ASDF
Internet-Draft
Intended status: Standards Track
Expires: 1 September 2024

M. Koster, Ed.
KTC
C. Bormann, Ed.
Universität Bremen TZI
A. Keränen
Ericsson

29 February 2024

Semantic Definition Format (SDF) for Data and Interactions of Things
draft-ietf-asdf-sdf-18

Abstract

The Semantic Definition Format (SDF) is a format for domain experts to use in the creation and maintenance of data and interaction models that describe Things, i.e., physical objects that are available for interaction over a network. An SDF specification describes definitions of SDF Objects/SDF Things and their associated interactions (Events, Actions, Properties), as well as the Data types for the information exchanged in those interactions. Tools convert this format to database formats and other serializations as needed.

// The present revision (-18) adds security considerations, a few
// editorial cleanups, discusses JSON pointer encodings, and adds
// sockets to the CDDL for easier future extension.

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/. Subscribe at https://www.ietf.org/mailman/listinfo/asdf/.

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

Status of This Memo

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

This Internet-Draft will expire on 1 September 2024.

Table of Contents

1. Introduction 4
1.1. Terminology and Conventions 4
2. Overview 7
2.1. Example Definition 7
2.2. Elements of an SDF model 9
2.2.1. sdfObject 10
2.2.2. sdfProperty 11
2.2.3. sdfAction 12
2.2.4. sdfEvent 12
2.2.5. sdfData 13
2.2.6. sdfThing 13
2.3. Member names: Given Names and Quality Names 14
2.3.1. Given Names and Quality Names 14
2.3.2. Hierarchical Names 14
2.3.3. Extensibility of Given Names and Quality Names . . . 15
3. SDF structure 16
3.1. Information block 16
3.2. Namespaces block 18
3.3. Definitions block 19
3.4. Top-level Affordances and sdfData 21

4. Names and namespaces 21
4.1. Structure 21
4.2. Contributing global names 21
4.3. Referencing global names 22
4.4. sdfRef 23
4.4.1. Resolved models 25
4.5. sdfRequired 26
4.6. Common Qualities 29
4.7. Data Qualities 29
4.7.1. sdfType 31
4.7.2. sdfChoice 32
5. Keywords for definition groups 34
5.1. sdfObject 34
5.2. sdfProperty 35
5.3. sdfAction 36
5.4. sdfEvent 37
5.5. sdfData 37
6. High Level Composition 38
6.1. Paths in the model namespaces 38
6.2. Modular Composition 38
6.2.1. Use of the "sdfRef" keyword to re-use a definition . 39
6.3. sdfThing 40
7. IANA Considerations 41
7.1. Media Type 41
7.2. Content-Format 42
7.3. IETF URN Sub-namespace for Unit Names (urn:ietf:params:unit) 43
7.4. Registries 43
7.4.1. Quality Name Prefixes 43
7.4.2. sdfType Values 44
8. Security Considerations 45
9. References 46
9.1. Normative References 47
9.2. Informative References 49
Appendix A. Formal Syntax of SDF 51
Appendix B. json-schema.org Rendition of SDF Syntax 56
Appendix C. Data Qualities inspired by json-schema.org 98
C.1. type "number", type "integer" 98
C.2. type "string" 98
C.3. type "boolean" 100
C.4. type "array" 100
C.5. type "object" 100
C.6. Implementation notes 100
Appendix D. Composition Examples 101
D.1. Outlet Strip Example 101
D.2. Refrigerator-Freezer Example 101
Acknowledgements 102
Contributors 103

Authors' Addresses 103

1. Introduction

The Semantic Definition Format (SDF) is a format for domain experts to use in the creation and maintenance of data and interaction models that describe Things, i.e., physical objects that are available for interaction over a network. An SDF specification describes definitions of SDF Objects/SDF Things and their associated interactions (Events, Actions, Properties), as well as the Data types for the information exchanged in those interactions. Tools convert this format to database formats and other serializations as needed.

