Workgroup: ASDF
Internet-Draft: draft-ietf-asdf-sdf-11
Published: 28 February 2022
Intended Status: Standards Track
Expires: 1 September 2022
Authors: M. Koster, Ed. C. Bormann, Ed.
PassiveLogic Universität Bremen TZI

Semantic Definition Format (SDF) for Data and Interactions of Things

Abstract

The Semantic Definition Format (SDF) is a format for domain experts to use in the creation and maintenance of data and interaction models in the Internet of Things. It was created as a common language for use in the development of the One Data Model liaison organization (OneDM) definitions. Tools convert this format to database formats and other serializations as needed.

An SDF specification describes definitions of SDF Objects and their associated interactions (Events, Actions, Properties), as well as the Data types for the information exchanged in those interactions.

A JSON format representation of SDF 1.0 was defined in version (-00) of this document; version (-05) was designated as an *implementation draft*, labeled SDF 1.1, at the IETF110 meeting of the ASDF WG (2021-03-11). The present version (-11) collects a few smaller changes as input to the 2022-02-28 ASDF WG interim. It also removes deprecated elements from SDF 1.0.

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/.

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/.

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

Status of This Memo

Copyright Notice

Table of Contents

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 in the Internet of Things. It was created as a common language for use in the development of the One Data Model liaison organization (OneDM) definitions. Tools convert this format to database formats and other serializations as needed.

An SDF specification describes definitions of SDF Objects and their associated interactions (Events, Actions, Properties), as well as the Data types for the information exchanged in those interactions.

A JSON format representation of SDF 1.0 was defined in version (-00) of this document; version (-05) was designated as an *implementation draft*, labeled SDF 1.1, at the IETF110 meeting of the ASDF WG (2021-03-11). The present version (-11) collects a few smaller changes as input to the 2022-02-28 ASDF WG interim. It also removes deprecated elements from SDF 1.0.

1.1. Terminology and Conventions

Thing: A physical item that is also made available in the Internet of Things. The term is used here for Things that are notable for their interaction with the physical world beyond interaction with humans; a temperature sensor or a light might be a Thing, but a router that employs both temperature sensors and indicator lights might exhibit less Thingness, as the effects of its functioning are mostly on the digital side.

Affordance: An element of an interface offered for interaction, defining its possible uses or making clear how it can or should be used. The term is used here for the digital interfaces of a Thing only; it might also have physical affordances such as buttons, dials, and displays.

Quality: A metadata item in a definition or declaration which says something about that definition or declaration. A quality is represented in SDF as an entry in a JSON map (object) that serves as a definition or declaration.

Entry: A key-value pair in a map. (In JSON maps, sometimes also called "member".)

Block: One or more entries in a JSON map that is part of an SDF specification; these entries together serve a specific function.

Group: An entry in the main SDF map and in certain nested definitions that has a Class Name Keyword as its key and a map of definition entries (Definition Group) as a value.

Class Name Keyword: One of sdfThing, sdfObject, sdfProperty, sdfAction, sdfEvent, or sdfData; the Classes for these type keywords are capitalized and prefixed with sdf.

Class: Abstract term for the information that is contained in groups identified by a Class Name Keyword.

Property: An affordance that can potentially be used to read, write, and/or observe state on an Object. (Note that Entries are often called properties in other environments; in this document, the term Property is specifically reserved for affordances, even

if the map key "properties" might be imported from a data definition language with the other semantics.)

Action: An affordance that can potentially be used to perform a named operation on an Object.

Event: An affordance that can potentially be used to obtain information about what happened to an Object.

Object: A grouping of Property, Action, and Event definitions; the main "atom" of reusable semantics for model construction. (Note that JSON maps are often called JSON objects due to JSON's JavaScript heritage; in this document, the term Object is specifically reserved for the above grouping, even if the type name "object" might be imported from a data definition language with the other semantics.)

Element: A part or an aspect of something abstract; used here in its usual English definition. (Occasionally, also used specifically for the elements of JSON arrays.)

Definition: An entry in a Definition Group; the entry creates a new semantic term for use in SDF models and associates it with a set of qualities.

Declaration: A reference to and a use of a definition within an enclosing definition, intended to create component instances within that enclosing definition. Every declaration can also be used as a definition for reference in a different place.

Protocol Binding: A companion document to an SDF specification that defines how to map the abstract concepts in the specification into the protocols in use in a specific ecosystem. Might supply URL components, numeric IDs, and similar details.

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

Conventions:

*The singular form is chosen as the preferred one for the keywords defined here.

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.

2. Overview

2.1. Example Definition

We start with an example for the SDF definition of a simple Object called "Switch" (Figure 1).

```
{
  "info": {
    "title": "Example file for OneDM Semantic Definition Format",
    "version": "2019-04-24",
    "copyright": "Copyright 2019 Example Corp. All rights reserved.",
    "license": "https://example.com/license"
  },
  "namespace": {
    "cap": "https://example.com/capability/cap"
  },
  "defaultNamespace": "cap",
  "sdfObject": {
    "Switch": {
      "sdfProperty": {
        "value": {
          "description": "The state of the switch; false for off and true for on.",
          "type": "boolean"
        }
      },
      "sdfAction": {
        "on": {
          "description": "Turn the switch on; equivalent to setting value to true."
        },
        "off": {
          "description": "Turn the switch off; equivalent to setting value to false."
        },
        "toggle": {
          "description": "Toggle the switch; equivalent to setting value to its complement."
        }
      }
    }
  }
}
```

Figure 1: A simple example of an SDF definition file

This is a model of a switch. The state value declared in the sdfProperty group, represented by a Boolean, will be true for "on" and will be false for "off". The actions on or off declared in the sdfAction group are redundant with setting the value and are in the example to illustrate that there are often different ways of

achieving the same effect. The action toggle will invert the value of the sdfProperty value, so that 2-way switches can be created; having such action will avoid the need for first retrieving the current value and then applying/setting the inverted value.

The sdfObject group lists the affordances of instances of this object. The sdfProperty group lists the property affordances described by the model; these represent various perspectives on the state of the object. Properties can have additional qualities to describe the state more precisely. Properties can be annotated to be read, write or read/write; how this is actually done by the underlying transfer protocols is not described in the SDF model but left to companion protocol bindings. Properties are often used with RESTful paradigms [I-D.irtf-t2trg-rest-iot], describing state. The sdfAction group is the mechanism to describe other interactions in terms of their names, input, and output data (no data are used in the example), as in a POST method in REST or in a remote procedure call. The example toggle is an Action that changes the state based on the current state of the Property named value. (The third type of affordance is Events, which are not described in this example.)

In the JSON representation, note how (with the exception of the info group) maps that have keys taken from the SDF vocabulary (info, namespace, sdfObject) alternate in nesting with maps that have keys that are freely defined by the model writer (Switch, value, on, etc.); the latter usually use the named<> production in the formal syntax of SDF (Appendix A), while the former SDF-defined vocabulary items are often, but not always, called *qualities*.

2.2. Elements of an SDF model

The SDF language uses six predefined Class Name Keywords for modeling connected Things which are illustrated in Figure 2.

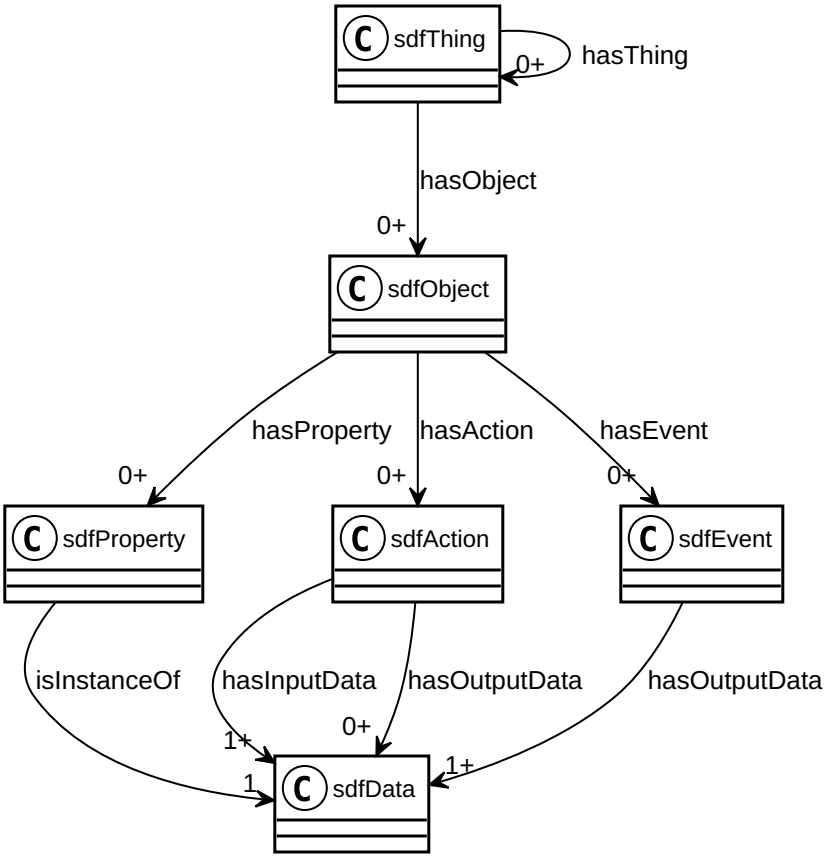

Figure 2: Main classes used in SDF models

The six main Class Name Keywords are discussed below.

2.2.1. sdfObject

Objects, the items listed in an sdfObject group, are the main "atom" of reusable semantics for model construction. It aligns in scope with common definition items from many IoT modeling systems, for example ZigBee Clusters [ZCL], OMA SpecWorks LwM2M Objects [OMA], and OCF Resource Types [OCF].

An sdfObject contains a set of sdfProperty, sdfAction, and sdfEvent definitions that describe the interaction affordances associated with some scope of functionality.

For the granularity of definition, sdfObject definitions are meant to be kept narrow enough in scope to enable broad reuse and interoperability. For example, defining a light bulb using separate sdfObject definitions for on/off control, dimming, and color control affordances will enable interoperable functionality to be configured for diverse product types. An sdfObject definition for a common on/off control may be used to control may different kinds of Things that require on/off control.

Optional qualities "minItems" and "maxItems" can be used to define sdfObjects as arrays.

2.2.2. sdfProperty

sdfProperty is used to model elements of state within sdfObject instances.

An instance of sdfProperty may be associated with some protocol affordance to enable the application to obtain the state variable and, optionally, modify the state variable. Additionally, some protocols provide for in-time reporting of state changes. (These three aspects are described by the qualities readable, writable, and observable defined for an sdfProperty.)

Definitions in sdfProperty groups include the definitions from sdfData groups, however, they actually also declare a Property with the given qualities to be potentially present in the containing Object.

For definitions in sdfProperty and sdfData, SDF provides qualities that can constrain the structure and values of data allowed in an instance of these data, as well as qualities that associate semantics to these data, for engineering units and unit scaling information.

For the data definition within sdfProperty or sdfData, SDF borrows some vocabulary proposed for the drafts 4 and 7 of the json-schema.org "JSON Schema" format (collectively called JSO here), enhanced by qualities that are specific to SDF. Details about the former are in Appendix C. For the current version of SDF, data are constrained to be of simple types (number, string, Boolean), JSON maps composed of named data ("objects"), and arrays of these types. Syntax extension points are provided that can be used to provide richer types in future versions of this specification (possibly more of which can be borrowed from json-schema.org).

Note that sdfProperty definitions (and sdfData definitions in general) are not intended to constrain the formats of data used for communication over network interfaces. Where needed, data definitions for payloads of protocol messages are expected to be part of the protocol binding.

