

ASDF
Internet-Draft
Intended status: Standards Track
Expires: 21 January 2026

C. Bormann, Ed.
Universität Bremen TZI
J. Romann
Universität Bremen
20 July 2025

Semantic Definition Format (SDF): Mapping files
draft-bormann-asdf-sdf-mapping-07

Abstract

The Semantic Definition Format (SDF) is a format for domain experts to use in the creation and maintenance of data and interaction models that describe Things, i.e., physical objects that are available for interaction over a network. It was created as a common language for use in the development of the One Data Model liaison organization (OneDM) models. Tools convert this format to database formats and other serializations as needed.

An SDF specification often needs to be augmented by additional information that is specific to its use in a particular ecosystem or application. SDF mapping files provide a mechanism to represent this augmentation.

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-bormann-asdf-sdf-mapping/>.

Discussion of this document takes place on the A Semantic Definition Format for Data and Interactions of Things (asdf) 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/cabo/sdf-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 21 January 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
1.1. Terminology and Conventions	3
2. Overview	3
2.1. Example Model 1 (ecosystem: IPSO/OMA)	4
2.2. Example Model 2 (ecosystem: W3C WoT)	4
3. Formal Syntax of SDF mapping files	8
4. Augmentation Mechanism	10
4.1. Logging Augmentation	13
5. IANA Considerations	14
5.1. Media Type	14
5.2. Registries	14
6. Security Considerations	15
7. References	15
7.1. Normative References	15
7.2. Informative References	15
List of Figures	16
List of Tables	16
Acknowledgements	16
Authors' Addresses	16

1. Introduction

The Semantic Definition Format (SDF) is a format for domain experts to use in the creation and maintenance of data and interaction models that describe Things, i.e., physical objects that are available for interaction over a network. It was created as a common language for use in the development of the One Data Model liaison organization (OneDM) models. Tools convert this format to database formats and other serializations as needed.

An SDF specification often needs to be augmented by additional information that is specific to its use in a particular ecosystem or application. SDF mapping files provide a mechanism to represent this augmentation.

1.1. Terminology and Conventions

The definitions of [I-D.ietf-asdf-sdf] apply.

The term "byte" is used in its now-customary sense as a synonym for "octet".

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 [BCP14] (RFC2119) (RFC8174) when, and only when, they appear in all capitals, as shown here.

2. Overview

An SDF mapping file provides augmentation information for one or more SDF models. Its main contents is a map from SDF name references (Section 4.3 of [I-D.ietf-asdf-sdf]) to a set of qualities.

When processing the mapping file together with one or more SDF models, these qualities are added to the SDF model at the referenced name, as in a merge-patch operation [RFC7396]. Note that this is somewhat similar to the way sdfRef (Section 4.4 of [I-D.ietf-asdf-sdf]) works, but in a mapping file the arrows point in the inverse direction (from the augments to the augmented).

2.1. Example Model 1 (ecosystem: IPSO/OMA)

An example for an SDF mapping file is given in Figure 1. This mapping file is meant to attach to an SDF specification published by OneDM, and to add qualities relevant to the IPSO/OMA ecosystem.

```
// Note that this example uses namespaces to identify elements of
// the referenced specification(s), but has un-namespaced quality
// names. These two kinds of namespaces are unrelated in SDF, and a
// more robust example may need to make use of Quality Name Prefixes
// as defined in Section 2.3.3-3 of [I-D.ietf-asdf-sdf] (independent
// of a potential feature to add namespace references to definitions
// that are not intended to go into the default namespace — these are
// SDF definition namespaces and not quality namespaces, which are
// one meta-level higher).
```

* Start of a mapping file for certain OneDM playground models:

```
{
  "info": {
    "title": "IPSO ID mapping"
  },
  "namespace": {
    "onedm": "https://onedm.org/models"
  },
  "defaultNamespace": "onedm",
  "map": {
    "#/sdfObject/Digital_Input": {
      "id": 3200
    },
    "#/sdfObject/Digital_Input/sdfProperty/Digital_Input_State": {
      "id": 5500
    },
    "#/sdfObject/Digital_Input/sdfProperty/Digital_Input_Counter": {
      "id": 5501
    }
  }
}
```