// The present revision (-18) adds security considerations, a few
// editorial cleanups, discusses JSON pointer encodings, and adds
// sockets to the CDDL for easier future extension.

SDF is designed to be an extensible format. The present document constitutes the base specification for SDF; we speak of "base SDF" for short. In addition, SDF extensions can be defined, some of which may make use of extension points specifically defined for this in base SDF. One area for such extensions would be refinements of SDF's abstract interaction models into protocol bindings for specific ecosystems (e.g., [I-D.bormann-asdf-sdf-mapping]). For other extensions, it may be necessary to indicate in the SDF document that a specific extension is in effect; see Section 3.1 for details of the features quality that can be used for such indications. With extension points and feature indications available, base SDF does not define a "version" concept for the SDF format itself (as opposed to version indications within SDF documents indicating their own evolution, see Section 3.1).

1.1. Terminology and Conventions

Thing: A physical item that is also available for interaction over a network.

Grouping: An sdfThing or sdfObject, i.e., (directly or indirectly) a combination of Affordances.

sdfThing: A grouping of Groupings as well as potentially Affordance declarations (Property, Action, and Event declarations).

Affordance: An element of an interface offered for interaction, for

which information is available (directly or indirectly) that indicates how it can or should be used. The term is used here for the digital (network-directed) 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 (JSON 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 JSON map representing the SDF document, and in certain nested definitions, that has a Class Name Keyword as its key and a map of named 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 (current/stored information) on an sdfObject. (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 sdfObject.

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

Object, sdfObject: A grouping containing only Affordance declarations (Property, Action, and Event declarations); the main "atom" of reusable semantics for model construction. sdfObjects are similar to sdfThings but do not allow nesting, i.e., they cannot contain other Groupings (sdfObjects or sdfThings). (Note

that JSON maps are often called JSON objects due to JSON's JavaScript heritage; in the context of SDF, the term Object as the colloquial shorthand for sdfObject, is specifically reserved for the above grouping, even if the type name "object" is 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. Unless it is also a Declaration, a definition just defines a term, it does not create a component item within the enclosing definition.

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

SDF Document: Container for SDF Definitions, together with data about the SDF Document itself (information block). Represented as a JSON text representing a single JSON map, which is built from nested maps.

SDF Model: Definitions and declarations that model the digital interaction opportunities offered by one or more kinds of Things, represented by sdfObjects and sdfThings. An SDF Model can be fully contained in a single SDF Document, or it can be built from an SDF Document that references definitions and declarations from additional SDF documents.

Protocol Binding: A companion document to an SDF Model that defines how to map the abstract concepts in the model into the protocols in use in a specific ecosystem. Might supply URL components, numeric IDs, and similar details. Protocol Bindings are one case of an Augmentation Mechanism.

Augmentation Mechanism: A companion document to a base SDF Model that provides additional information ("augments" the base specification), possibly for use in a specific ecosystem or with a specific protocol ("Protocol Binding"). No specific Augmentation Mechanisms are defined in base SDF. A simple mechanism for such augmentations has been discussed as a "mapping file" [I-D.bormann-asdf-sdf-mapping].

