A Semantic Definition Format for Data and Interactions of Things
Internet-Draft
Intended status: Standards Track
Expires: 8 November 2026

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

7 May 2026

SDF Protocol Mapping
draft-ietf-asdf-sdf-protocol-mapping-08

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 non-IP and IP protocols such as Bluetooth Low Energy, Zigbee or HTTP and CoAP. This document also describes a method to extend SCIM with an SDF model mapping.

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) [RFC9880] provides a protocol-agnostic way to describe IoT devices and their capabilities through properties, actions, and events (collectively called affordances). When implementing an SDF model for a device using specific communication protocols, there needs to be a mechanism to map the protocol-agnostic SDF definitions to protocol-specific operations, translating the model into a real-world implementation. Moreover, such a mechanism needs to be extensible for enabling implementers to provide novel SDF protocol mappings to expand the SDF ecosystem. SDF protocol mappings may target a variety of protocols spanning from non-IP protocols commonly used in IoT environments, such as [BLE53] and [Zigbee30], to IP-based protocols such as HTTP [RFC9110] and CoAP [RFC7252]. This document provides the required mechanism by defining:

* The sdfProtocolMap keyword, which allows SDF models to include protocol-specific mapping information attached to the protocol-agnostic definitions, see Section 3. An sdfProtocolMap MAY be applied to an SDF affordance, be it an sdfProperty, sdfEvent or sdfAction. The mapping enables use cases such as application gateways or multi-protocol gateways that translate between different IoT protocols, automated generation of protocol-specific implementations from SDF models, and interoperability across heterogeneous device ecosystems.

* Two SDF protocol mappings for Bluetooth and Zigbee protocols, see Section 5.1 and Section 5.2 respectively.

* An SDF model extension for SCIM. While SDF provides a way to describe a class of devices, SCIM describes a device instance. The SDF model extension for SCIM enables the inclusion of SDF models for the class of devices a device belongs to in the SCIM object, see Section 6.

* An IANA registry for defining additional SDF protocol mappings (in addition to the BLE and Zigbee provided in this document), see Section 8.1.

2. Conventions and Definitions

The syntax extensions to [RFC9880] in terms of its JSON structures are shown in the Concise Data Definition Language (CDDL) [RFC8610].

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. SDF Protocol Mapping Structure

This section defines the structure of an sdfProtocolMap. Because each protocol has its own addressing model, a single SDF affordance requires a distinct mapping per protocol. For example, BLE addresses a property as a service characteristic, while Zigbee addresses it as an attribute in a cluster of an endpoint.

A protocol mapping object is a JSON object identified by the sdfProtocolMap keyword, nested inside an SDF affordance definition (sdfProperty, sdfAction, or sdfEvent). Protocol-specific attributes are embedded within this object, keyed by a protocol name registered in the IANA "SDF Protocol Mapping" registry (Section 8.1), e.g., "ble" or "zigbee".

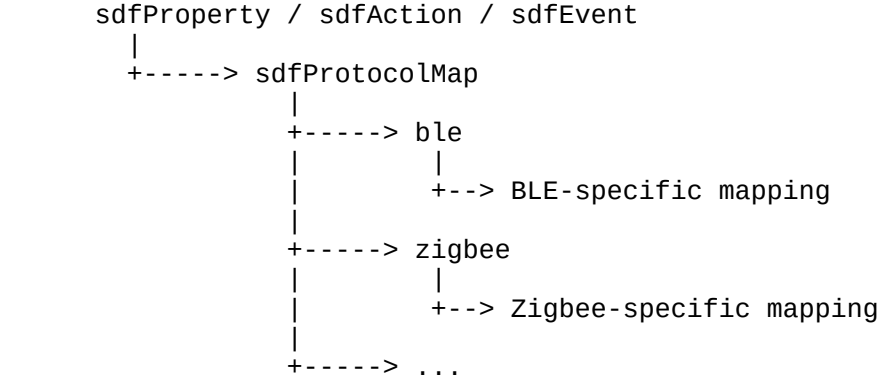

Figure 1: SDF Protocol Mapping Structure

3.1. SDF Extension Points

The sdfProtocolMap keyword is introduced into SDF affordance definitions through the extension points defined in the formal syntax of Appendix A of [RFC9880]. For each affordance type, an sdfProtocolMap entry is added via the corresponding CDDL group socket. The contents of the sdfProtocolMap object are in turn extensible through a protocol-mapping-specific group socket.