2.2.3. sdfAction

The sdfAction group contains declarations of Actions, model affordances that, when triggered, have more effect than just reading, updating, or observing Thing state, often resulting in some outward physical effect (which, itself, cannot be modeled in SDF).

From a programmer's perspective, they might be considered to be roughly analogous to method calls.

Actions may have data parameters; these are modeled as a single item of input data and output data, each. (Where multiple parameters need to be modeled, an "object" type can be used to combine these parameters into one.) Actions may be long-running, that is to say that the effects may not take place immediately as would be expected for an update to an sdfProperty; the effects may play out over time and emit action results. Actions may also not always complete and may result in application errors, such as an item blocking the closing of an automatic door.

Actions may have (or lack) qualities of idempotency and side-effect safety.

The current version of SDF only provides data constraint modeling and semantics for the input and output data of definitions in sdfAction groups. Again, data definitions for payloads of protocol messages, and detailed protocol settings for invoking the action, are expected to be part of the protocol binding.

2.2.4. sdfEvent

The sdfEvent group contains declarations of Events, which can model affordances that inform about "happenings" associated with an instance of an Object; these may result in a signal being stored or emitted as a result.

Note that there is a trivial overlap with sdfProperty state changes, which may also be defined as events but are not generally required to be defined as such. However, Events may exhibit certain ordering, consistency, and reliability requirements that are expected to be supported in various implementations of sdfEvent that do distinguish sdfEvent from sdfProperty. For instance, while a state change may simply be superseded by another state change, some events are "precious" and need to be preserved even if further events follow.

The current version of SDF only provides data constraint modeling and semantics for the output data of Event affordances. Again, data definitions for payloads of protocol messages, and detailed protocol settings for invoking the action, are expected to be part of the protocol binding.

2.2.5. sdfData

Definitions in sdfData groups are provided separately from those in sdfProperty groups to enable common modeling patterns, data constraints, and semantic anchor concepts to be factored out for

data items that make up sdfProperty items and serve as input and output data for sdfAction and sdfEvent items.

It is a common use case for such a data definition to be shared between an sdfProperty item and input or output parameters of an sdfAction or output data provided by an sdfEvent. sdfData definitions also enable factoring out extended application data types such as mode and machine state enumerations to be reused across multiple definitions that have similar basic characteristics and requirements.

2.2.6. sdfThing

Back at the top level, the sdfThing groups enables definition of models for complex devices that will use one or more sdfObject definitions.

A definition in an sdfThing group can refine the metadata of the definitions it is composed from: other definitions in sdfThing groups definitions in sdfObject groups.

3. SDF structure

SDF definitions are contained in SDF files. One or more SDF files can work together to provide the definitions and declarations that are the payload of the SDF format.

A SDF definition file contains a single JSON map (JSON object). This object has three blocks: the information block, the namespaces block, and the definitions block.

3.1. Information block

The information block contains generic meta data for the file itself and all included definitions. To enable tool integration, the information block is optional in the grammar of SDF; most processes for working with SDF files will have policies that only SDF models with an info block can be processed. It is therefore **RECOMMENDED** that SDF validator tools emit a warning when no information block is found.

The keyword (map key) that defines an information block is "info". Its value is a JSON map in turn, with a set of entries that represent qualities that apply to the included definition.

Qualities of the information block are shown in Table 1.

Quality	Type	Required	Description
title	string	no	

Quality	Type	Required	Description
			A short summary to be displayed in search results, etc.
version	string	no	The incremental version of the definition, format TBD
copyright	string	no	Link to text or embedded text containing a copyright notice
license	string	no	Link to text or embedded text containing license terms

Table 1: Qualities of the Information Block

While the format of the version string is marked as TBD, it is intended to be lexicographically increasing over the life of a model: a newer model always has a version string that string-compares higher than all previous versions. This is easily achieved by following the convention to start the version with an [RFC3339] date-time or, if new versions are generated less frequently than once a day, just the full-date (i.e., YYYY-MM-DD); in many cases, that will be all that is needed (see Figure 1 for an example).

The license string is preferably either a URI that points to a web page with an unambiguous definition of the license, or an [SPDX] license identifier. (For models to be handled by the One Data Model liaison group, this will typically be "BSD-3-Clause".)

3.2. Namespaces block

The namespaces block contains the namespace map and the defaultNamespace setting.

The namespace map is a map from short names for URIs to the namespace URIs themselves.

The defaultNamespace setting selects one of the entries in the namespace map by giving its short name. The associated URI (value of this entry) becomes the default namespace for the SDF definition file.

Quality	Type	Required	Description
namespace	map	no	Defines short names mapped to namespace URIs, to be used as identifier prefixes
defaultNamespace	string	no	Identifies one of the prefixes in the namespace map to be used as a default in resolving identifiers

Table 2: Namespaces Block

The following example declares a set of namespaces and defines cap as the default namespace. By convention, the values in the namespace map contain full URIs without a fragment identifier, and the fragment identifier is then added, if needed, where the namespace entry is used.

```
"namespace": {
  "cap": "https://example.com/capability/cap",
  "zcl": "https://zcl.example.com/sdf"
},
"defaultNamespace": "cap"
```

If no defaultNamespace setting is given, the SDF definition file does not contribute to a global namespace. As the defaultNamespace is set by giving a namespace short name, its presence requires a namespace map that contains a mapping for that namespace short name.

If no namespace map is given, no short names for namespace URIs are set up, and no defaultNamespace can be given.

3.3. Definitions block

The Definitions block contains one or more groups, each identified by a Class Name Keyword (there can only be one group per keyword; the actual grouping is just a shortcut and does not carry any specific semantics). The value of each group is a JSON map (object), the keys of which serve for naming the individual definitions in this group, and the corresponding values provide a set of qualities (name-value pairs) for the individual definition. (In short, we speak of the map entries as "named sets of qualities".)

Each group may contain zero or more definitions. Each identifier defined creates a new type and term in the target namespace. Declarations have a scope of the current definition block.

A definition may in turn contain other definitions. Each definition is a named set of qualities, i.e., it consists of the newly defined identifier and a set of key-value pairs that represent the defined qualities and contained definitions.

An example for an Object definition is given in Figure 3:

```
"sdfObject": {
  "foo": {
    "sdfProperty": {
      "bar": {
        "type": "boolean"
      }
    }
  }
}
```

Figure 3: Example Object definition

This example defines an Object "foo" that is defined in the default namespace (full address: #/sdfObject/foo), containing a property that can be addressed as #/sdfObject/foo/sdfProperty/bar, with data of type boolean.

Some of the definitions are also declarations: the definition of the entry "bar" in the property "foo" means that each instance of a "foo" can have zero or one instance of a "bar". Entries within sdfProperty, sdfAction, and sdfEvent, within sdfObject entries, are declarations. Similarly, entries within an sdfThing describe instances of sdfObject (or nested sdfThing) that form part of instances of the Thing.

4. Names and namespaces

SDF definition files may contribute to a global namespace, and may reference elements from that global namespace. (An SDF definition file that does not set a defaultNamespace does not contribute to a global namespace.)

4.1. Structure

Global names look exactly like https:// URIs with attached fragment identifiers.

There is no intention to require that these URIs can be dereferenced. (However, as future versions of SDF might find a use for dereferencing global names, the URI should be chosen in such a way that this may become possible in the future.)

The absolute URI of a global name should be a URI as per Section 3 of [RFC3986], with a scheme of "https" and a path (hier-part in [RFC3986]). For the present version of this specification, the query part should not be used (it might be used in later versions).

The fragment identifier is constructed as per Section 6 of [RFC6901].

4.2. Contributing global names

The fragment identifier part of a global name defined in an SDF definition file is constructed from a JSON pointer that selects the element defined for this name in the SDF definition file.

The absolute URI part is a copy of the default namespace, i.e., the default namespace is always the target namespace for a name for which a definition is contributed. When emphasizing that name definitions are contributed to the default namespace, we therefore also call it the "target namespace" of the SDF definition file.

E.g., in Figure 1, definitions for the following global names are contributed:

*https://example.com/capability/cap#/sdfObject/Switch

*https://example.com/capability/cap#/sdfObject/Switch/sdfProperty/value

*https://example.com/capability/cap#/sdfObject/Switch/sdfAction/on

*https://example.com/capability/cap#/sdfObject/Switch/sdfAction/off

Note the #, which separates the absolute-URI part (Section 4.3 of [RFC3986]) from the fragment identifier part.

4.3. Referencing global names

A name reference takes the form of the production curie in [W3C.NOTE-curie-20101216] (note that this excludes the production safe-curie), but also limiting the IRIs involved in that production to URIs as per [RFC3986] and the prefixes to ASCII characters [RFC0020].

A name that is contributed by the current SDF definition file can be referenced by a Same-Document Reference as per Section 4.4 of [RFC3986]. As there is little point in referencing the entire SDF definition file, this will be a # followed by a JSON pointer. This is the only kind of name reference to itself that is possible in an SDF definition file that does not set a default namespace.

Name references that point outside the current SDF definition file need to contain curie prefixes. These then reference namespace declarations in the namespaces block.

For example, if a namespace prefix is defined:

```
"namespace": {
  "foo": "https://example.com/"
}
```

Then this reference to that namespace:

```
{ "sdfRef": "foo:#/sdfData/temperatureData" }
```

references the global name:

```
"https://example.com/#/sdfData/temperatureData"
```

Note that there is no way to provide a URI scheme name in a curie, so all references outside of the document need to go through the namespace map.

Name references occur only in specific elements of the syntax of SDF:

*copying elements via sdfRef values

*pointing to elements via sdfRequired value elements

4.4. sdfRef

In a JSON map establishing a definition, the keyword "sdfRef" is used to copy all of the qualities of the referenced definition, indicated by the included name reference, into the newly formed definition. (This can be compared to the processing of the "$ref" keyword in [I-D.handrews-json-schema-validation].)

For example, this reference:

```
"temperatureProperty": {
  "sdfRef": "#/sdfData/temperatureData"
}
```

creates a new definition "temperatureProperty" that contains all of the qualities defined in the definition at /sdfData/temperatureData.

The sdfRef member need not be the only member of a map. Additional members may be present with the intention to override parts of the

referenced map. More formally, for a JSON map that contains an sdfRef member, the semantics is defined to be as if the following steps were performed:

1. The JSON map that contains the sdfRef member is copied into a variable named "patch".

2. The sdfRef member of the copy in "patch" is removed.

3. the JSON pointer that is the value of the sdfRef member is dereferenced and the result is copied into a variable named "original".

4. The JSON Merge Patch algorithm [RFC7396] is applied to patch the contents of "original" with the contents of "patch".

5. The result of the Merge Patch is used in place of the value of the original JSON map.

TODO: Make sure that the grammar in Appendix A allows specifying the null values that are necessary to remove members in a merge-patch.

4.5. sdfRequired

The keyword "sdfRequired" is provided to apply a constraint that defines for which declarations corresponding data are mandatory in an instance conforming the current definition.

The value of "sdfRequired" is an array of name references (JSON pointers), each indicating one declaration that is mandatory to be represented.

