworks
                   - passkey
                   - oob
```

```
   ##  Protocol Mapping for BLE Service Map
       ProtocolMap-BLE-ServiceMap:
         required:
           - services
         type: object
         properties:
           ble:
             type: array
             items:
               $ref: '#/components/schemas/ProtocolMap-BLE-Service'

       ProtocolMap-BLE-Service:
         required:
           - serviceID
           - characteristics
         type: object
         properties:
           serviceID:
             type: string
             format: uuid
             example: 00001809-0000-1000-8000-00805f9b34fb
           characteristics:
             type: array
             items:
               $ref: '#/components/schemas/ProtocolMap-BLE-Characterist\
   ic'

       ProtocolMap-BLE-Characteristic:
         required:
           - characteristicID
           - flags
           - descriptors
         type: object
         properties:
           characteristicID:
             type: string
             format: uuid
             example: 00002a1c-0000-1000-8000-00805f9b34fb
           flags:
             type: array
             example:
             - read
             - write
             items:
               type: string
               enum:
                 - read
                 - write
```

```
              - notify
          descriptors:
            type: array
            items:
              $ref: '#/components/schemas/ProtocolMap-BLE-Descriptor'

      ProtocolMap-BLE-Descriptor:
        required:
          - descriptorID
        type: object
        properties:
          descriptorID:
            type: string
            format: uuid
            example: 00002902-0000-1000-8000-00805f9b34fb

  ##  Protocol Mapping for BLE Broadcast
      ProtocolMap-BLE-Broadcast:
        required:
          - ble
        type: object
        properties:
          ble:
            type: object
            properties:
              connectable:
                type: boolean

  ## Protocol Mapping for BLE Property
      ProtocolMap-BLE-Propmap:
        required:
          - ble
        type: object
        properties:
          ble:
            required:
              - serviceID
              - characteristicID
            type: object
            properties:
              serviceID:
                type: string
                format: uuid
                example: 00001809-0000-1000-8000-00805f9b34fb
              characteristicID:
                type: string
                format: uuid
                example: 00002a1c-0000-1000-8000-00805f9b34fb
```

```
   ## Defines different types of BLE events
       ProtocolMap-BLE-Event:
         required:
           - ble
         type: object
         properties:
           ble:
             required:
               - type
             type: object
             properties:
               type:
                 type: string
                 example: gatt
                 enum:
                   - gatt
                   - connection_events
                   - advertisements
               serviceID:
                 type: string
                 example: 00001809-0000-1000-8000-00805f9b34fb
               characteristicID:
                 type: string
                 example: 00002a1c-0000-1000-8000-00805f9b34fb
   <CODE ENDS>
```

Figure 10

B.2. Protocol map for Zigbee

```
   <CODE BEGINS> file "ProtocolMap-Zigbee.yaml"
   =============== NOTE: '\' line wrapping per RFC 8792 ================

   openapi: 3.0.3
   info:
     title: SDF Protocol Mapping for Zigbee
     description: |-
       SDF Protocol Mapping for Zigbee devices.
     version: 0.10.0
   externalDocs:
     description: SDF Protocol Mapping IETF draft
     url: https://datatracker.ietf.org/doc/draft-ietf-asdf-sdf-protocol\
   -mapping/

   paths: {}

   components:
     schemas:
```

```
   # Zigbee Protocol Mapping
   ##  Protocol Mapping for Zigbee Service Map
       ProtocolMap-Zigbee-ServiceMap:
         required:
           - zigbee
         type: object
         properties:
           zigbee:
             type: array
             items:
               $ref: '#/components/schemas/ProtocolMap-Zigbee-Endpoint'

       ProtocolMap-Zigbee-Endpoint:
         required:
           - endpointID
           - clusters
         type: object
         properties:
           endpointID:
             type: integer
             format: int32
             example: 10
           clusters:
             type: array
             items:
               $ref: '#/components/schemas/ProtocolMap-Zigbee-Cluster'

       ProtocolMap-Zigbee-Cluster:
         required:
           - clusterID
           - properties
         type: object
         properties:
           clusterID:
             type: integer
             format: int32
             example: 0
           properties:
             type: array
             items:
               $ref: '#/components/schemas/ProtocolMap-Zigbee-Property'

       ProtocolMap-Zigbee-Property:
         required:
           - attributeID
           - propertyType
         type: object
         properties:
```

```
          attributeID:
            type: integer
            format: int32
            example: 1
          propertyType:
            type: integer
            format: int32
            example: 32

   ## Protocol Mapping for Zigbee broadcast
       ProtocolMap-Zigbee-Broadcast:
         required:
           - zigbee
         type: object
         properties:
           zigbee:
             type: object

   ## Protocol mapping for Zigbee property
       ProtocolMap-Zigbee-Propmap:
         required:
           - zigbee
         type: object
         properties:
           zigbee:
             required:
               - endpointID
               - clusterID
               - attributeID
             type: object
             properties:
               endpointID:
                 type: integer
                 format: int32
                 example: 1
               clusterID:
                 type: integer
                 format: int32
                 example: 6
               attributeID:
                 type: integer
                 format: int32
                 example: 16
               type:
                 type: integer
                 format: int32
                 example: 1
```

```
      ProtocolMap-Zigbee-Event:
        allOf:
          - $ref: '#/components/schemas/ProtocolMap-Zigbee-Propmap'
  <CODE ENDS>
```

Figure 11

Acknowledgments

TODO acknowledge.

Authors' Addresses

Rohit Mohan
Cisco Systems
170 West Tasman Drive
San Jose, 95134
United States of America
Email: rohitmo@cisco.com

Bart Brinckman
Cisco Systems
170 West Tasman Drive
San Jose, 95134
United States of America
Email: bbrinckm@cisco.com

Lorenzo Corneo
Ericsson
Hirsalantie 11
FI-1296 Jorvas
Finland
Email: lorenzo.corneo@ericsson.com