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 sdfObject called "Switch" (Figure 1).

```
{
  "info": {
    "title": "Example document for SDF (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 document

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 Things modeled by this sdfObject. The sdfProperty group lists the property affordances described by the model; these represent various perspectives on the state of the sdfObject. 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.

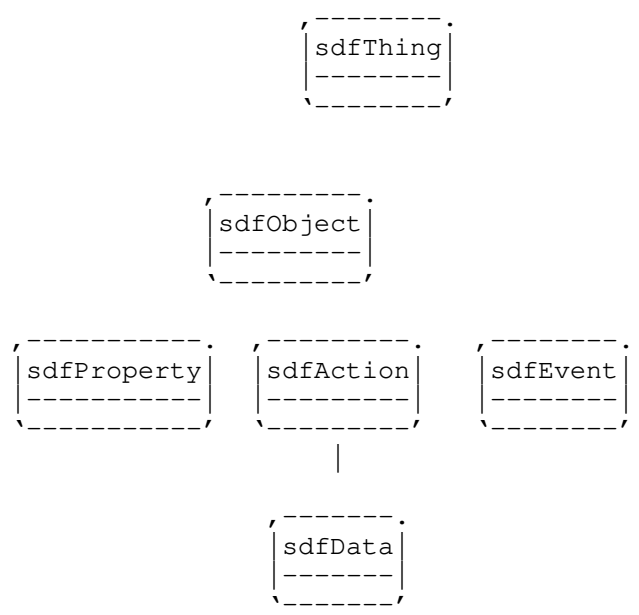

Figure 2: Main classes used in SDF models

The six main Class Name Keywords are discussed below.

2.2.1. sdfObject

sdfObjects, the items listed in an sdfObject definition group, are the main "atom" of reusable semantics for model construction. The concept 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 definition 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.

The presence of one or both of the optional qualities "minItems" and "maxItems" defines the sdfObject as an array, i.e., all the affordances defined for the sdfObject exist a number of times, indexed by a number constrained to be between minItems and maxItems, inclusive, if given. (Note: Setting "minItems" to zero and leaving out "maxItems" puts the minimum constraints on that array.)

2.2.2. sdfProperty

sdfProperty is used to model elements of state within Things modeled by the enclosing grouping.

A named definition entry in an 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 that a Property with the given qualities potentially is present in the containing sdfObject.

For definitions in sdfProperty and sdfData, SDF provides qualities that can constrain the structure and values of data allowed in the interactions modeled by them, as well as qualities that associate semantics to these data, such as engineering units and unit scaling information.

For the data definition within sdfProperty or sdfData, SDF borrows some vocabulary proposed for the drafts 4 [JSO4] [JSO4V] and 7 [JSO7] [JSO7V] of the json-schema.org "JSON Schema" format (collectively called JSO here), enhanced by qualities that are specific to SDF. Details about the JSO-inspired vocabulary are in Appendix C. For base SDF, data are constrained to be of simple types (number, string, Boolean), JSON maps composed of named data, and arrays of these types. Syntax extension points are provided that can be used to provide richer types in a future extension 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.

One idiom for giving an action initiator status and control about the ongoing action is to provide a URI for an ephemeral "action resource" in the sdfAction output data, allowing the action to deliver immediate feedback (including errors that prevent the action from starting) and the action initiator to use the action resource for further observation or modification of the ongoing action (including canceling it). Base SDF does not provide any tailored support for describing such action resources; an extension for modeling links in more detail (for instance, [I-D.bormann-asdf-sdftype-link]) may be all that is needed to fully enable modeling them.

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

Base 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 a Thing modeled by the enclosing sdfObject; 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.

Base 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 do not themselves specify affordances. These definitions 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. The sdfData definitions only spring to life by being referenced in one of these contexts (directly or indirectly via some other sdfData definitions).

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 group enables definition of models for complex devices that will use one or more sdfObject definitions. Like sdfObject, sdfThing groups also allow for including interaction affordances, sdfData, as well as "minItems" and "maxItems" qualities. Therefore, they can be seen as a superset of sdfObject groups, additionally allowing for composition.

As a result, an sdfThing directly or indirectly contains a set of sdfProperty, sdfAction, and sdfEvent definitions that describe the interaction affordances associated with some scope of functionality.

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

2.3. Member names: Given Names and Quality Names

SDF documents are JSON maps that mostly employ JSON maps as member values, which in turn mostly employ JSON maps as their member values, and so on. This nested structure of JSON maps creates a tree, where the edges are the member names (map keys) used in these JSON maps. (In certain cases, where member names are not needed, JSON arrays may be interspersed in this tree.)

2.3.1. Given Names and Quality Names

For any particular JSON map in an SDF document, the set of member names that are used is either of:

* A set of "_Quality Names_", where the entries in the map are Qualities. Quality Names are defined by the present specification and its extensions, together with specific semantics to be associated with the member value given with a certain Quality Name.

* A set of "_Given Names_", where the entries in the map are separate entities (definitions, declarations, etc.) that each have names that are chosen by the SDF document author in order that these names can be employed by a user of that model.