The example in Figure 4 shows two required elements in the sdfObject definition for "temperatureWithAlarm", the sdfProperty "currentTemperature", and the sdfEvent "overTemperatureEvent". The example also shows the use of JSON pointer with "sdfRef" to use a pre-existing definition in this definition, for the "alarmType" data (sdfOutputData) produced by the sdfEvent "overTemperatureEvent".

```
{
  "sdfObject": {
    "temperatureWithAlarm": {
      "sdfRequired": [
        "#/sdfObject/temperatureWithAlarm/sdfProperty/currentTemperature",
        "#/sdfObject/temperatureWithAlarm/sdfEvent/overTemperatureEvent"
      ],
      "sdfData":{
        "temperatureData": {
          "type": "number"
        }
      },
      "sdfProperty": {
        "currentTemperature": {
          "sdfRef": "#/sdfObject/temperatureWithAlarm/sdfData/temperatureData"
        }
      },
      "sdfEvent": {
        "overTemperatureEvent": {
          "sdfOutputData": {
            "type": "object",
            "properties": {
              "alarmType": {
                "sdfRef": "cap:#/sdfData/alarmTypes/quantityAlarms",
                "const": "OverTemperatureAlarm"
              },
              "temperature": {
                "sdfRef": "#/sdfObject/temperatureWithAlarm/sdfData/temperatureData"
              }
            }
          }
        }
      }
    }
  }
}
```

Figure 4: Using sdfRequired

4.6. Common Qualities

Definitions in SDF share a number of qualities that provide metadata for them. These are listed in Table 3. None of these qualities are required or have default values that are assumed if the quality is absent. If a label is required for an application and no label is given in the SDF model, the last part (reference-token, Section 3 of [RFC6901]) of the JSON pointer to the definition can be used.

Quality	Type	Description
description	text	long text (no constraints)
label	text	short text (no constraints)
$comment	text	source code comments only, no semantics
sdfRef	sdf-pointer	(see Section 4.4)
sdfRequired	pointer-list	(see Section 4.5, applies to qualities of properties, of data)

Table 3: Common Qualities

4.7. Data Qualities

Data qualities are used in sdfData and sdfProperty definitions, which are named sets of data qualities (abbreviated as named-sdq).

Appendix C lists data qualities inspired by the various proposals at json-schema.org; the intention is that these (information model level) qualities are compatible with the (data model) semantics from the versions of the json-schema.org proposal they were imported from.

Table 4 lists data qualities defined specifically for the present specification.

Quality	Type	Description	Default
(common)		Section 4.6	
unit	string	unit name (note 1)	N/A
scaleMinimum	number	lower limit of value in units given by unit (note 2)	N/A
scaleMaximum	number	upper limit of value in units given by unit (note 2)	N/A
nullable	boolean	indicates a null value is available for this type	true
contentFormat	string	content type (IANA media type string plus parameters), encoding	N/A
sdfType	string (Section 4.7.1)	sdfType enumeration (extensible)	N/A
sdfChoice	named set of data qualities (Section 4.7.2)	named alternatives	N/A
enum	array of strings	abbreviation for string-valued named alternatives	N/A

Table 4: SDF-defined Qualities of sdfData

1. Note that the quality unit was called units in SDF 1.0. The unit name **SHOULD** be as per the SenML Units Registry or the Secondary Units Registry in [IANA.senml] as specified by Sections 4.5.1 and 12.1 of [RFC8428] and Section 3 of [RFC8798], respectively.

   Exceptionally, if a registration in these registries cannot be obtained or would be inappropriate, the unit name can also be a URI that is pointing to a definition of the unit. Note that SDF processors are not expected to (and normally **SHOULD NOT**) dereference these URIs; they may be useful to humans, though. A URI unit name is distinguished from a registered unit name by the presence of a colon; registered unit names that contain a colon (at the time of writing, none) can therefore not be used in SDF.

   For use by translators into ecosystems that require URIs for unit names, the URN sub-namespace "urn:ietf:params:unit" is provided (Section 7.2); URNs from this sub-namespace **MUST NOT** be used in a unit quality, in favor of simply notating the unit name (e.g., kg instead of urn:ietf:params:unit:kg).

2. these qualities were included in SDF 1.0, but were not fully defined; they are not included in SDF 1.1. In 1.next, they will be replaced by qualities to express scaling that are more aligned with the processes that combine ecosystem and instance specific information with an SDF model.

4.7.1. sdfType

SDF defines a number of basic types beyond those provided by JSON or JSO. These types are identified by the sdfType quality, which is a text string from a set of type names defined by SDF.

To aid interworking with [I-D.handrews-json-schema-validation] implementations, it is **RECOMMENDED** that sdfType is always used in conjunction with the type quality inherited from [I-D.handrews-json-schema-validation], in such a way as to yield a common representation of the type's values in JSON.

Values for sdfType that are defined in SDF 1.1 are shown in Table 5. This table also gives a description of the semantics of the sdfType, the conventional value for type to be used with the sdfType value, and a conventional JSON representation for values of the type.

sdfType	Description	type	JSON Representation
		string	

sdfType	Description	type	JSON Representation
byte-string	A sequence of zero or more bytes		base64url without padding (Section 3.4.5.2 of [RFC8949])
unix-time	A point in civil time (note 1)	number	POSIX time (Section 3.4.2 of [RFC8949])

Table 5: Values defined in SDF 1.1 for sdfType quality

(1) Note that the definition of unix-time does not imply the capability to represent points in time that fall on leap seconds. More date/time-related sdfTypes are likely to be added in future versions of this specification.

In SDF 1.0, a similar concept was called subtype.

4.7.2. sdfChoice

Data can be a choice of named alternatives, called sdfChoice. Each alternative is identified by a name (string, key in the JSON object used to represent the choice) and a set of dataqualities (object, the value in the JSON object used to represent the choice).

sdfChoice merges the functions of two constructs found in [I-D.handrews-json-schema-validation]:

*enum

What would have been

```
"enum": ["foo", "bar", "baz"]
```

in SDF 1.0, is often best represented as:

```
"sdfChoice": {
  "foo": { "description": "This is a foonly"},
  "bar": { "description": "As defined in the second world congress"},
  "baz": { "description": "From zigbee foobaz"}
}
```

This allows the placement of other dataqualities such as description in the example.

If an enum needs to use a data type different from text string, e.g. what would have been

```
"type": "number",
"enum": [1, 2, 3]
```

in SDF 1.0, is represented as:

```
"type": "number",
"sdfChoice": {
  "a-better-name-for-alternative-1": { "const": 1 },
  "alternative-2": { "const": 2 },
  "the-third-alternative": { "const": 3 }
}
```

where the string names obviously would be chosen in a way that is descriptive for what these numbers actually stand for; sdfChoice also makes it easy to add number ranges into the mix.

(Note that const can also be used for strings as in the previous example, e.g., if the actual string value is indeed a crucial element for the data model.)

*anyOf

[I-D.handrews-json-schema-validation] provides a type union called anyOf, which provides a choice between anonymous alternatives.

What could have been

```
"anyOf": [
  {"type": "array", "minItems": 3, "maxItems": "3", "items": {
     "$ref": "#/sdfData/rgbVal"}},
  {"type": "array", "minItems": 4, "maxItems": "4", "items": {
     "$ref": "#/sdfData/cmykVal"}}
]
```

in [I-D.handrews-json-schema-validation] can be more descriptively notated in SDF as:

```
"sdfChoice": {
  "rgb": {"type": "array", "minItems": 3, "maxItems": "3", "items": {
            "sdfRef": "#/sdfData/rgbVal"}},
  "cmyk": {"type": "array", "minItems": 4, "maxItems": "4", "items": {
            "sdfRef": "#/sdfData/cmykVal"}}
}
```

Note that there is no need in SDF for the type intersection construct allOf or the peculiar type-xor construct oneOf found in [I-D.handrews-json-schema-validation].

As a simplification for readers of SDF specifications accustomed to the [I-D.handrews-json-schema-validation] enum keyword, this is retained, but limited to a choice of text string values, such that

```
"enum": ["foo", "bar", "baz"]
```

is syntactic sugar for

```
"sdfChoice": {
  "foo": { "const": "foo"},
  "bar": { "const": "bar"},
  "baz": { "const": "baz"}
}
```

5. Keywords for definition groups

The following SDF keywords are used to create definition groups in the target namespace. All these definitions share some common qualities as discussed in Section 4.6.

5.1. sdfObject

The sdfObject keyword denotes a group of zero or more Object definitions. Object definitions may contain or include definitions of Properties, Actions, Events declared for the object, as well as data types (sdfData group) to be used in this or other Objects.

The qualities of an sdfObject include the common qualities, additional qualities are shown in Table 6. None of these qualities are required or have default values that are assumed if the quality is absent.

Quality	Type	Description
(common)		Section 4.6
sdfProperty	property	zero or more named property definitions for this object
sdfAction	action	zero or more named action definitions for this object
sdfEvent	event	zero or more named event definitions for this object
sdfData		

Quality	Type	Description
	named-sdq	zero or more named data type definitions that might be used in the above
minItems	number	(array) Minimum number of sdfObject instances in array
maxItems	number	(array) Maximum number of sdfObject instances in array

Table 6: Qualities of sdfObject

5.2. sdfProperty

The sdfProperty keyword denotes a group of zero or more Property definitions.

Properties are used to model elements of state.

The qualities of a Property definition include the data qualities (and thus the common qualities), see Section 4.7, additional qualities are shown in Table 7.

Quality	Type	Description	Default
(data)		Section 4.7	
readable	boolean	Reads are allowed	true
writable	boolean	Writes are allowed	true
observable	boolean	flag to indicate asynchronous notification is available	true

Table 7: Qualities of sdfProperty

5.3. sdfAction

The sdfAction keyword denotes a group of zero or more Action definitions.

Actions are used to model commands and methods which are invoked. Actions have parameter data that are supplied upon invocation.

The qualities of an Action definition include the common qualities, additional qualities are shown in Table 8.

Quality	Type	Description
(common)		Section 4.6
sdfInputData	map	data qualities of the input data for an Action
sdfOutputData	map	data qualities of the output data for an Action
sdfData	named-sdq	zero or more named data type definitions that might be used in the above

Table 8: Qualities of sdfAction

sdfInputData defines the input data of the action. sdfOutputData defines the output data of the action. As discussed in Section 2.2.3, a set of data qualities with type "object" can be used to substructure either data item, with optionality indicated by the data quality required.

5.4. sdfEvent

The sdfEvent keyword denotes zero or more Event definitions.

Events are used to model asynchronous occurrences that may be communicated proactively. Events have data elements which are communicated upon the occurrence of the event.

The qualities of sdfEvent include the common qualities, additional qualities are shown in Table 9.

Quality	Type	Description
(common)		Section 4.6
sdfOutputData	map	data qualities of the output data for an Event
sdfData	named-sdq	zero or more named data type definitions that might be used in the above

Table 9: Qualities of sdfEvent

sdfOutputData defines the output data of the action. As discussed in Section 2.2.4, a set of data qualities with type "object" can be used to substructure the output data item, with optionality indicated by the data quality required.

5.5. sdfData

The sdfData keyword denotes a group of zero or more named data type definitions (named-sdq).

An sdfData definition provides a reusable semantic identifier for a type of data item and describes the constraints on the defined type. It is not itself a declaration, i.e., it does not cause any of these data items to be included in an affordance definition.

The qualities of sdfData include the data qualities (and thus the common qualities), see Section 4.7.

6. High Level Composition

The requirements for high level composition include the following:

*The ability to represent products, standardized product types, and modular products while maintaining the atomicity of Objects.

*The ability to compose a reusable definition block from Objects, for example a single plug unit of an outlet strip with on/off control, energy monitor, and optional dimmer objects, while retaining the atomicity of the individual objects.

*The ability to compose Objects and other definition blocks into a higher level thing that represents a product, while retaining the atomicity of objects.

*The ability to enrich and refine a base definition to have product-specific qualities and quality values, e.g. unit, range, and scale settings.

*The ability to reference items in one part of a complex definition from another part of the same definition, for example to summarize the energy readings from all plugs in an outlet strip.

6.1. Paths in the model namespaces

The model namespace is organized according to terms that are defined in the definition files that are present in the namespace. For example, definitions that originate from an organization or vendor are expected to be in a namespace that is specific to that organization or vendor. There is expected to be an SDF namespace for common SDF definitions used in OneDM.