Figure 1: A simple example of an SDF mapping file

2.2. Example Model 2 (ecosystem: W3C WoT)

This example shows a translation of a hypothetical W3C WoT Thing Model (as defined in Section 10 of [W3C.wot-thing-description11]) into an SDF model plus a mapping file to catch Thing Model attributes that don't currently have SDF qualities defined (namely, titles and descriptions members used for internationalization).

A second mapping file is more experimental in that it captures information that is actually instance-specific, in this case a forms member that binds the status property to an instance-specific CoAP resource.

// Namespaces are all wrong in this example.

The form really should be part of the class level; defining the entire form instead of just the link in the instance information is a symptom of not yet getting the class/instance boundary right.

* The input: WoT Thing Model

```
{
  "@context": "https://www.w3.org/2022/wot/td/v1.1",
  "@type" : "tm:ThingModel",
  "title": "Lamp Thing Model",
  "titles": {
    "en": "Lamp Thing Model",
    "de": "Thing Model fr eine Lampe"
  },
  "properties": {
    "status": {
      "description": "Current status of the lamp",
      "descriptions": {
        "en": "Current status of the lamp",
        "de": "Aktueller Status der Lampe"
      },
      "type": "string",
      "readOnly": true,
      "forms": [
        {
          "href": "coap://example.org/status"
        }
      ]
    }
  }
}
```

Figure 2: Input: WoT Thing Model

* The output: SDF model

```
{
  "info": {
    "title": "Lamp Thing Model"
  },
  "namespace": {
    "wot": "http://www.w3.org/ns/td"
  },
  "defaultNamespace": "wot",
  "sdfObject": {
    "LampThingModel": {
      "label": "Lamp Thing Model",
      "sdfProperty": {
        "status": {
          "description": "Current status of the lamp",
          "writable": false,
          "type": "string"
        }
      }
    }
  }
}
```

Figure 3: Output 1: SDF Model

* The other output: SDF mapping file for class information

```

{
  "info": {
    "title": "Lamp Thing Model: WoT TM mapping"
  },
  "namespace": {
    "wot": "http://www.w3.org/ns/td"
  },
  "defaultNamespace": "wot",
  "map": {
    "#/sdfObject/LampThingModel": {
      "titles": {
        "en": "Lamp Thing Model",
        "de": "Thing Model fr eine Lampe"
      }
    },
    "#/sdfObject/LampThingModel/sdfProperty/status": {
      "descriptions": {
        "en": "Current status of the lamp",
        "de": "Aktueller Status der Lampe"
      }
    }
  }
}

```

Figure 4: Output 2: SDF Mapping File

* A third output: SDF mapping file for Protocol Bindings

```

{
  "info": {
    "title": "Lamp Thing Model: WoT TM Protocol Binding"
  },
  "namespace": {
    "wot": "http://www.w3.org/ns/td"
  },
  "defaultNamespace": "wot",
  "map": {
    "#/sdfObject/LampThingModel/sdfProperty/status": {
      "forms": [
        {
          "href": "coap://example.org/status"
        }
      ]
    }
  }
}

```

Figure 5: Output 3: SDF Mapping File for Protocol Bindings

3. Formal Syntax of SDF mapping files

An SDF mapping file has three optional components that are taken unchanged from SDF: The info block, the namespace declaration, and the default namespace. The mandatory fourth component, the "map", contains the mappings from an SDF name reference (usually a namespace and a JSON pointer) to a nested map providing a set of qualities to be merged in at the site identified in the name reference.