In a path from the root of the tree to any leaf, Quality Names and Given Names roughly alternate (with the information block, Section 3.1, as a prominent exception).

The meaning of the JSON map that is the member value associated with a Given Name is derived from the Quality Name that was used as the member name associated to the parent. In the CDDL grammar given in Appendix A, JSON maps with member names that are Given Names are defined using the CDDL generic rule reference named<membervalues>, where membervalues is in turn the structure of the member values of the JSON map, i.e., the value of the member named by the Given Name. As quality-named maps and given-named maps roughly alternate in a path down the tree, membervalues is usually a map built from Quality Names as keys.

2.3.2. Hierarchical Names

From the outside of a specification, Given Names are usually used as part of a hierarchical name that looks like a JSON pointer [RFC6901], itself generally rooted in (used as the fragment identifier in) an outer namespace that looks like an https:// URL (see Section 4).

As Quality Names and Given Names roughly alternate in a path into the model, the JSON pointer part of the hierarchical name also alternates between Quality Names and Given Names.

Note that the actual Given Names may need to be encoded when specified via the JSON pointer fragment identifier syntax, and that there are two layers of such encoding: tilde encoding of ~ and / as per Section 3 of [RFC6901], and then percent encoding of the tilde-encoded name into a valid URI fragment as per Section 6 of [RFC6901]. For example, when a model is using the Given Name

warning/danger alarm

(with an embedded slash and a space) for an sdfObject, that sdfObject may need to be referenced as

#/sdfObject/warning~1danger%20alarm

To sidestep potential interoperability problems, it is probably wise to avoid characters in Given Names that need such encoding (Quality Names are already defined in such a way that they never do).

2.3.3. Extensibility of Given Names and Quality Names

In SDF, both Quality Names and Given Names are _extension points_. This is more obvious for Quality Names: Extending SDF is mostly done by defining additional qualities. To enable non-conflicting third party extensions to SDF, qualified names (names with an embedded colon) can be used as Quality Names.

A nonqualified Quality Name is composed of ASCII letters, digits, and $ signs, starting with a lower case letter or a $ sign (i.e., using a pattern of "[a-z$][A-Za-z$0-9]*"). Names with $ signs are intended to be used for functions separate from most other names; for instance, in this specification $comment is used for the comment quality (the presence or absence of a $comment quality does not change the meaning of the SDF model). Names that are composed of multiple English words can use the "lowerCamelCase" convention [CamelCase] for indicating the word boundaries; no other use is intended for upper case letters in quality names.

A qualified Quality Name is composed of a Quality Name Prefix, a : (colon) character, and a nonqualified Quality Name. Quality Name Prefixes are registered in the "Quality Name Prefixes" sub-registry in the "SDF Parameters" registry (Section 7.4.1); they are composed of lower case ASCII letters and digits, starting with a lower case ASCII letter (i.e., using a pattern of "[a-z][a-z0-9]*").

Given Names are not restricted by the formal SDF syntax. To enable non-surprising name translations in tools, combinations of ASCII alphanumeric characters and - (ASCII hyphen/minus) are preferred, typically employing kebab-case for names constructed out of multiple words [KebabCase]. ASCII hyphen/minus can then unambiguously be translated to an ASCII _ underscore character and back depending on the programming environment. Some styles also allow a dot . in given names. Given Names are often sufficiently self-explanatory that they can be used in place of the label quality if that is not given. In turn, if a given name turns out too complicated, a more elaborate label can be given and the given name kept simple. Base SDF does not address internationalization of given names.

Further, to enable Given Names to have a more powerful role in building global hierarchical names, an extension is planned that makes use of qualified names for Given Names. So, until that extension is defined, Given Names with (one or more) embedded colons are reserved and MUST NOT be used in an SDF document.

All names in SDF are case-sensitive.

3. SDF structure

SDF definitions are contained in SDF documents, together with data about the SDF document itself (information block). Definitions and declarations from additional SDF documents can be referenced; together with the definitions and declarations in the referencing SDF document they build the SDF model expressed by that SDF document.

Each SDF document is represented as a single JSON map. This map has three blocks: the information block, the namespaces block, and the definitions block.

3.1. Information block

The information block contains generic metadata for the SDF document 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 documents will have policies that only SDF documents 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	A short summary to be displayed in search results, etc.