The structure of a path in a namespace is defined by the JSON Pointers to the definitions in the files in that namespace. For example, if there is a file defining an object "Switch" with an action "on", then the reference to the action would be "ns:/sdfObject/Switch/sdfAction/on" where ns is the namespace prefix (short name for the namespace).

6.2. Modular Composition

Modular composition of definitions enables an existing definition (could be in the same file or another file) to become part of a new definition by including a reference to the existing definition within the model namespace.

6.2.1. Use of the "sdfRef" keyword to re-use a definition

An existing definition may be used as a template for a new definition, that is, a new definition is created in the target namespace which uses the defined qualities of some existing definition. This pattern will use the keyword "sdfRef" as a quality of a new definition with a value consisting of a reference to the existing definition that is to be used as a template.

In the definition that uses "sdfRef", new qualities may be added and existing qualities from the referenced definition may be overridden. (Note that JSON maps (objects) do not have a defined order, so the SDF processor may see these overrides before seeing the sdfRef.)

As a convention, overrides are intended to be used only for further restricting the set of data values, as shown in Figure 5: any value for a cable-length also is a valid value for a length, with the additional restriction that the length cannot be smaller than 5 cm. (This is labeled as a convention as it cannot be checked in the general case; a quality of implementation consideration for a tool might be to provide at least some form of checking.) Note that a description is provided that overrides the description of the referenced definition; as this quality is intended for human consumption there is no conflict with the intended goal.

```
"sdfData":
  "length" : {
    "type": "number",
    "minimum": 0,
    "unit": "m"
    "description": "There can be no negative lengths."
  }
...
  "cable-length" : {
    "sdfRef": "#/sdfData/length"
    "minimum": 5e-2,
    "description": "Cables must be at least 5 cm."
  }
```

Figure 5

6.3. sdfThing

An sdfThing is a set of declarations and qualities that may be part of a more complex model. For example, the object declarations that make up the definition of a single socket of an outlet strip could be encapsulated in an sdfThing, and the socket-thing itself could be used in a declaration in the sdfThing definition for the outlet strip.

sdfThing definitions carry semantic meaning, such as a defined refrigerator compartment and a defined freezer compartment, making up a combination refrigerator-freezer product.

An sdfThing may be composed of sdfObjects and other sdfThings.

The qualities of sdfThing are shown in Table 10.

Quality	Type	Description
(common)		Section 4.6
sdfThing	thing	
sdfObject	object	

Table 10: Qualities of sdfThing

7. IANA Considerations

7.1. Media Type

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

Name	Template	Reference
sdf+json	application/sdf+json	RFC XXXX, Section 7.1

Table 11

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

Type name: application
Subtype name: sdf+json
Required parameters: none
Optional parameters: none
Encoding considerations: binary (JSON is UTF-8-encoded text)
Security considerations: Section 8 of RFC XXXX
Interoperability considerations: none
Published specification: Section 7.1 of RFC XXXX
Applications that use this media type: Tools for data and interaction modeling in the Internet of Things
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

7.2. IETF URN Sub-namespace for Unit Names (urn:ietf:params:unit)

IANA is requested to register the following value in the "IETF URN Sub-namespace for Registered Protocol Parameter Identifiers" registry, following the template in [RFC3553]:

Registry name: unit

Specification: RFC XXXX

Repository: combining the symbol values from the SenML Units Registry and the Secondary Units Registry in [IANA.senml] as specified by Sections 4.5.1 and 12.1 of [RFC8428] and Section 3 of [RFC8798], respectively (which by the registration policy are guaranteed to be non-overlapping).

Index value: Percent-encoding (Section 2.1 of [RFC3986]) is required of any characters in unit names as required by ABNF rule "pchar" in Section 3.3 of [RFC3986], specifically at the time of writing for the unit names "%" (deprecated in favor of "/"), "%RH", "%EL".

7.3. Registries

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

8. Security Considerations

Some wider issues are discussed in [RFC8576].

(Specifics: TBD.)

9. References

9.1. Normative References

[IANA.params] IANA, "Uniform Resource Name (URN) Namespace for IETF Use", <https://www.iana.org/assignments/params>.

[IANA.senml] IANA, "Sensor Measurement Lists (SenML)", <https://www.iana.org/assignments/senml>.

[RFC0020] Cerf, V., "ASCII format for network interchange", STD 80, RFC 20, DOI 10.17487/RFC0020, October 1969, <https://www.rfc-editor.org/info/rfc20>.

[RFC2119] Bradner, S., "Key words for use in RFCs to Indicate Requirement Levels", BCP 14, RFC 2119, DOI 10.17487/RFC2119, March 1997, <https://www.rfc-editor.org/info/rfc2119>.

[RFC3339] Klyne, G. and C. Newman, "Date and Time on the Internet: Timestamps", RFC 3339, DOI 10.17487/RFC3339, July 2002, <https://www.rfc-editor.org/info/rfc3339>.

[RFC3553] Mealling, M., Masinter, L., Hardie, T., and G. Klyne, "An IETF URN Sub-namespace for Registered Protocol Parameters", BCP 73, RFC 3553, DOI 10.17487/RFC3553, June 2003, <https://www.rfc-editor.org/info/rfc3553>.

[RFC3986] Berners-Lee, T., Fielding, R., and L. Masinter, "Uniform Resource Identifier (URI): Generic Syntax", STD 66, RFC 3986, DOI 10.17487/RFC3986, January 2005, <https://www.rfc-editor.org/info/rfc3986>.

[RFC4122] Leach, P., Mealling, M., and R. Salz, "A Universally Unique IDentifier (UUID) URN Namespace", RFC 4122, DOI 10.17487/RFC4122, July 2005, <https://www.rfc-editor.org/info/rfc4122>.

[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/info/rfc6901>.

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

[RFC8174] Leiba, B., "Ambiguity of Uppercase vs Lowercase in RFC 2119 Key Words", BCP 14, RFC 8174, DOI 10.17487/RFC8174, May 2017, <https://www.rfc-editor.org/info/rfc8174>.

[RFC8428] Jennings, C., Shelby, Z., Arkko, J., Keranen, A., and C. Bormann, "Sensor Measurement Lists (SenML)", RFC 8428, DOI 10.17487/RFC8428, August 2018, <https://www.rfc-editor.org/info/rfc8428>.

[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/info/rfc8610>.

[RFC8798] Bormann, C., "Additional Units for Sensor Measurement Lists (SenML)", RFC 8798, DOI 10.17487/RFC8798, June 2020, <https://www.rfc-editor.org/info/rfc8798>.

[RFC8949] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", STD 94, RFC 8949, DOI 10.17487/

RFC8949, December 2020, <https://www.rfc-editor.org/info/rfc8949>.

[RFC9165] Bormann, C., "Additional Control Operators for the Concise Data Definition Language (CDDL)", RFC 9165, DOI 10.17487/RFC9165, December 2021, <https://www.rfc-editor.org/info/rfc9165>.

[SPDX] "SPDX License List", <https://spdx.org/licenses/>.

[W3C.NOTE-curie-20101216] Birbeck, M. and S. McCarron, "CURIE Syntax 1.0", World Wide Web Consortium NOTE NOTE-curie-20101216, 16 December 2010, <https://www.w3.org/TR/2010/NOTE-curie-20101216>.

9.2. Informative References

[ECMA-262] Ecma International, "ECMAScript 2020 Language Specification", ECMA Standard ECMA-262, 11th Edition, June 2020, <https://www.ecma-international.org/wp-content/uploads/ECMA-262.pdf>.

[I-D.bormann-jsonpath-iregexp] Bormann, C., "I-Regexp: An Interoperable Regexp Format", Work in Progress, Internet-Draft, draft-bormann-jsonpath-iregexp-02, 17 January 2022, <https://www.ietf.org/archive/id/draft-bormann-jsonpath-iregexp-02.txt>.

[I-D.handrews-json-schema-validation] Wright, A., Andrews, H., and B. Hutton, "JSON Schema Validation: A Vocabulary for Structural Validation of JSON", Work in Progress, Internet-Draft, draft-handrews-json-schema-validation-02, 17 September 2019, <https://www.ietf.org/archive/id/draft-handrews-json-schema-validation-02.txt>.

[I-D.irtf-t2trg-rest-iot] Keranen, A., Kovatsch, M., and K. Hartke, "Guidance on RESTful Design for Internet of Things Systems", Work in Progress, Internet-Draft, draft-irtf-t2trg-rest-iot-09, 26 February 2022, <https://www.ietf.org/archive/id/draft-irtf-t2trg-rest-iot-09.txt>.

[I-D.wright-json-schema] Wright, A. and H. Andrews, "JSON Schema: A Media Type for Describing JSON Documents", Work in Progress, Internet-Draft, draft-wright-json-schema-01, 16

April 2017, <https://www.ietf.org/archive/id/draft-wright-json-schema-01.txt>.

[OCF] "OCF Resource Type Specification", <https://openconnectivity.org/specs/OCF_Resource_Type_Specification.pdf>.

[OMA] "OMA LightweightM2M (LwM2M) Object and Resource Registry", <http://www.openmobilealliance.org/wp/omna/lwm2m/lwm2mregistry.html>.

[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/info/rfc8576>.

[ZCL] "The ZigBee Cluster Library", Zigbee Wireless Networking pp. 239-271, DOI 10.1016/b978-0-7506-8597-9.00006-9, 2008, <https://doi.org/10.1016/b978-0-7506-8597-9.00006-9>.

Appendix A. Formal Syntax of SDF

This appendix describes the syntax of SDF using CDDL [RFC8610]. Note that this appendix was derived from Ari Keranen's "alt-schema" and Michael Koster's "schema", with a view of covering the syntax that is currently in use at the One Data Model playground repository.

This appendix shows the framework syntax only, i.e., a syntax with liberal extension points. Since this syntax is nearly useless in finding typos in an SDF specification, a second syntax, the validation syntax, is defined that does not include the extension points. The validation syntax can be generated from the framework syntax by leaving out all lines containing the string EXTENSION-POINT; as this is trivial, the result is not shown here.