Figure 6 describes the syntax of SDF mapping files using CDDL [RFC8610].

```
start = sdf-mapping

sdf-mapping = {
  ; info will be required in most process policies
  ? info: sdfinfo
  ? namespace: named<text>
  ? defaultNamespace: text
  map: { * global-sdf-pointer => additionalqualities}
}

; we can't really be much more specific here:
additionalqualities = named<any>

; ----- import from SDF-base:

sdfinfo = {
  ? title: text
  ? description: text
  ? version: text
  ? copyright: text
  ? license: text
  ? modified: modified-date-time
  ? features: [
    * (any .feature "feature-name") ; EXTENSION-POINT
  ]
  optional-comment
  EXTENSION-POINT<"info-ext">
}

; Shortcut for a map that gives names to instances of X
; (has keys of type text and values of type X)
named<X> = { * text => X }

; EXTENSION-POINT is only used in framework syntax
EXTENSION-POINT<f> = ( * (quality-name .feature f) => any )
quality-name = text .regexp "([a-z][a-z0-9]*:)?[a-z$][A-Za-z$0-9]*"
```

```

; rough CURIE or JSON Pointer syntax:
global-sdf-pointer = text .regexp ".*[:#].*"

optional-comment = (
  ? $comment: text      ; source code comments only, no semantics
)

modified-date-time = text .abnf modified-dt-abnf
modified-dt-abnf = "modified-dt" .det rfc3339z

; RFC 3339 sans time-numoffset, slightly condensed
rfc3339z = '
  date-fullyear    = 4DIGIT
  date-month       = 2DIGIT ; 01-12
  date-mday        = 2DIGIT ; 01-28, 01-29, 01-30, 01-31 based on
                        ; month/year
  time-hour        = 2DIGIT ; 00-23
  time-minute      = 2DIGIT ; 00-59
  time-second      = 2DIGIT ; 00-58, 00-59, 00-60 based on leap sec
                        ; rules
  time-secfrac     = "." 1*DIGIT
  DIGIT            = %x30-39 ; 0-9

  partial-time     = time-hour ":" time-minute ":" time-second
                    [time-secfrac]
  full-date        = date-fullyear "-" date-month "-" date-mday

  modified-dt      = full-date ["T" partial-time "Z"]
,
```

Figure 6: CDDL definition of SDF mapping file

The JSON pointer that is used on the left-hand side of the map can point to a JSON map in the SDF model to be augmented by adding or replacing map entries. If necessary, the JSON map is created at the position indicated with the contents of right-hand side of the map // (add examples). Alternatively, the JSON pointer can point to an array (also possibly created if not existing before) and add an element to that array by using the "-" syntax introduced in the penultimate paragraph of Section 4 of [RFC6901].

4. Augmentation Mechanism

An SDF model and a compatible mapping file can be combined to create an `_augmented_` SDF model. (This process can be repeated with multiple mapping files by using the outcome of one augmentation as the input of the next one.) As augmentation is not equal to instantiation, augmented SDF models are still abstract in nature, but are enriched with ecosystem-specific information.

| Note that it might be necessary to specify an augmentation
| mechanism for instance descriptions as well at a later point in
| time, once it has been decided what the instance description
| format might look like and whether such a format is needed.

The augmentation mechanism is related to the resolution mechanism defined in Section 4.4 of [I-D.ietf-asdf-sdf], but fundamentally different:

Instead of a model file reaching out to other model files and integrating aspects into itself via `sdfRef` (`_pull_` approach), the mapping file `_pushes_` information into a new copy of a specific given SDF model. The original SDF model does not need to know which mapping files it will be used with and can be used with several such mapping files independently of each other.

An augmented SDF model is produced from two inputs: An SDF model and a compatible mapping file, i.e. every JSON pointer within the keys of the mapping file's map object points to a location that already exists within the SDF model. To perform the augmentation, a processor needs to create a copy of the original SDF model. It then iterates over all entries within the mapping file's map object [in an order to be specified; probably lexicographical]. During each iteration, the processor first obtains a reference to the target referred to by the JSON pointer contained an entry's key. This reference is then used as the Target argument of the JSON Merge Patch algorithm [RFC7396] and the entry's value as the Patch argument; the target is replaced with the result of the merge-patch.