description	string	no	Long-form text description (no constraints)
version	string	no	The incremental version of the definition
modified	string	no	Time of the latest modification
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
features	array of strings	no	List of extension features used
$comment	string	no	Source code comments only, no semantics

Table 1: Qualities of the Information Block

The version quality is used to indicate version information about the set of definitions in the SDF document. The version is RECOMMENDED 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). This specification does not give a strict definition for the format of the version string but each using system or organization should define internal structure and semantics to the level needed for their use. If no further details are provided, a date-time or full-date in this field can be assumed to indicate the latest update time of the definitions in the SDF document.

The modified quality can be used with a value using [RFC3339] date-time (with Z for time-zone) or full-date format to express time of the latest revision of the definitions.

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. (As an example, for models to be handled by the One Data Model liaison group, this license identifier will typically be "BSD-3-Clause".)

The features quality can be used to list names of critical (i.e., cannot be safely ignored) SDF extension features that need to be understood for the definitions to be properly processed. Extension feature names will be specified in extension documents.

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 document.

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 document does not contribute to a global namespace (all definitions remain local to the model and are not accessible for re-use by other models). 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, 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 definition block they are directly contained in.

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 sdfObject definition is given in Figure 3:

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

Figure 3: Example sdfObject definition

This example defines an sdfObject "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.

Often, definitions are also declarations: the definition of the entry "bar" in the property "foo" means that data corresponding to the "foo" property in a property interaction offered by Thing can have zero or one components modeled by "bar". Entries within sdfProperty, sdfAction, and sdfEvent, in turn within sdfObject or sdfThing entries, are declarations; entries within sdfData are not. Similarly, sdfObject or sdfThing entries within an sdfThing definition specify that the interactions offered by a Thing modeled by this sdfThing include the interactions modeled by the nested sdfObject or sdfThing.

3.4. Top-level Affordances and sdfData

Besides their placement within an sdfObject or sdfThing, affordances (i.e., sdfProperty, sdfAction, and sdfEvent) as well as sdfData can also be placed at the top level of an SDF document. Since they are not associated with an sdfObject or sdfThing, these kinds of definitions are intended to be re-used via the sdfRef mechanism (see Section 4.4).

4. Names and namespaces

SDF documents may contribute to a global namespace, and may reference elements from that global namespace. (An SDF document 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 extensions 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. See also [I-D.bormann-t2trg-deref-id] for a discussion of dereferenceable identifiers.)

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 base SDF, the query part should not be used (it might be used in extensions).

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 document is constructed from a JSON pointer that selects the element defined for this name in the SDF document.

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 document.

For instance, 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 document 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 document, 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 document that does not set a default namespace.

Name references that point outside the current SDF document 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 to 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 and enclosed definitions 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 [JSO7].)

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 or to add new qualities or definitions.

When processing sdfRef, if the target definition contains also sdfRef (i.e., is based on yet another definition), that MUST be processed as well.

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.

Note that the formal syntaxes given in Appendices A and B generally describe the _result_ of applying a merge-patch; the notations are not powerful enough to describe, for instance, the effect of null values given with the sdfRef to remove members of JSON maps from the referenced target. Nonetheless, the syntaxes also give the syntax of the sdfRef itself, which vanishes during the resolution; in many cases therefore even merge-patch inputs will validate with these formal syntaxes.

Given the example (Figure 1), and the following definition:

```
{
  "info": {
    "title": "Example light switch using sdfRef"
  },
  "namespace": {
    "cap": "https://example.com/capability/cap"
  },
  "defaultNamespace": "cap",
  "sdfObject": {
    "BasicSwitch": {
      "sdfRef": "cap:#/sdfObject/Switch",
      "sdfAction": {
        "toggle": null
      }
    }
  }
}
```

The resulting definition of the "BasicSwitch" sdfObject would be identical to the definition of the "Switch" sdfObject except it would not contain the "toggle" Action.

```
{
  "info": {
    "title": "Example light switch using sdfRef"
  },
  "namespace": {
    "cap": "https://example.com/capability/cap"
  },
  "defaultNamespace": "cap",
  "sdfObject": {
    "BasicSwitch": {
      "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."
        }
      }
    }
  }
}
```

4.4.1. Resolved models

A model where all sdfRef references are processed as described in Section 4.4 is called a resolved model.

For example, given the following sdfData definitions:

```
"sdfData": {
  "Coordinate" : {
    "type": "number", "unit": "m"
  },
  "X-Coordinate" : {
    "sdfRef" : "#/sdfData/Coordinate",
    "description":
"Distance from the base of the Thing along the X axis."
  },
  "Non-neg-X-Coordinate" : {
    "sdfRef": "#/sdfData/X-Coordinate",
    "minimum": 0
  }
}
```

After resolving the definitions would look as follows:

```
"sdfData": {
  "Coordinate" : {
    "type": "number", "unit": "m"
  },
  "X-Coordinate" : {
    "description":
"Distance from the base of the Thing along the X axis.",
    "type": "number", "unit": "m"
  },
  "Non-neg-X-Coordinate" : {
    "description":
"Distance from the base of the Thing along the X axis.",
    "minimum": 0, "type": "number", "unit": "m"
  }
}
```

4.5. sdfRequired

The keyword sdfRequired is provided to apply a constraint that defines for which declarations the corresponding data are mandatory in a Thing modeled by the current definition.

The value of sdfRequired is an array of references, each indicating one or more declarations that are mandatory to be represented.

References in this array can be SDF names (JSON Pointers), or one of two abbreviated reference formats:

* a text string with a "referenceable-name", i.e., an affordance name or grouping name. All affordance declarations that are directly (i.e., not nested further in another grouping) in the

same grouping and that carry this name (there can be multiple ones, one per affordance type) are declared to be mandatory to be represented. The same applies for groupings made mandatory within groupings containing them.

* the Boolean value true. The affordance/grouping itself that carries the sdfRequired keyword is declared to be mandatory to be represented.

Note that referenceable-names are not subject to the encoding JSON pointers require as discussed in Section 2.3.2. To ensure that referenceable-names are reliably distinguished from JSON pointers, they are defined such that they cannot contain ":" or "#" characters (see rule referenceable-name in Appendix A). (If these characters are indeed contained in a Given Name, a JSON pointer needs to be formed instead in order to reference it in "sdfRequired", potentially requiring further path elements as well as JSON pointer encoding. The need for this is best avoided by choosing Given Names without these characters.)

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

In Figure 4, the same sdfRequired can also be represented in short form:

```
    "sdfRequired": ["currentTemperature", "overTemperatureEvent"]
```

Or, for instance "overTemperatureEvent" could carry