This appendix makes use of CDDL "features" as defined in Section 4 of [RFC9165]. A feature named "1.0" is used to indicate parts of the syntax being deprecated towards SDF 1.1, and a feature named "1.1" is used to indicate new syntax intended for SDF 1.1. Features whose names end in "-ext" indicate extension points for further evolution.

```
start = sdf-syntax

sdf-syntax = {
 ? info: sdfinfo                  ; This will be required in most process policies, but not a
 ? namespace: named<text>
 ? defaultNamespace: text
 ? sdfThing: named<thingqualities>       ; Thing is a composition of objects that work togeth
 ? sdfProduct: named<productqualities>   ; Product is a composition of things and objects tha
 ? sdfObject: named<objectqualities>     ; Object is a set of Properties, Actions, and Events
 ? sdfProperty: named<propertyqualities> ; Property represents the state of an instance of an
 ? sdfAction: named<actionqualities>     ; Action is a directive to invoke an application lay
 ? sdfEvent: named<eventqualities>       ; Event represents an occurrence of something associ
 ? sdfData: named<dataqualities>         ; Data represents a piece of information that can be
 EXTENSION-POINT<"top-ext">
}

sdfinfo = {
 ? title: text
 ? version: text
 ? copyright: text
 ? license: text
 EXTENSION-POINT<"info-ext">
}

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

EXTENSION-POINT<f> = ( * (text .feature f) => any ) ; only used in framework syntax

sdf-pointer = text ; .regexp curie-regexp -- TO DO!
pointer-list = [* sdf-pointer]  ; ISSUE: no point in having an empty list, no? but used for s

commonqualities = (
 ? description: text            ; long text (no constraints)
 ? label: text                  ; short text (no constraints); default to key
 ? $comment: text               ; source code comments only, no semantics
 ? sdfRef: sdf-pointer
 ? sdfRequired: pointer-list    ; applies to qualities of properties, of data
)

; for building hierarchy
thingqualities = {
 commonqualities
 ? sdfObject: named<objectqualities>
 ? sdfThing: named<thingqualities>
 EXTENSION-POINT<"thing-ext">
}

productqualities = thingqualities ; ISSUE: get rid of sdfProduct?
```

```
; for single objects, or for arrays of objects (1.2)
objectqualities = {
 commonqualities
 ? ("minItems" .feature "1.2") => number
 ? ("maxItems" .feature "1.2") => number
 ? sdfProperty: named<propertyqualities>
 ? sdfAction: named<actionqualities>
 ? sdfEvent: named<eventqualities>
 ? sdfData: named<dataqualities>
 EXTENSION-POINT<"object-ext">
}

propertyqualities = dataqualities ; the definitions in sdfData are declarations in sdfPropert

parameter-list =
  pointer-list .feature (["1.0", "pointerlist-as-parameter"]) /
  dataqualities .feature (["1.1", "dataqualities-as-parameter"])

actionqualities = {
 commonqualities
 ? sdfInputData: parameter-list   ; sdfRequiredInputData applies here (a bit redundant)
 ? ("sdfRequiredInputData" .feature "1.0") => pointer-list
 ? sdfOutputData: parameter-list  ; sdfRequired applies here
 ? sdfData: named<dataqualities>         ; zero or more named data type definitions that migh
 EXTENSION-POINT<"action-ext">
}

eventqualities = {
 commonqualities
 ? sdfOutputData: parameter-list  ; sdfRequired applies here
 ? sdfData: named<dataqualities>         ; zero or more named data type definitions that migh
 EXTENSION-POINT<"event-ext">
}

dataqualities = {               ; also propertyqualities
 commonqualities
 jsonschema
 ? ("units" .feature "1.0") => text
 ? ("unit" .feature "1.1") => text
 ? ("scaleMinimum" .feature "1.0") => number
 ? ("scaleMaximum" .feature "1.0") => number
 ? observable: bool
 ? readable: bool
 ? writable: bool
 ? nullable: bool
 ? ("subtype" .feature "1.0") => "byte-string" / "unix-time"
            / (text .feature "subtype-ext")                      ; EXTENSION-POINT
 ? ("sdfType" .feature "1.1") => "byte-string" / "unix-time"
            / (text .feature "sdftype-ext")                      ; EXTENSION-POINT
```

```
 ? contentFormat: text
 EXTENSION-POINT<"data-ext">
}

allowed-types = number / text / bool / null
              / [* number] / [* text] / [* bool]
              / {* text => any}
              / (any .feature "allowed-ext")                    ; EXTENSION-POINT

compound-type = (
  "type" => ("object" .feature "1.1")
  ? required: [+text]
  ? properties: named<dataqualities>
)

choice-type = (
  ("sdfChoice" .feature "1.1") => named<dataqualities>
)

jsonschema = (
 ? (("type" => "number" / "string" / "boolean" / "integer" / "array")
    // compound-type
    // choice-type
    // (type: text .feature "type-ext")                      ; EXTENSION-POINT
   )
 ? "enum" => [+ text]               ; limited to text strings in SDF 1.1
 ? ("enum" .feature "1.0") => [+ allowed-types]    ; should validate against type
 ? const: allowed-types                            ; should validate against type
 ? default: allowed-types                          ; should validate against type
 ; number/integer constraints
 ? minimum: number
 ? maximum: number
 ? exclusiveMinimum: bool / number      ; jso draft 4/7
 ? exclusiveMaximum: bool / number      ; jso draft 4/7
 ? multipleOf: number                   ; ISSUE: Do we need this?
 ; text string constraints
 ? minLength: number
 ? maxLength: number
 ? pattern: text                ; regexp
 ? format: "date-time" / "date" / "time"
           / "uri" / "uri-reference" / "uuid"
           / (text .feature "format-ext")                       ; EXTENSION-POINT
 ; array constraints
 ? minItems: number
 ? maxItems: number
 ? uniqueItems: bool
 ? items: { ;;; ultimately, this will be mostly recursive, but, for now
            ;;; let's find out what we actually need
     ? sdfRef: sdf-pointer          ; import limited to the subset that we allow here...
```

```
    ? description: text            ; long text (no constraints)
    ? $comment: text               ; source code comments only, no semantics
    ; commonqualities, ; -- ISSUE: should leave this out for non-complex data types, but nee
    ? ((type: "number" / "string" / "boolean" / "integer") ; no "array"
       // compound-type
       // choice-type              ; do we really need arrays of choices?
       // (type: text .feature "itemtype-ext")                      ; EXTENSION-POINT
      )
    ; jso subset
    ? minimum: number
    ? maximum: number
    ? "enum" => [+ text] ; limited to text strings in SDF 1.1
    ? ("enum" .feature "1.0") => [+ any]
    ? format: text
    ? minLength: number
    ? maxLength: number
    EXTENSION-POINT<"items-ext">
  }
)
```

Appendix B. json-schema.org Rendition of SDF Syntax

This appendix describes the syntax of SDF defined in Appendix A, but using a version of the description techniques advertised on json-schema.org [I-D.handrews-json-schema-validation].