A protocol MAY choose to extend only the affordance types that are applicable to it. For example, the BLE protocol mapping defines extensions for properties and events but not for actions.

3.1.1. Property Extension

The $$SDF-EXTENSION-PROPERTY group socket in the propertyqualities rule of Appendix A of [RFC9880] is used to add protocol mapping to sdfProperty definitions:

```
$$SDF-EXTENSION-PROPERTY //= (
  sdfProtocolMap: {
    * $$SDF-PROPERTY-PROTOCOL-MAP
  }
)

property-protocol-map<name, props, read-props, write-props> = (
  name => props /
    {
      read: read-props,
      write: write-props
    }
)
```

Figure 2: SDF Property Extension Point for Protocol Mapping

The property-protocol-map generic (Figure 2) captures the common structure of property protocol mappings. The name parameter is the protocol name and props is the protocol-specific map of attributes. A protocol can provide either:

* A single mapping that applies to both read and write operations, or

* Separate read and write mappings when the protocol uses different attributes for each direction.

To extend $$SDF-PROPERTY-PROTOCOL-MAP for a new protocol (e.g., "new-protocol"), implementers MUST use the property-protocol-map generic with the protocol name and a map type defining the protocol-specific attributes.

It is to be noted that the protocol name (e.g., "new-protocol") MUST be registered in the IANA registry defined in Section 8.1.

For example:

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

$$SDF-PROPERTY-PROTOCOL-MAP //= (
  property-protocol-map<"new-protocol", new-protocol-property, new-\
                            protocol-property, new-protocol-property>
)

new-protocol-property = {
  attributeA: text,
  attributeB: uint
}
```

Figure 3: Example Property Protocol Map Extension

The corresponding JSON in an SDF model looks like:

```
{
  "sdfProperty": {
    "temperature": {
      "type": "number",
      "unit": "Cel",
      "sdfProtocolMap": {
        "new-protocol": {
          "attributeA": "temperature-service",
          "attributeB": 1
        }
      }
    }
  }
}
```

Figure 4: Example Property Protocol Map in JSON

When a property uses different protocol attributes for read and write operations, the mapping can be split:

```
{
  "sdfProperty": {
    "temperature": {
      "type": "number",
      "unit": "Cel",
      "sdfProtocolMap": {
        "new-protocol": {
          "read": {
            "attributeA": "temperature-read-service",
            "attributeB": 1
          },
          "write": {
            "attributeA": "temperature-write-service",
            "attributeB": 2
          }
        }
      }
    }
  }
}
```

Figure 5: Example Property Protocol Map with Read/Write in JSON

3.1.2. Action Extension

The $$SDF-EXTENSION-ACTION group socket in the actionqualities rule of Appendix A of [RFC9880] is used to add protocol mapping to sdfAction definitions:

```
$$SDF-EXTENSION-ACTION //= (
  sdfProtocolMap: {
    * $$SDF-ACTION-PROTOCOL-MAP
  }
)
```

Figure 6: SDF Action Extension Point for Protocol Mapping

Actions use a simpler structure than properties, as they do not require the read/write distinction. To extend $$SDF-ACTION-PROTOCOL-MAP for a new protocol, implementers MUST add a group entry that maps the protocol name to the protocol-specific attributes:

```
$$SDF-ACTION-PROTOCOL-MAP //= (
  "new-protocol": new-protocol-action
)

new-protocol-action = {
  commandID: uint
}
```

Figure 7: Example Action Protocol Map Extension

The corresponding JSON in an SDF model would look like:

```
{
  "sdfAction": {
    "reset": {
      "sdfProtocolMap": {
        "new-protocol": {
          "commandID": 42
        }
      }
    }
  }
}
```

Figure 8: Example Action Protocol Map in JSON

3.1.3. Event Extension

The $$SDF-EXTENSION-EVENT group socket in the eventqualities rule of Appendix A of [RFC9880] is used to add protocol mapping to sdfEvent definitions:

```
$$SDF-EXTENSION-EVENT //= (
  sdfProtocolMap: {
    * $$SDF-EVENT-PROTOCOL-MAP
  }
)
```

Figure 9: SDF Event Extension Point for Protocol Mapping

Events follow the same simple pattern as actions. To extend $$SDF-EVENT-PROTOCOL-MAP for a new protocol:

```
$$SDF-EVENT-PROTOCOL-MAP //= (
  "new-protocol": new-protocol-event
)

new-protocol-event = {
  eventID: uint
}
```

Figure 10: Example Event Protocol Map Extension

The corresponding JSON in an SDF model looks like:

```
{
  "sdfEvent": {
    "alert": {
      "sdfProtocolMap": {
        "new-protocol": {
          "eventID": 3
        }
      }
    }
  }
}
```

Figure 11: Example Event Protocol Map in JSON

4. New Protocol Registration Procedure

Protocol names used as keys in the sdfProtocolMap object (e.g., "ble", "zigbee") MUST be registered in the IANA registry defined in Section 8.1.

A new SDF protocol mapping MUST be defined by a specification that mandatorily includes:

* A CDDL definition that extends at least one of the group sockets defined in this document: $$SDF-PROPERTY-PROTOCOL-MAP (Section 3.1.1), $$SDF-ACTION-PROTOCOL-MAP (Section 3.1.2), or $$SDF-EVENT-PROTOCOL-MAP (Section 3.1.3). Property mappings MUST use the property-protocol-map generic (Section 3.1.1) to ensure a consistent structure.

* A description of the protocol-specific attributes introduced by the CDDL extension, including their semantics and how they relate to the underlying protocol operations.

5. Registered Protocol Mappings

This section defines the protocol mappings registered by this document.

5.1. BLE

The BLE protocol mapping allows SDF models to specify how properties and events should be accessed using Bluetooth Low Energy (BLE) protocol [BLE53]. The mapping includes details such as service IDs and characteristic IDs that are used to access the corresponding SDF affordances.

5.1.1. Properties

For sdfProperty, the BLE protocol mapping structure is defined as follows:

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

$$SDF-PROPERTY-PROTOCOL-MAP //= (
  property-protocol-map<"ble", ble-property, ble-property, ble-\
                                                          property>
)

ble-property = {
  serviceID: text,
  characteristicID: text
}
```

Figure 12: CDDL definition for BLE Protocol Mapping for sdfProperty

Where:

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

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

For example, a BLE protocol mapping for a temperature property:

```
{
  "sdfProperty": {
    "temperature": {
      "sdfProtocolMap": {
        "ble": {
          "serviceID": "00001809-0000-1000-8000-00805f9b34fb",
          "characteristicID": "00002a1c-0000-1000-8000-00805f9b34fb"
        }
      }
    }
  }
}
```

For a temperature property that has different mappings for read and write operations, here is an example of the BLE protocol mapping:

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

```
{
  "sdfProperty": {
    "temperature": {
      "sdfProtocolMap": {
        "ble": {
          "read": {
            "serviceID": "00001809-0000-1000-8000-00805f9b34fb",
            "characteristicID": "00002a1c-0000-1000-8000-\
                                                  00805f9b34fb"
          },
          "write": {
            "serviceID": "00001809-0000-1000-8000-00805f9b34fb",
            "characteristicID": "00002a51-0000-1000-8000-\
                                                  00805f9b34fb"
          }
        }
      }
    }
  }
}
```

5.1.2. Events

For sdfEvents, the BLE protocol mapping structure is similar to sdfProperties, but it MUST include additional attributes such as the type of the event.

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

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

Figure 13: BLE Protocol Mapping for sdfEvents

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:

```
{
  "sdfEvent": {
    "heartRate": {
      "sdfProtocolMap": {
        "ble": {
          "type": "gatt",
          "serviceID": "0000180d-0000-1000-8000-00805f9b34fb",
          "characteristicID": "00002a37-0000-1000-8000-00805f9b34fb"
        }
      }
    }
  }
}
```

Here is an example of an isPresent event using BLE advertisements:

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

5.2. Zigbee

The Zigbee protocol mapping allows SDF models to specify how properties, actions, and events should be accessed using the Zigbee protocol [Zigbee30]. The mapping includes details such as cluster IDs and attribute IDs that are used to access the corresponding SDF affordances.