```
      "overTemperatureEvent": {
        "sdfRequired": [true],
        "...": "..."
      }
```

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	string	long text (no constraints)
label	string	short text (no constraints)
$comment	string	source code comments only, no semantics
sdfRef	sdf-pointer	(see Section 4.4)
sdfRequired	pointer-list	(see Section 4.5, used in affordances or groupings)

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
nullable	boolean	indicates a null value is available for this type	true
contentFormat	string	content type (IANA media type string plus parameters), encoding (note 2)	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 earlier drafts of SDF. 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 (see also [I-D.bormann-t2trg-deref-id]); they may be useful to humans, though. A URI unit name is distinguished from a registered unit name by the presence of a colon; any 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 7.3); URNs from this sub-namespace MUST NOT be used in a unit quality, in favor of simply notating the unit name (such as kg instead of urn:ietf:params:unit:kg).

2. The contentFormat quality follows the Content-Format-Spec as defined in Section 6 of [RFC9193], allowing for expressing both numeric and string based Content-Formats.

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 the "sdfType values" sub-registry in the "SDF Parameters" registry (Section 7.4.2). The sdfType name is composed of lower case ASCII letters, digits, and - (ASCII hyphen/minus) characters, starting with a lower case ASCII letter (i.e., using a pattern of "[a-z][-a-z0-9]*"), typically employing kebab-case for names constructed out of multiple words [KebabCase].

To aid interworking with JSO implementations, it is RECOMMENDED that sdfType is always used in conjunction with the type quality inherited from [JSO7V], in such a way as to yield a common representation of the type's values in JSON.

Values for sdfType that are defined in this specification 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
byte-string	A sequence of zero or more bytes	string	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 base SDF for the 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 the sdfType value registry.

(In earlier drafts of this specification, 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 outer JSON map used to represent the overall choice) and a set of dataqualities (each in an inner JSON map, the value used to represent the individual alternative in the outer JSON map). Dataqualities that are specified at the same level as the sdfChoice apply to all choices in the sdfChoice, except those specific choices where the dataquality is overridden at the choice level.

sdfChoice merges the functions of two constructs found in [JSO7V]:

* enum

  What would have been

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

  in earlier drafts of this specification, 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, what would for instance have been:

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

  in earlier drafts of this specification, 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, for instance, if the actual string value is indeed a crucial element for the data model.)

* anyOf

JSO provides a type union called anyOf, which provides a choice between anonymous alternatives.

What could have been in JSO:

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

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 [JSO7V].

As a simplification for users of SDF models who are accustomed to the JSO 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"}
}
```

In a single definition, the keyword enum cannot be used at the same time as the keyword sdfChoice, as the former is just syntactic sugar for the latter.

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 sdfObject definitions. sdfObject definitions may contain or include definitions of Properties, Actions, Events declared for the sdfObject, as well as data types (sdfData group) to be used in this or other sdfObjects.

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 sdfObject
sdfAction	action	zero or more named action definitions for this sdfObject
sdfEvent	event	zero or more named event definitions for this sdfObject
sdfData	named-sdq	zero or more named data type definitions that might be used in the above
minItems	number	(array) Minimum number of multiplied affordances in array
maxItems	number	(array) Maximum number of multiplied affordances 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 sdfObjects.

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

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

* The ability to enrich and refine a base definition to have product-specific qualities and quality values, such as 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 SDF documents that contribute to 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.

The structure of a path in a namespace is defined by the JSON Pointers to the definitions in the SDF documents in that namespace. For example, if there is an SDF document defining an sdfObject "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 or another SDF document) 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 do not have a defined order, so the SDF processor may see these overrides before seeing the sdfRef.)

Note that if the referenced definition contains qualities or definitions that are not valid in the context where the sdfRef is used (for instance, if an sdfThing definition would be added in an sdfObject definition), the resulting model, when resolved, may be invalid.