The appendix shows both the validation and the framework syntax. Since most of the lines are the same between these two files, those lines are shown only once, with a leading space, in the form of a unified diff. Lines leading with a - are part of the validation syntax, and lines leading with a + are part of the framework syntax.

```
 {
-  "title": "sdf-validation.cddl",
+  "title": "sdf-framework.cddl",
   "$schema": "http://json-schema.org/draft-07/schema#",
   "$ref": "#/definitions/sdf-syntax",
   "definitions": {
     "sdf-syntax": {
       "type": "object",
       "properties": {
         "info": {
           "$ref": "#/definitions/sdfinfo"
         },
         "namespace": {
           "type": "object",
           "additionalProperties": {
             "type": "string"
           }
         },
         "defaultNamespace": {
           "type": "string"
         },
         "sdfThing": {
           "type": "object",
           "additionalProperties": {
             "$ref": "#/definitions/thingqualities"
           }
         },
         "sdfProduct": {
           "type": "object",
           "additionalProperties": {
             "$ref": "#/definitions/productqualities"
           }
         },
         "sdfObject": {
           "type": "object",
           "additionalProperties": {
             "$ref": "#/definitions/objectqualities"
           }
         },
         "sdfProperty": {
           "type": "object",
           "additionalProperties": {
             "$ref": "#/definitions/propertyqualities"
           }
         },
         "sdfAction": {
           "type": "object",
           "additionalProperties": {
             "$ref": "#/definitions/actionqualities"
```

```
          }
        },
        "sdfEvent": {
          "type": "object",
          "additionalProperties": {
            "$ref": "#/definitions/eventqualities"
          }
        },
        "sdfData": {
          "type": "object",
          "additionalProperties": {
            "$ref": "#/definitions/dataqualities"
          }
        }
      },
-     "additionalProperties": false
+     "additionalProperties": {
+     }
    },
    "sdfinfo": {
      "type": "object",
      "properties": {
        "title": {
          "type": "string"
        },
        "version": {
          "type": "string"
        },
        "copyright": {
          "type": "string"
        },
        "license": {
          "type": "string"
        }
      },
-     "additionalProperties": false
+     "additionalProperties": {
+     }
    },
    "thingqualities": {
      "type": "object",
      "properties": {
        "description": {
          "type": "string"
        },
        "label": {
          "type": "string"
        },
        "$comment": {
```

```
            "type": "string"
          },
          "sdfRef": {
            "$ref": "#/definitions/sdf-pointer"
          },
          "sdfRequired": {
            "$ref": "#/definitions/pointer-list"
          },
          "sdfObject": {
            "type": "object",
            "additionalProperties": {
              "$ref": "#/definitions/objectqualities"
            }
          },
          "sdfThing": {
            "type": "object",
            "additionalProperties": {
              "$ref": "#/definitions/thingqualities"
            }
          }
        },
-       "additionalProperties": false
+       "additionalProperties": {
+       }
      },
      "sdf-pointer": {
        "type": "string"
      },
      "pointer-list": {
        "type": "array",
        "items": {
          "$ref": "#/definitions/sdf-pointer"
        }
      },
      "objectqualities": {
        "type": "object",
        "properties": {
          "description": {
            "type": "string"
          },
          "label": {
            "type": "string"
          },
          "$comment": {
            "type": "string"
          },
          "sdfRef": {
            "$ref": "#/definitions/sdf-pointer"
          },
```

```
          "sdfRequired": {
            "$ref": "#/definitions/pointer-list"
          },
          "minItems": {
            "type": "number"
          },
          "maxItems": {
            "type": "number"
          },
          "sdfProperty": {
            "type": "object",
            "additionalProperties": {
              "$ref": "#/definitions/propertyqualities"
            }
          },
          "sdfAction": {
            "type": "object",
            "additionalProperties": {
              "$ref": "#/definitions/actionqualities"
            }
          },
          "sdfEvent": {
            "type": "object",
            "additionalProperties": {
              "$ref": "#/definitions/eventqualities"
            }
          },
          "sdfData": {
            "type": "object",
            "additionalProperties": {
              "$ref": "#/definitions/dataqualities"
            }
          }
        },
-       "additionalProperties": false
+       "additionalProperties": {
+       }
      },
      "propertyqualities": {
        "$ref": "#/definitions/dataqualities"
      },
      "dataqualities": {
        "anyOf": [
          {
            "type": "object",
            "properties": {
              "type": {
                "type": "string",
                "enum": [
```

```
                    "number",
                    "string",
                    "boolean",
                    "integer",
                    "array"
                  ]
                },
                "enum": {
                  "type": "array",
                  "items": {
-                   "type": "string"
+                   "$ref": "#/definitions/allowed-types"
                  },
                  "minItems": 1
                },
                "const": {
                  "$ref": "#/definitions/allowed-types"
                },
                "default": {
                  "$ref": "#/definitions/allowed-types"
                },
                "minimum": {
                  "type": "number"
                },
                "maximum": {
                  "type": "number"
                },
                "exclusiveMinimum": {
                  "anyOf": [
                    {
                      "type": "boolean"
                    },
                    {
                      "type": "number"
                    }
                  ]
                },
                "exclusiveMaximum": {
                  "anyOf": [
                    {
                      "type": "boolean"
                    },
                    {
                      "type": "number"
                    }
                  ]
                },
                "multipleOf": {
                  "type": "number"
```

```
                },
                "minLength": {
                  "type": "number"
                },
                "maxLength": {
                  "type": "number"
                },
                "pattern": {
                  "type": "string"
                },
                "format": {
-                 "type": "string",
-                 "enum": [
-                   "date-time",
-                   "date",
-                   "time",
-                   "uri",
-                   "uri-reference",
-                   "uuid"
+                 "anyOf": [
+                   {
+                     "type": "string",
+                     "const": "date-time"
+                   },
+                   {
+                     "type": "string",
+                     "const": "date"
+                   },
+                   {
+                     "type": "string",
+                     "const": "time"
+                   },
+                   {
+                     "type": "string",
+                     "const": "uri"
+                   },
+                   {
+                     "type": "string",
+                     "const": "uri-reference"
+                   },
+                   {
+                     "type": "string",
+                     "const": "uuid"
+                   },
+                   {
+                     "type": "string"
+                   }
                  ]
                },
```

```
            "minItems": {
              "type": "number"
            },
            "maxItems": {
              "type": "number"
            },
            "uniqueItems": {
              "type": "boolean"
            },
            "items": {
              "anyOf": [
                {
                  "type": "object",
                  "properties": {
                    "type": {
                      "type": "string",
                      "enum": [
                        "number",
                        "string",
                        "boolean",
                        "integer"
                      ]
                    },
                    "sdfRef": {
                      "$ref": "#/definitions/sdf-pointer"
                    },
                    "description": {
                      "type": "string"
                    },
                    "$comment": {
                      "type": "string"
                    },
                    "minimum": {
                      "type": "number"
                    },
                    "maximum": {
                      "type": "number"
                    },
                    "enum": {
                      "type": "array",
-                     "items": {
-                       "type": "string"
-                     },
                      "minItems": 1
                    },
                    "format": {
                      "type": "string"
                    },
                    "minLength": {
```

```
                        "type": "number"
                      },
                      "maxLength": {
                        "type": "number"
                      }
                    },
-                   "additionalProperties": false
+                   "additionalProperties": {
+                   }
                  },
                  {
                    "type": "object",
                    "properties": {
                      "type": {
                        "type": "string",
                        "const": "object"
                      },
                      "required": {
                        "type": "array",
                        "items": {
                          "type": "string"
                        },
                        "minItems": 1
                      },
                      "properties": {
                        "type": "object",
                        "additionalProperties": {
                          "$ref": "#/definitions/dataqualities"
                        }
                      },
                      "sdfRef": {
                        "$ref": "#/definitions/sdf-pointer"
                      },
                      "description": {
                        "type": "string"
                      },
                      "$comment": {
                        "type": "string"
                      },
                      "minimum": {
                        "type": "number"
                      },
                      "maximum": {
                        "type": "number"
                      },
                      "enum": {
                        "type": "array",
-                       "items": {
-                         "type": "string"
```

```
-                       },
                        "minItems": 1
                      },
                      "format": {
                        "type": "string"
                      },
                      "minLength": {
                        "type": "number"
                      },
                      "maxLength": {
                        "type": "number"
                      }
                    },
-                   "additionalProperties": false
+                   "additionalProperties": {
+                   }
                  },
                  {
                    "type": "object",
                    "properties": {
                      "sdfChoice": {
                        "type": "object",
                        "additionalProperties": {
                          "$ref": "#/definitions/dataqualities"
                        }
                      },
                      "sdfRef": {
                        "$ref": "#/definitions/sdf-pointer"
                      },
                      "description": {
                        "type": "string"
                      },
                      "$comment": {
                        "type": "string"
                      },
                      "minimum": {
                        "type": "number"
                      },
                      "maximum": {
                        "type": "number"
                      },
                      "enum": {
                        "type": "array",
-                       "items": {
-                         "type": "string"
-                       },
                        "minItems": 1
                      },
                      "format": {
```

```
                      "type": "string"
                    },
                    "minLength": {
                      "type": "number"
                    },
                    "maxLength": {
                      "type": "number"
                    }
                  },
-                 "additionalProperties": false
+                 "additionalProperties": {
+                 }
+               },
+               {
+                 "type": "object",
+                 "properties": {
+                   "type": {
+                     "type": "string"
+                   },
+                   "sdfRef": {
+                     "$ref": "#/definitions/sdf-pointer"
+                   },
+                   "description": {
+                     "type": "string"
+                   },
+                   "$comment": {
+                     "type": "string"
+                   },
+                   "minimum": {
+                     "type": "number"
+                   },
+                   "maximum": {
+                     "type": "number"
+                   },
+                   "enum": {
+                     "type": "array",
+                     "minItems": 1
+                   },
+                   "format": {
+                     "type": "string"
+                   },
+                   "minLength": {
+                     "type": "number"
+                   },
+                   "maxLength": {
+                     "type": "number"
+                   }
+                 },
+                 "additionalProperties": {
```

```
+                 }
                }
              ]
            },
            "description": {
              "type": "string"
            },
            "label": {
              "type": "string"
            },
            "$comment": {
              "type": "string"
            },
            "sdfRef": {
              "$ref": "#/definitions/sdf-pointer"
            },
            "sdfRequired": {
              "$ref": "#/definitions/pointer-list"
            },
+           "units": {
+             "type": "string"
+           },
            "unit": {
              "type": "string"
            },
+           "scaleMinimum": {
+             "type": "number"
+           },
+           "scaleMaximum": {
+             "type": "number"
+           },
            "observable": {
              "type": "boolean"
            },
            "readable": {
              "type": "boolean"
            },
            "writable": {
              "type": "boolean"
            },
            "nullable": {
              "type": "boolean"
            },
+           "subtype": {
+             "anyOf": [
+               {
+                 "type": "string",
+                 "const": "byte-string"
+               },
```

```
+                {
+                  "type": "string",
+                  "const": "unix-time"
+                },
+                {
+                  "type": "string"
+                }
+              ]
+            },
             "sdfType": {
-              "type": "string",
-              "enum": [
-                "byte-string",
-                "unix-time"
+              "anyOf": [
+                {
+                  "type": "string",
+                  "const": "byte-string"
+                },
+                {
+                  "type": "string",
+                  "const": "unix-time"
+                },
+                {
+                  "type": "string"
+                }
               ]
             },
             "contentFormat": {
               "type": "string"
             }
           },
-          "additionalProperties": false
+          "additionalProperties": {
+          }
         },
         {
           "type": "object",
           "properties": {
             "type": {
               "type": "string",
               "const": "object"
             },
             "required": {
               "type": "array",
               "items": {
                 "type": "string"
               },
               "minItems": 1
```

```
              },
              "properties": {
                "type": "object",
                "additionalProperties": {
                  "$ref": "#/definitions/dataqualities"
                }
              },
              "enum": {
                "type": "array",
                "items": {
-                 "type": "string"
+                 "$ref": "#/definitions/allowed-types"
                },
                "minItems": 1
              },
              "const": {
                "$ref": "#/definitions/allowed-types"
              },
              "default": {
                "$ref": "#/definitions/allowed-types"
              },
              "minimum": {
                "type": "number"
              },
              "maximum": {
                "type": "number"
              },
              "exclusiveMinimum": {
                "anyOf": [
                  {
                    "type": "boolean"
                  },
                  {
                    "type": "number"
                  }
                ]
              },
              "exclusiveMaximum": {
                "anyOf": [
                  {
                    "type": "boolean"
                  },
                  {
                    "type": "number"
                  }
                ]
              },
              "multipleOf": {
                "type": "number"
```

```
                },
                "minLength": {
                  "type": "number"
                },
                "maxLength": {
                  "type": "number"
                },
                "pattern": {
                  "type": "string"
                },
                "format": {
-                 "type": "string",
-                 "enum": [
-                   "date-time",
-                   "date",
-                   "time",
-                   "uri",
-                   "uri-reference",
-                   "uuid"
+                 "anyOf": [
+                   {
+                     "type": "string",
+                     "const": "date-time"
+                   },
+                   {
+                     "type": "string",
+                     "const": "date"
+                   },
+                   {
+                     "type": "string",
+                     "const": "time"
+                   },
+                   {
+                     "type": "string",
+                     "const": "uri"
+                   },
+                   {
+                     "type": "string",
+                     "const": "uri-reference"
+                   },
+                   {
+                     "type": "string",
+                     "const": "uuid"
+                   },
+                   {
+                     "type": "string"
+                   }
                  ]
                },
```

```
            "minItems": {
              "type": "number"
            },
            "maxItems": {
              "type": "number"
            },
            "uniqueItems": {
              "type": "boolean"
            },
            "items": {
              "anyOf": [
                {
                  "type": "object",
                  "properties": {
                    "type": {
                      "type": "string",
                      "enum": [
                        "number",
                        "string",
                        "boolean",
                        "integer"
                      ]
                    },
                    "sdfRef": {
                      "$ref": "#/definitions/sdf-pointer"
                    },
                    "description": {
                      "type": "string"
                    },
                    "$comment": {
                      "type": "string"
                    },
                    "minimum": {
                      "type": "number"
                    },
                    "maximum": {
                      "type": "number"
                    },
                    "enum": {
                      "type": "array",
-                     "items": {
-                       "type": "string"
-                     },
                      "minItems": 1
                    },
                    "format": {
                      "type": "string"
                    },
                    "minLength": {
```

```
                        "type": "number"
                      },
                      "maxLength": {
                        "type": "number"
                      }
                    },
-                   "additionalProperties": false
+                   "additionalProperties": {
+                   }
                  },
                  {
                    "type": "object",
                    "properties": {
                      "type": {
                        "type": "string",
                        "const": "object"
                      },
                      "required": {
                        "type": "array",
                        "items": {
                          "type": "string"
                        },
                        "minItems": 1
                      },
                      "properties": {
                        "type": "object",
                        "additionalProperties": {
                          "$ref": "#/definitions/dataqualities"
                        }
                      },
                      "sdfRef": {
                        "$ref": "#/definitions/sdf-pointer"
                      },
                      "description": {
                        "type": "string"
                      },
                      "$comment": {
                        "type": "string"
                      },
                      "minimum": {
                        "type": "number"
                      },
                      "maximum": {
                        "type": "number"
                      },
                      "enum": {
                        "type": "array",
-                       "items": {
-                         "type": "string"
```

```
-                       },
                        "minItems": 1
                      },
                      "format": {
                        "type": "string"
                      },
                      "minLength": {
                        "type": "number"
                      },
                      "maxLength": {
                        "type": "number"
                      }
                    },
-                   "additionalProperties": false
+                   "additionalProperties": {
+                   }
                  },
                  {
                    "type": "object",
                    "properties": {
                      "sdfChoice": {
                        "type": "object",
                        "additionalProperties": {
                          "$ref": "#/definitions/dataqualities"
                        }
                      },
                      "sdfRef": {
                        "$ref": "#/definitions/sdf-pointer"
                      },
                      "description": {
                        "type": "string"
                      },
                      "$comment": {
                        "type": "string"
                      },
                      "minimum": {
                        "type": "number"
                      },
                      "maximum": {
                        "type": "number"
                      },
                      "enum": {
                        "type": "array",
-                       "items": {
-                         "type": "string"
-                       },
                        "minItems": 1
                      },
                      "format": {
```

```
                      "type": "string"
                    },
                    "minLength": {
                      "type": "number"
                    },
                    "maxLength": {
                      "type": "number"
                    }
                  },
-                 "additionalProperties": false
+                 "additionalProperties": {
+                 }
+               },
+               {
+                 "type": "object",
+                 "properties": {
+                   "type": {
+                     "type": "string"
+                   },
+                   "sdfRef": {
+                     "$ref": "#/definitions/sdf-pointer"
+                   },
+                   "description": {
+                     "type": "string"
+                   },
+                   "$comment": {
+                     "type": "string"
+                   },
+                   "minimum": {
+                     "type": "number"
+                   },
+                   "maximum": {
+                     "type": "number"
+                   },
+                   "enum": {
+                     "type": "array",
+                     "minItems": 1
+                   },
+                   "format": {
+                     "type": "string"
+                   },
+                   "minLength": {
+                     "type": "number"
+                   },
+                   "maxLength": {
+                     "type": "number"
+                   }
+                 },
+                 "additionalProperties": {
```

```
+                 }
                }
              ]
            },
            "description": {
              "type": "string"
            },
            "label": {
              "type": "string"
            },
            "$comment": {
              "type": "string"
            },
            "sdfRef": {
              "$ref": "#/definitions/sdf-pointer"
            },
            "sdfRequired": {
              "$ref": "#/definitions/pointer-list"
            },
+           "units": {
+             "type": "string"
+           },
            "unit": {
              "type": "string"