Once the iteration has finished, the processor returns the resulting augmented SDF model. Should the resolution of a JSON pointer or an application of the JSON Merge Patch algorithm fail, an error is thrown instead.

Note that, in contrast to an array, the entries of a JSON object are considered unordered, which means that the sequence in which the map entries are applied is implementation-dependent. For this reason, we need to make sure that the contents of mapping files can be applied independently of each other. (We need to understand the onus in ensuring this that is put on author of a mapping file.)

The problem can be "avoided" by changing the data structure used by the map quality to an array of objects to ensure a deterministic application order. This would essentially put most of the onus on the author of a mapping file to get any order dependencies right. More preferable would be to ensure the entries in the mapping file can be applied independently and in any order. More discussion and implementation experience is required.

An example for an augmented SDF model can be seen in Figure 7. This is the result of applying the WoT-specific mapping file from Figure 4 to the SDF model shown in Figure 3. This augmented SDF model is one step away from being converted to a WoT Thing Model or Thing Description, which requires some information that cannot be provided in an SDF model that is limited to the vocabulary defined in the SDF base specification.

```

{
  "info": {
    "title": "Lamp Thing Model"
  },
  "namespace": {
    "wot": "http://www.w3.org/ns/td"
  },
  "defaultNamespace": "wot",
  "sdfObject": {
    "LampThingModel": {
      "label": "Lamp Thing Model",
      "titles": {
        "en": "Lamp Thing Model",
        "de": "Thing Model fr eine Lampe"
      },
      "sdfProperty": {
        "status": {
          "description": "Current status of the lamp",
          "descriptions": {
            "en": "Current status of the lamp",
            "de": "Aktueller Status der Lampe"
          },
          "writable": false,
          "type": "string"
        }
      }
    }
  }
}

```

Figure 7: An SDF model that has been augmented with WoT-specific vocabulary.

Since the pair of an SDF model and a mapping file is equivalent in semantics to the augmented model created from the two, there is no fundamental difference between specifying aspects in the SDF model or leaving them in a mapping file. Also, parts of an ecosystem-specific vocabulary may in fact be mappable to the SDF base vocabulary. Therefore, developing the mapping between SDF and an ecosystem requires careful consideration which of the features should be available to other ecosystems and therefore should best be part of the common SDF model, and which are best handled in a mapping file specific to the ecosystem.

4.1. Logging Augmentation

Since an augmented model is not fundamentally different from any other SDF model, it may be necessary to trace the provenance of the information that flowed into it, e.g., in the info block. For this purpose, a new quality called `augmentationLog` is introduced that contains an array of URIs pointing to the mapping files that have been used to augment the original SDF file (which can also be indicated via the `originalSdfModel` quality). These additional qualities allow for reproducing the augmentation process.

For logging while performing an augmentation, the processor has to perform the following steps:

1. If the info block is not present in the model that is being augmented, the processor creates it.
2. If the info block does not contain an `augmentationLog` quality, the processor performs the following steps:
 1. If the `originalSdfModel` quality is not present in the info block, the processor adds it with a URI that can be used to access the SDF model that is currently being augmented as its value.
 2. The processor creates the `augmentationLog` quality with an array containing URIs that can be used to access the current mapping file as its sole item.
3. Otherwise, if `augmentationLog` does not contain an array, stop and throw an error.
4. Otherwise, the processor adds a URI that can be used to access the current mapping file to the array of the `augmentationLog` quality.

```
{
  "info": {
    "title": "Augmented SDF model with augmentation log.",
    "augmentationLog": [
      "https://example.org/sdf-mapping-file-1",
      "https://example.org/sdf-mapping-file-2"
    ],
    "originalSdfModel": "https://example.org/original-sdf-model"
  }
}
```

Figure 8: An augmented SDF model with an augmentation log and information regarding the original SDF model.