5.2.1. Properties

An sdfProperty is mapped to a Zigbee cluster attribute. The Zigbee property protocol mapping structure is defined as follows:

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

$$SDF-PROPERTY-PROTOCOL-MAP //= (
  property-protocol-map<"zigbee", zigbee-property, zigbee-property, \
                                                        zigbee-property>
)

zigbee-property = {
  endpointID: uint,
  clusterID: uint,
  attributeID: uint,
  attributeType: uint,
  ? manufacturerCode: uint,
}
```

Figure 14: CDDL definition for Zigbee Protocol Mapping for sdfProperty

Where:

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

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

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

* attributeType is the Zigbee data type of the attribute.

* manufacturerCode is the Zigbee manufacturer code of the attribute (optional).

For example, a Zigbee protocol mapping for a temperature property may look as follows:

```
{
  "sdfProperty": {
    "temperature": {
      "sdfProtocolMap": {
        "zigbee": {
          "endpointID": 1,
          "clusterID": 1026,
          "attributeID": 0,
          "attributeType": 41
        }
      }
    }
  }
}
```

5.2.2. Events

An sdfEvent is mapped to Zigbee cluster attribute reporting. The Zigbee event protocol mapping structure is defined as follows:

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

zigbee-event-map = {
  endpointID: uint,
  clusterID: uint,
  attributeID: uint,
  attributeType: uint,
  ? manufacturerCode: uint,
}
```

Figure 15: CDDL definition for Zigbee Protocol Mapping for sdfEvents

Where:

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

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

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

* attributeType is the Zigbee data type of the attribute.

* manufacturerCode is the Zigbee manufacturer code of the attribute (optional).

For example, a Zigbee event mapping for a temperature change report:

```
{
  "sdfEvent": {
    "temperatureChange": {
      "sdfProtocolMap": {
        "zigbee": {
          "endpointID": 1,
          "clusterID": 1026,
          "attributeID": 0,
          "attributeType": 41
        }
      }
    }
  }
}
```

5.2.3. Actions

An sdfAction is mapped to a Zigbee cluster command. The Zigbee protocol mapping structure for actions is defined as follows:

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

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

Figure 16: CDDL definition for Zigbee Protocol Mapping for sdfAction

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.

* manufacturerCode is the Zigbee manufacturer code of the command (optional).

For example, a Zigbee protocol mapping to set a temperature:

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

6. SCIM SDF Extension

While SDF provides a way to describe a device class and SCIM defines a device instance, a method is needed to associate a mapping between an instance of a device and its associated SDF models. To accomplish so, this document defines a SCIM extension that MAY be used in conjunction with [I-D.ietf-scim-device-model] in Figure 17. 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:\
                                              extension:sdf:2.0:Device"
    }
}
```

Figure 17: SCIM SDF Extension Schema

Here is an example SCIM device schema extension with SDF models:

```
{
    "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"
        ]
    }
}
```

An SDF model MUST be referenced with the sdf keyword inside the SCIM device schema as described in [I-D.ietf-scim-device-model].

7. Security Considerations

The security considerations of [RFC9880] apply to this document as well.

Each protocol mapped using this mechanism has its own security model. The protocol mapping mechanism defined in this document does not provide additional security beyond what is offered by the underlying protocols. Implementations MUST ensure that appropriate protocol-level security mechanisms are employed when accessing affordances through the mapped protocol operations.

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 registration policy for this registry is "Specification Required" as defined in Section 4.6 of [RFC8126].

The registry must contain the following attributes:

* Protocol map name, as per sdfProtocolMap
* Protocol name
* Description
* Reference of the specification describing the protocol mapping.

The specification requirements for a registration request are defined in Section 4.

The designated expert(s) MUST verify that the protocol map name is appropriate and not likely to cause confusion with existing entries.

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.

The following protocol mappings are described in this document:

Protocol Map Name	Protocol Name	Description	Reference
ble	Bluetooth Low Energy (BLE)	Protocol mapping for BLE devices	This document, Section 5.1
zigbee	Zigbee	Protocol mapping for Zigbee devices	This document, Section 5.2

Table 1: Protocol Mapping Registry

8.2. SCIM Device Schema SDF Extension

IANA is requested to create the following extension 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 document, Section 6

Table 2: SCIM Device Schema SDF Extension

9. References

9.1. Normative References

[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>.

[RFC8610] Birkholz, H., Vigano, C., and C. Bormann, "Concise Data Definition Language (CDDL): A Notational Convention to Express Concise Binary Object Representation (CBOR) and JSON Data Structures", RFC 8610, DOI 10.17487/RFC8610, June 2019, <https://www.rfc-editor.org/rfc/rfc8610>.

[RFC9880] Koster, M., Ed., Bormann, C., Ed., and A. Ker辰nen, "Semantic Definition Format (SDF) for Data and Interactions of Things", RFC 9880, DOI 10.17487/RFC9880, January 2026, <https://www.rfc-editor.org/rfc/rfc9880>.

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/>.

[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>.

[RFC9110] Fielding, R., Ed., Nottingham, M., Ed., and J. Reschke, Ed., "HTTP Semantics", STD 97, RFC 9110, DOI 10.17487/RFC9110, June 2022, <https://www.rfc-editor.org/rfc/rfc9110>.

[Zigbee30] CSA IoT, "Zigbee 3.0 Specification", 2026, <https://csa-iot.org/all-solutions/zigbee/>.

Appendix A. CDDL Definition

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

<CODE BEGINS> file "sdf-protocol-map.cddl"

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

$$SDF-EXTENSION-PROPERTY //= (
  sdfProtocolMap: {
    * $$SDF-PROPERTY-PROTOCOL-MAP
  }
)

property-protocol-map<name, props, read-props, write-props> = (
  name => props /
    {
      read: read-props,
      write: write-props
    }
)

$$SDF-EXTENSION-ACTION //= (
  sdfProtocolMap: {
    * $$SDF-ACTION-PROTOCOL-MAP
  }
)

$$SDF-EXTENSION-EVENT //= (
  sdfProtocolMap: {
    * $$SDF-EVENT-PROTOCOL-MAP
  }
)

$$SDF-PROPERTY-PROTOCOL-MAP //= (
  property-protocol-map<"ble", ble-property, ble-property, ble-\
                                                          property>
)

ble-property = {
  serviceID: text,
  characteristicID: text
}

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

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

   $$SDF-PROPERTY-PROTOCOL-MAP //= (
     property-protocol-map<"zigbee", zigbee-property, zigbee-property, \
                                                        zigbee-property>
   )

   zigbee-property = {
     endpointID: uint,
     clusterID: uint,
     attributeID: uint,
     attributeType: uint,
     ? manufacturerCode: uint,
   }

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

   zigbee-event-map = {
     endpointID: uint,
     clusterID: uint,
     attributeID: uint,
     attributeType: uint,
     ? manufacturerCode: uint,
   }

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

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

Figure 18: CDDL for SDF protocol mappings

Appendix B. OpenAPI Definition

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

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

## Protocol Map for an event
    ProtocolMap-Event:
      type: object
      properties:
        sdfProtocolMap:
          oneOf:
            - $ref: './ProtocolMap-BLE.yaml#/components/schemas/\
                                               ProtocolMap-BLE-Event'
            - $ref: './ProtocolMap-Zigbee.yaml#/components/schemas/\
                                             ProtocolMap-Zigbee-Event'
```

<CODE ENDS>

Figure 19: OpenAPI model

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:
## 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:
             oneOf:
               - type: object
                 required:
                   - type
                   - serviceID
                   - characteristicID
                 properties:
                   type:
                     type: string
                     example: gatt
                     enum:
                       - gatt
                   serviceID:
                     type: string
                     example: 00001809-0000-1000-8000-00805f9b34fb
                   characteristicID:
                     type: string
                     example: 00002a1c-0000-1000-8000-00805f9b34fb
               - type: object
                 required:
                   - type
                 properties:
                   type:
                     type: string
                     example: advertisements
                     enum:
                       - advertisements
               - type: object
                 required:
                   - type
                 properties:
                   type:
                     type: string
                     example: connection_events
                     enum:
                       - connection_events
   <CODE ENDS>
```

Figure 20: OpenAPI model for BLE

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:
   ## 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 21: OpenAPI model for Zigbee

Acknowledgements

This document relies on SDF models described in [RFC9880], as such, we are grateful to the authors of this document for putting their time and effort into defining SDF in depth, allowing us to make use of it. The authors would also like to thank the ASDF working group for their excellent feedback and steering of the document.

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