            },
+           "scaleMinimum": {
+             "type": "number"
+           },
+           "scaleMaximum": {
+             "type": "number"
+           },
            "observable": {
              "type": "boolean"
            },
            "readable": {
              "type": "boolean"
            },
            "writable": {
              "type": "boolean"
            },
            "nullable": {
              "type": "boolean"
            },
+           "subtype": {
+             "anyOf": [
+               {
+                 "type": "string",
+                 "const": "byte-string"
+               },
```

```
+                {
+                  "type": "string",
+                  "const": "unix-time"
+                },
+                {
+                  "type": "string"
+                }
+              ]
+            },
             "sdfType": {
-              "type": "string",
-              "enum": [
-                "byte-string",
-                "unix-time"
+              "anyOf": [
+                {
+                  "type": "string",
+                  "const": "byte-string"
+                },
+                {
+                  "type": "string",
+                  "const": "unix-time"
+                },
+                {
+                  "type": "string"
+                }
               ]
             },
             "contentFormat": {
               "type": "string"
             }
           },
-          "additionalProperties": false
+          "additionalProperties": {
+          }
         },
         {
           "type": "object",
           "properties": {
             "sdfChoice": {
               "type": "object",
               "additionalProperties": {
                 "$ref": "#/definitions/dataqualities"
               }
             },
             "enum": {
               "type": "array",
               "items": {
-                "type": "string"
```

```
+                 "$ref": "#/definitions/allowed-types"
                  },
                  "minItems": 1
                },
                "const": {
                  "$ref": "#/definitions/allowed-types"
                },
                "default": {
                  "$ref": "#/definitions/allowed-types"
                },
                "minimum": {
                  "type": "number"
                },
                "maximum": {
                  "type": "number"
                },
                "exclusiveMinimum": {
                  "anyOf": [
                    {
                      "type": "boolean"
                    },
                    {
                      "type": "number"
                    }
                  ]
                },
                "exclusiveMaximum": {
                  "anyOf": [
                    {
                      "type": "boolean"
                    },
                    {
                      "type": "number"
                    }
                  ]
                },
                "multipleOf": {
                  "type": "number"
                },
                "minLength": {
                  "type": "number"
                },
                "maxLength": {
                  "type": "number"
                },
                "pattern": {
                  "type": "string"
                },
                "format": {
```

```
-               "type": "string",
-               "enum": [
-                 "date-time",
-                 "date",
-                 "time",
-                 "uri",
-                 "uri-reference",
-                 "uuid"
+               "anyOf": [
+                 {
+                   "type": "string",
+                   "const": "date-time"
+                 },
+                 {
+                   "type": "string",
+                   "const": "date"
+                 },
+                 {
+                   "type": "string",
+                   "const": "time"
+                 },
+                 {
+                   "type": "string",
+                   "const": "uri"
+                 },
+                 {
+                   "type": "string",
+                   "const": "uri-reference"
+                 },
+                 {
+                   "type": "string",
+                   "const": "uuid"
+                 },
+                 {
+                   "type": "string"
+                 }
               ]
             },
             "minItems": {
               "type": "number"
             },
             "maxItems": {
               "type": "number"
             },
             "uniqueItems": {
               "type": "boolean"
             },
             "items": {
               "anyOf": [
```

```
                    {
                      "type": "object",
                      "properties": {
                        "type": {
                          "type": "string",
                          "enum": [
                            "number",
                            "string",
                            "boolean",
                            "integer"
                          ]
                        },
                        "sdfRef": {
                          "$ref": "#/definitions/sdf-pointer"
                        },
                        "description": {
                          "type": "string"
                        },
                        "$comment": {
                          "type": "string"
                        },
                        "minimum": {
                          "type": "number"
                        },
                        "maximum": {
                          "type": "number"
                        },
                        "enum": {
                          "type": "array",
-                         "items": {
-                           "type": "string"
-                         },
                          "minItems": 1
                        },
                        "format": {
                          "type": "string"
                        },
                        "minLength": {
                          "type": "number"
                        },
                        "maxLength": {
                          "type": "number"
                        }
                      },
-                     "additionalProperties": false
+                     "additionalProperties": {
+                     }
                    },
                    {
```

```
                    "type": "object",
                    "properties": {
                      "type": {
                        "type": "string",
                        "const": "object"
                      },
                      "required": {
                        "type": "array",
                        "items": {
                          "type": "string"
                        },
                        "minItems": 1
                      },
                      "properties": {
                        "type": "object",
                        "additionalProperties": {
                          "$ref": "#/definitions/dataqualities"
                        }
                      },
                      "sdfRef": {
                        "$ref": "#/definitions/sdf-pointer"
                      },
                      "description": {
                        "type": "string"
                      },
                      "$comment": {
                        "type": "string"
                      },
                      "minimum": {
                        "type": "number"
                      },
                      "maximum": {
                        "type": "number"
                      },
                      "enum": {
                        "type": "array",
-                       "items": {
-                         "type": "string"
-                       },
                        "minItems": 1
                      },
                      "format": {
                        "type": "string"
                      },
                      "minLength": {
                        "type": "number"
                      },
                      "maxLength": {
                        "type": "number"
```

```
                      }
                    },
-                   "additionalProperties": false
+                   "additionalProperties": {
+                   }
                  },
                  {
                    "type": "object",
                    "properties": {
                      "sdfChoice": {
                        "type": "object",
                        "additionalProperties": {
                          "$ref": "#/definitions/dataqualities"
                        }
                      },
                      "sdfRef": {
                        "$ref": "#/definitions/sdf-pointer"
                      },
                      "description": {
                        "type": "string"
                      },
                      "$comment": {
                        "type": "string"
                      },
                      "minimum": {
                        "type": "number"
                      },
                      "maximum": {
                        "type": "number"
                      },
                      "enum": {
                        "type": "array",
-                       "items": {
-                         "type": "string"
-                       },
                        "minItems": 1
                      },
                      "format": {
                        "type": "string"
                      },
                      "minLength": {
                        "type": "number"
                      },
                      "maxLength": {
                        "type": "number"
                      }
                    },
-                   "additionalProperties": false
+                   "additionalProperties": {
```

```
+                  }
+                },
+                {
+                  "type": "object",
+                  "properties": {
+                    "type": {
+                      "type": "string"
+                    },
+                    "sdfRef": {
+                      "$ref": "#/definitions/sdf-pointer"
+                    },
+                    "description": {
+                      "type": "string"
+                    },
+                    "$comment": {
+                      "type": "string"
+                    },
+                    "minimum": {
+                      "type": "number"
+                    },
+                    "maximum": {
+                      "type": "number"
+                    },
+                    "enum": {
+                      "type": "array",
+                      "minItems": 1
+                    },
+                    "format": {
+                      "type": "string"
+                    },
+                    "minLength": {
+                      "type": "number"
+                    },
+                    "maxLength": {
+                      "type": "number"
+                    }
+                  },
+                  "additionalProperties": {
+                  }
                }
              ]
            },
            "description": {
              "type": "string"
            },
            "label": {
              "type": "string"
            },
            "$comment": {
```

```
                "type": "string"
              },
              "sdfRef": {
                "$ref": "#/definitions/sdf-pointer"
              },
              "sdfRequired": {
                "$ref": "#/definitions/pointer-list"
              },
+             "units": {
+               "type": "string"
+             },
              "unit": {
                "type": "string"
              },
+             "scaleMinimum": {
+               "type": "number"
+             },
+             "scaleMaximum": {
+               "type": "number"
+             },
              "observable": {
                "type": "boolean"
              },
              "readable": {
                "type": "boolean"
              },
              "writable": {
                "type": "boolean"
              },
              "nullable": {
                "type": "boolean"
              },
-             "sdfType": {
-               "type": "string",
-               "enum": [
-                 "byte-string",
-                 "unix-time"
-               ]
-             },
+             "subtype": {
+               "anyOf": [
+                 {
+                   "type": "string",
+                   "const": "byte-string"
+                 },
+                 {
+                   "type": "string",
+                   "const": "unix-time"
+                 },
```

```
+                {
+                  "type": "string"
+                }
+              ]
+            },
+            "sdfType": {
+              "anyOf": [
+                {
+                  "type": "string",
+                  "const": "byte-string"
+                },
+                {
+                  "type": "string",
+                  "const": "unix-time"
+                },
+                {
+                  "type": "string"
+                }
+              ]
+            },
             "contentFormat": {
               "type": "string"
             }
           },
-          "additionalProperties": false
+          "additionalProperties": {
+          }
+        },
+        {
+          "type": "object",
+          "properties": {
+            "type": {
+              "type": "string"
+            },
+            "enum": {
+              "type": "array",
+              "items": {
+                "$ref": "#/definitions/allowed-types"
+              },
+              "minItems": 1
+            },
+            "const": {
+              "$ref": "#/definitions/allowed-types"
+            },
+            "default": {
+              "$ref": "#/definitions/allowed-types"
+            },
+            "minimum": {
+              "type": "number"
```

```
+            },
+            "maximum": {
+              "type": "number"
+            },
+            "exclusiveMinimum": {
+              "anyOf": [
+                {
+                  "type": "boolean"
+                },
+                {
+                  "type": "number"
+                }
+              ]
+            },
+            "exclusiveMaximum": {
+              "anyOf": [
+                {
+                  "type": "boolean"
+                },
+                {
+                  "type": "number"
+                }
+              ]
+            },
+            "multipleOf": {
+              "type": "number"
+            },
+            "minLength": {
+              "type": "number"
+            },
+            "maxLength": {
+              "type": "number"
+            },
+            "pattern": {
+              "type": "string"
+            },
+            "format": {
+              "anyOf": [
+                {
+                  "type": "string",
+                  "const": "date-time"
+                },
+                {
+                  "type": "string",
+                  "const": "date"
+                },
+                {
+                  "type": "string",
+                  "const": "time"
```

```
+                 },
+                 {
+                   "type": "string",
+                   "const": "uri"
+                 },
+                 {
+                   "type": "string",
+                   "const": "uri-reference"
+                 },
+                 {
+                   "type": "string",
+                   "const": "uuid"
+                 },
+                 {
+                   "type": "string"
+                 }
+               ]
+             },
+             "minItems": {
+               "type": "number"
+             },
+             "maxItems": {
+               "type": "number"
+             },
+             "uniqueItems": {
+               "type": "boolean"
+             },
+             "items": {
+               "anyOf": [
+                 {
+                   "type": "object",
+                   "properties": {
+                     "type": {
+                       "type": "string",
+                       "enum": [
+                         "number",
+                         "string",
+                         "boolean",
+                         "integer"
+                       ]
+                     },
+                     "sdfRef": {
+                       "$ref": "#/definitions/sdf-pointer"
+                     },
+                     "description": {
+                       "type": "string"
+                     },
+                     "$comment": {
+                       "type": "string"
```

```
+                      },
+                      "minimum": {
+                        "type": "number"
+                      },
+                      "maximum": {
+                        "type": "number"
+                      },
+                      "enum": {
+                        "type": "array",
+                        "minItems": 1
+                      },
+                      "format": {
+                        "type": "string"
+                      },
+                      "minLength": {
+                        "type": "number"
+                      },
+                      "maxLength": {
+                        "type": "number"
+                      }
+                    },
+                    "additionalProperties": {
+                    }
+                  },
+                  {
+                    "type": "object",
+                    "properties": {
+                      "type": {
+                        "type": "string",
+                        "const": "object"
+                      },
+                      "required": {
+                        "type": "array",
+                        "items": {
+                          "type": "string"
+                        },
+                        "minItems": 1
+                      },
+                      "properties": {
+                        "type": "object",
+                        "additionalProperties": {
+                          "$ref": "#/definitions/dataqualities"
+                        }
+                      },
+                      "sdfRef": {
+                        "$ref": "#/definitions/sdf-pointer"
+                      },
+                      "description": {
+                        "type": "string"
```

```
+                      },
+                      "$comment": {
+                        "type": "string"
+                      },
+                      "minimum": {
+                        "type": "number"
+                      },
+                      "maximum": {
+                        "type": "number"
+                      },
+                      "enum": {
+                        "type": "array",
+                        "minItems": 1
+                      },
+                      "format": {
+                        "type": "string"
+                      },
+                      "minLength": {
+                        "type": "number"
+                      },
+                      "maxLength": {
+                        "type": "number"
+                      }
+                    },
+                    "additionalProperties": {
+                    }
+                  },
+                  {
+                    "type": "object",
+                    "properties": {
+                      "sdfChoice": {
+                        "type": "object",
+                        "additionalProperties": {
+                          "$ref": "#/definitions/dataqualities"
+                        }
+                      },
+                      "sdfRef": {
+                        "$ref": "#/definitions/sdf-pointer"
+                      },
+                      "description": {
+                        "type": "string"
+                      },
+                      "$comment": {
+                        "type": "string"
+                      },
+                      "minimum": {
+                        "type": "number"
+                      },
+                      "maximum": {
```

```
+                      "type": "number"
+                    },
+                    "enum": {
+                      "type": "array",
+                      "minItems": 1
+                    },
+                    "format": {
+                      "type": "string"
+                    },
+                    "minLength": {
+                      "type": "number"
+                    },
+                    "maxLength": {
+                      "type": "number"
+                    }
+                  },
+                  "additionalProperties": {
+                  }
+                },
+                {
+                  "type": "object",
+                  "properties": {
+                    "type": {
+                      "type": "string"
+                    },
+                    "sdfRef": {
+                      "$ref": "#/definitions/sdf-pointer"
+                    },
+                    "description": {
+                      "type": "string"
+                    },
+                    "$comment": {
+                      "type": "string"
+                    },
+                    "minimum": {
+                      "type": "number"
+                    },
+                    "maximum": {
+                      "type": "number"
+                    },
+                    "enum": {
+                      "type": "array",
+                      "minItems": 1
+                    },
+                    "format": {
+                      "type": "string"
+                    },
+                    "minLength": {
+                      "type": "number"
```

```
+                    },
+                    "maxLength": {
+                      "type": "number"
+                    }
+                  },
+                  "additionalProperties": {
+                  }
+                }
+              ]
+            },
+            "description": {
+              "type": "string"
+            },
+            "label": {
+              "type": "string"
+            },
+            "$comment": {
+              "type": "string"
+            },
+            "sdfRef": {
+              "$ref": "#/definitions/sdf-pointer"
+            },
+            "sdfRequired": {
+              "$ref": "#/definitions/pointer-list"
+            },
+            "units": {
+              "type": "string"
+            },
+            "unit": {
+              "type": "string"
+            },
+            "scaleMinimum": {
+              "type": "number"
+            },
+            "scaleMaximum": {
+              "type": "number"
+            },
+            "observable": {
+              "type": "boolean"
+            },
+            "readable": {
+              "type": "boolean"
+            },
+            "writable": {
+              "type": "boolean"
+            },
+            "nullable": {
+              "type": "boolean"
+            },
```

```
+            "subtype": {
+              "anyOf": [
+                {
+                  "type": "string",
+                  "const": "byte-string"
+                },
+                {
+                  "type": "string",
+                  "const": "unix-time"
+                },
+                {
+                  "type": "string"
+                }
+              ]
+            },
+            "sdfType": {
+              "anyOf": [
+                {
+                  "type": "string",
+                  "const": "byte-string"
+                },
+                {
+                  "type": "string",
+                  "const": "unix-time"
+                },
+                {
+                  "type": "string"
+                }
+              ]
+            },
+            "contentFormat": {
+              "type": "string"
+            }
+          },
+          "additionalProperties": {
+          }
          }
        ]
      },
      "allowed-types": {
        "anyOf": [
          {
            "type": "number"
          },
          {
            "type": "string"
          },
          {
            "type": "boolean"
```

```
          },
          {
            "type": "null"
          },
          {
            "type": "array",
            "items": {
              "type": "number"
            }
          },
          {
            "type": "array",
            "items": {
              "type": "string"
            }
          },
          {
            "type": "array",
            "items": {
              "type": "boolean"
            }
          },
          {
            "type": "object",
            "additionalProperties": {
            }
+         },
+         {
          }
        ]
      },
      "actionqualities": {
        "type": "object",
        "properties": {
          "description": {
            "type": "string"
          },
          "label": {
            "type": "string"
          },
          "$comment": {
            "type": "string"
          },
          "sdfRef": {
            "$ref": "#/definitions/sdf-pointer"
          },
          "sdfRequired": {
            "$ref": "#/definitions/pointer-list"
          },
```

```
          "sdfInputData": {
            "$ref": "#/definitions/parameter-list"
          },
+         "sdfRequiredInputData": {
+           "$ref": "#/definitions/pointer-list"
+         },
          "sdfOutputData": {
            "$ref": "#/definitions/parameter-list"
          },
          "sdfData": {
            "type": "object",
            "additionalProperties": {
              "$ref": "#/definitions/dataqualities"
            }
          }
        },
-       "additionalProperties": false
+       "additionalProperties": {
+       }
      },
      "parameter-list": {
-       "$ref": "#/definitions/dataqualities"
+       "anyOf": [
+         {
+           "$ref": "#/definitions/pointer-list"
+         },
+         {
+           "$ref": "#/definitions/dataqualities"
+         }
+       ]
      },
      "eventqualities": {
        "type": "object",
        "properties": {
          "description": {
            "type": "string"
          },
          "label": {
            "type": "string"
          },
          "$comment": {
            "type": "string"
          },
          "sdfRef": {
            "$ref": "#/definitions/sdf-pointer"
          },
          "sdfRequired": {
            "$ref": "#/definitions/pointer-list"
          },
```

```
        "sdfOutputData": {
          "$ref": "#/definitions/parameter-list"
        },
        "sdfData": {
          "type": "object",
          "additionalProperties": {
            "$ref": "#/definitions/dataqualities"
          }
        }
      },
-     "additionalProperties": false
+     "additionalProperties": {
+     }
    },
    "productqualities": {
      "$ref": "#/definitions/thingqualities"
    }
  }
}
```