5. IANA Considerations

5.1. Media Type

IANA is requested to add the following Media-Type to the "Media Types" registry.

Name	Template	Reference
sdf-mapping+json	application/sdf-mapping+json	RFC XXXX, Section 5.1

Table 1: A media type for SDF mapping files

```
// RFC Editor: please replace RFC XXXX with this RFC number and
// remove this note.

Type name:  application
Subtype name:  sdf-mapping+json
Required parameters:  none
Optional parameters:  none
Encoding considerations:  binary (JSON is UTF-8-encoded text)
Security considerations:  Section 6 of RFC XXXX
Interoperability considerations:  none
Published specification:  Section 5.1 of RFC XXXX
Applications that use this media type:  Tools for data and
    interaction modeling that describes Things, i.e., physical objects
    that are available for interaction over a network
Fragment identifier considerations:  A JSON Pointer fragment
    identifier may be used, as defined in Section 6 of [RFC6901].
Person & email address to contact for further information:  ASDF WG
    mailing list (asdf@ietf.org), or IETF Applications and Real-Time
    Area (art@ietf.org)
Intended usage:  COMMON
Restrictions on usage:  none
Author/Change controller:  IETF
Provisional registration:  no
```

5.2. Registries

(TBD: After any future additions, check if we need any.)

6. Security Considerations

Some wider issues are discussed in [RFC8576].

(Specifics: TBD.)

7. References

7.1. Normative References

[BCP14] Best Current Practice 14,
<<https://www.rfc-editor.org/info/bcp14>>.
At the time of writing, this BCP comprises the following:

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>>.

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>>.

[I-D.ietf-asdf-sdf]
Koster, M., Bormann, C., and A. Kernén, "Semantic Definition Format (SDF) for Data and Interactions of Things", Work in Progress, Internet-Draft, draft-ietf-asdf-sdf-23, 17 March 2025, <<https://datatracker.ietf.org/doc/html/draft-ietf-asdf-sdf-23>>.

[RFC6901] Bryan, P., Ed., Zyp, K., and M. Nottingham, Ed., "JavaScript Object Notation (JSON) Pointer", RFC 6901, DOI 10.17487/RFC6901, April 2013, <<https://www.rfc-editor.org/rfc/rfc6901>>.

[RFC7396] Hoffman, P. and J. Snell, "JSON Merge Patch", RFC 7396, DOI 10.17487/RFC7396, October 2014, <<https://www.rfc-editor.org/rfc/rfc7396>>.

[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>>.

7.2. Informative References

[RFC8576] Garcia-Morchon, O., Kumar, S., and M. Sethi, "Internet of Things (IoT) Security: State of the Art and Challenges", RFC 8576, DOI 10.17487/RFC8576, April 2019, <<https://www.rfc-editor.org/rfc/rfc8576>>.

[W3C.wot-thing-description11] "Web of Things (WoT) Thing Description 1.1", W3C REC wot-thing-description11, W3C wot-thing-description11, <<https://www.w3.org/TR/wot-thing-description11/>>.

List of Figures

Figure 1: A simple example of an SDF mapping file
Figure 2: Input: WoT Thing Model
Figure 3: Output 1: SDF Model
Figure 4: Output 2: SDF Mapping File
Figure 5: Output 3: SDF Mapping File for Protocol Bindings
Figure 6: CDDL definition of SDF mapping file
Figure 7: An SDF model that has been augmented with WoT-specific vocabulary.

List of Tables

Table 1: A media type for SDF mapping files

Acknowledgements

This draft is based on discussions in the Thing-to-Thing Research Group (T2TRG) and the SDF working group. Input for Section 2.1 was provided by Ari Kernen.

Authors' Addresses

Carsten Bormann (editor)
Universitt Bremen TZI
Postfach 330440
D-28359 Bremen
Germany
Phone: +49-421-218-63921
Email: cabo@tzi.org

Jan Romann
Universitt Bremen
Germany
Email: jan.romann@uni-bremen.de