

A Semantic Definition Format for Data and Interactions of ThingsR. Mohan
Internet-Draft
Intended status: Standards Track
Expires: 19 April 2026

B. Brinckman
Cisco Systems
L. Corneo
Ericsson
16 October 2025

Protocol Mapping for SDF
draft-ietf-asdf-sdf-protocol-mapping-01

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

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

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

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

This Internet-Draft will expire on 19 April 2026.

Copyright Notice

Copyright (c) 2025 IETF Trust and the persons identified as the document authors. All rights reserved.

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

Table of Contents

1. Introduction	3
2. Conventions and Definitions	3
3. Structure	3
4. Usage	6
5. Examples	6
5.1. BLE Protocol Mapping	7
5.1.1. BLE Protocol Mapping Structure	7
5.2. Zigbee Protocol Mapping	10
5.2.1. Zigbee Protocol Mapping Structure	10
5.3. IP based Protocol Mapping	12
6. SCIM SDF Extension	15
7. Security Considerations	16
8. IANA Considerations	16
8.1. Protocol mapping	16
8.2. SCIM Device Schema SDF Extension	17
9. References	17
9.1. Normative References	17
9.2. Informative References	18
Appendix A. CDDL Definition	18
Appendix B. SCIM SDF Extension Schema	20
Acknowledgments	20
Authors' Addresses	20

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.

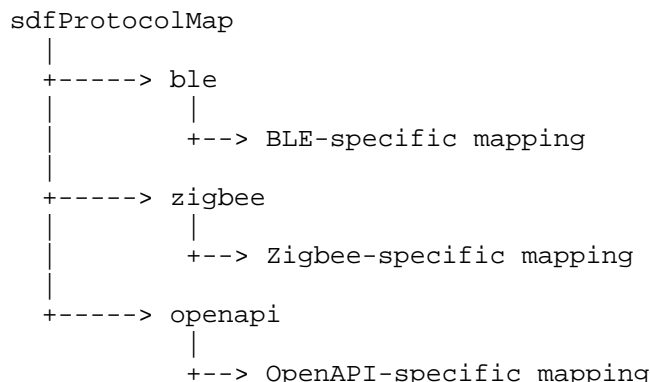


Figure 1: Property Mapping

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], the details of which can be found in Appendix B.

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 following attributes:

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

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. Keränen, "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>>.
- [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>>.

- [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/>>.
- [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>>.
- [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

```
$$SDF-EXTENSION-DATA //= (  
  sdfProtocolMap: {  
    * $$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,
```

```
}
```

Appendix B. SCIM SDF Extension Schema

===== 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 global names supported by the device\
.",
      "multiValued": true,
      "required": true,
      "caseExact": false,
      "mutability": "readWrite",
      "returned": "default",
      "uniqueness": "none"
    }
  ],
  "meta": {
    "resourceType": "Schema",
    "location": "/v2/Schemas/urn:ietf:params:scim:schemas:extens\
ion:sdf:2.0:Device"
  }
}
```

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