Appendix C. Data Qualities inspired by json-schema.org

Data qualities define data used in SDF affordances at an information model level. A popular way to describe JSON data at a data model level is proposed by a number of drafts on json-schema.org (which collectively are abbreviated JSO here)); for reference to a popular version we will point here to [I-D.handrews-json-schema-validation]. As the vocabulary used by JSO is familiar to many JSON modellers, the present specification borrows some of the terms and ports their semantics to the information model level needed for SDF.

The main data quality imported is the "type". In SDF, this can take one of six (text string) values, which are discussed in the following subsections (note that the JSO type "null" is not supported as a value of this data quality in SDF).

The additional quality "const" restricts the data to one specific value (given as the value of the const quality).

Similarly, the additional quality "default" provides data that can be used in the absence of the data (given as the value of the const quality); this is mainly documentary and not very well-defined for SDF as no process is defined that would add default values to an instance of something.

C.1. type "number", type "integer"

The types "number" and "integer" are associated with floating point and integer numbers, as they are available in JSON. A type value of integer means that only integer values of JSON numbers can be used (note that 10.0 is an integer value, even if it is in a notation that would also allow non-zero decimal fractions).

The additional data qualities "minimum", "maximum", "exclusiveMinimum", "exclusiveMaximum" provide number values that serve as inclusive/exclusive lower/upper bounds for the number. (Note that the Boolean form of "exclusiveMinimum"/"exclusiveMaximum" found in earlier JSO drafts is not used.)

The data quality "multipleOf" gives a positive number that constrains the data value to be an integer multiple of the number given. (Type "integer" can also be expressed as a "multipleOf" quality of value 1, unless another "multipleOf" quality is present.)

C.2. type "string"

The type "string" is associated with Unicode text string values as they are available in JSON.

The length (as measured in characters) can be constrained by the additional data qualities "minLength" and "maxLength", which are inclusive bounds. Note that the previous version of the present document explained text string length values in bytes, which however is not meaningful unless bound to a specific encoding (which could be UTF-8, if this unusual behavior is to be restored).

The data quality "pattern" takes a string value that is interpreted as an [ECMA-262] regular expression in Unicode mode that constrain the string (note that these are not anchored by default, so unless ^ and $ anchors are employed, ECMA-262 regular expressions match any string that *contains* a match). The JSO proposals acknowledge that regular expression support is rather diverse in various platforms, so the suggestion is to limit them to:

*characters;
*character classes in square brackets, including ranges; their complements;
*simple quantifiers *, +, ?, and range quantifiers {n}, {n,m}, and {n,};
*grouping parentheses;
*the choice operator |;
*and anchors (beginning-of-input ^ and end-of-input $).

Note that this subset is somewhat similar to the subset introduced by iregexps [I-D.bormann-jsonpath-iregexp], which however are

anchored regular expressions, and which include certain backslash escapes for characters and character classes.

The additional data quality "format" can take one of the following values. Note that, at an information model level, the presence of this data quality changes the type from being a simple text string to the abstract meaning of the format given (i.e., the format "date-time" is less about the specific syntax employed in [RFC3339] than about the usage as an absolute point in civil time).

* "date-time", "date", "time": An [RFC3339] date-time, full-date, or full-time, respectively.
* "uri", "uri-reference": An [RFC3986] URI or URI Reference, respectively.
* "uuid": An [RFC4122] UUID.

C.3. type "boolean"

The type "boolean" can take the values "true" or "false".

C.4. type "array"

The type "array" is associated with arrays as they are available in JSON.

The additional quality "items" gives the type that each of the elements of the array must match.

The number of elements in the array can be constrained by the additional data qualities "minItems" and "maxItems", which are inclusive bounds.

The additional data quality "uniqueItems" gives a Boolean value that, if true, requires the elements to be all different.

C.5. type "object"

The type "object" is associated with maps, from strings to values, as they are available in JSON ("objects").

The additional quality "properties" is a map the entries of which describe entries in the specified JSON object: The key gives an allowable map key for the specified JSON object, and the value is a map with a named set of data qualities giving the type for the corresponding value in the specified JSON object.

All entries specified this way are optional, unless they are listed in the value of the additional quality "required", which is an array of string values that give the key names of required entries.

Note that the term "properties" as an additional quality for defining map entries is unrelated to sdfProperty.

C.6. Implementation notes

JSO-based keywords are also used in the specification techniques of a number of ecosystems, but some adjustments may be required.

E.g., [OCF] is based on Swagger 2.0 which appears to be based on "draft-4" [I-D.wright-json-schema] (also called draft-5, but semantically intended to be equivalent to draft-4). The "exclusiveMinimum" and "exclusiveMaximum" keywords use the Boolean form there, so on import to SDF their values have to be replaced by the values of the respective "minimum"/"maximum" keyword, which are themselves then removed; the reverse transformation applies on export.

TBD: add any useful implementation notes we can find for other ecosystems that use JSO.

Acknowledgements

This draft is based on sdf.md and sdf-schema.json in the old one-data-model language repository, as well as Ari Keränen's "alt-schema" from the Ericsson Research ipso-odm repository (which is now under subdirectory sdflint in the one-data model tools repository).

Contributors

Ari Keränen
Ericsson
FI-02420 Jorvas
Finland

Email: ari.keranen@ericsson.com

Wouter van der Beek
Cascoda Ltd.
Threefield House
Threefield Lane
Southampton
United Kingdom

Email: w.vanderbeek@cascoda.com

Authors' Addresses

Michael Koster (editor)
PassiveLogic
524 H Street

Antioch, CA, 94509
United States of America

Phone: +1-707-502-5136
Email: michaeljohnkoster@gmail.com

Carsten Bormann (editor)
Universität Bremen TZI
Postfach 330440
D-28359 Bremen
Germany

Phone: +49-421-218-63921
Email: cabo@tzi.org