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 sdfObject declarations that make up the definition of a single socket of an outlet strip could be encapsulated in an sdfThing, which itself could be used in a declaration in the sdfThing definition for the outlet strip (see Figure 6 in Appendix D.1 for an example SDF model).

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. It can also contain sdfData definitions, as well as declarations of interaction affordances itself, such as a status (on/off) for the refrigerator-freezer as a whole (see Figure 7 in Appendix D.2 for an example SDF model illustrating these aspects).

The qualities of sdfThing are shown in Table 10. Analogous to sdfObject, the presence of one or both of the optional qualities "minItems" and "maxItems" defines the sdfThing as an array.

Quality	Type	Description
(common)		Section 4.6
sdfThing	thing	
sdfObject	object	
sdfProperty	property	zero or more named property definitions for this thing
sdfAction	action	zero or more named action definitions for this thing
sdfEvent	event	zero or more named event definitions for this thing
sdfData	named-sdq	zero or more named data type definitions that might be used in the above
minItems	number	(array) Minimum number of multiplied affordances in array
maxItems	number	(array) Maximum number of multiplied affordances in array

Table 10: Qualities of sdfThing

7. IANA Considerations

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

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: Media Type Registration for SDF

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 and related environments
Fragment identifier considerations: A JSON Pointer fragment identifier may be used, as defined in Section 6 of [RFC6901].
Additional information: Magic number(s): n/a

File extension(s): .sdf.json

Windows Clipboard Name: "Semantic Definition Format (SDF) for Data and Interactions of Things"

Macintosh file type code(s): n/a

Macintosh Universal Type Identifier code: org.ietf.sdf-json conforms to public.text
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. Content-Format

This document adds the following Content-Format to the "CoAP Content-Formats", within the "Constrained RESTful Environments (CoRE) Parameters" registry, where TBD1 comes from the "IETF Review" 256-999 range.

Content Type	Content Coding	ID	Reference
application/sdf+json	-	TBD1	RFC XXXX

Table 12: SDF Content-format Registration

// RFC Ed.: please replace TBD1 with the assigned ID, remove the requested range, and remove this note.
// RFC Ed.: please replace RFC XXXX with this RFC number and remove this note.

7.3. 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 in [IANA.params], 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".

7.4. Registries

IANA is requested to create an "SDF Parameters" registry, with the sub-registries defined in this Section.

7.4.1. Quality Name Prefixes

IANA is requested to create a "Quality Name Prefixes" sub-registry in the "SDF Parameters" registry, with the following template:

Prefix: A name composed of lower case ASCII letters and digits, starting with a lower case ASCII letter (i.e., using a pattern of "[a-z][a-z0-9]*").

Contact: A contact point for the organization that assigns quality names with this prefix.

Quality Name Prefixes are intended to be registered by organizations that intend to define quality names constructed with an organization-specifix prefix (Section 2.3.3).

The registration policy is Expert Review as per Section 4.5 of [BCP26]. The instructions to the Expert are to ascertain that the organization will handle quality names constructed using their prefix in a way that roughly achieves the objectives for an IANA registry that support interoperability of SDF models employing these quality names, including:

* Stability, "stable and permanent";

* Transparency, "readily available", "in sufficient detail" (Section 4.6 of [BCP26]).

The Expert will take into account that other organizations operate in different ways than the IETF, and that as a result some of these overall objectives will be achieved in a different way and to a different level of comfort.

The "Quality Name Prefixes" sub-registry starts out empty.

7.4.2. sdfType Values

IANA is requested to create a "sdfType values" sub-registry in the "SDF Parameters" registry, with the following template:

Name: A name composed of lower case ASCII letters, digits and - (ASCII hyphen/minus) characters, starting with a lower case ASCII letter (i.e., using a pattern of "[a-z][-a-z0-9]*").

Description: A short description of the information model level structure and semantics

type: The value of the quality "type" to be used with this sdfType

JSON Representation A short description of a JSON representation that can be used for this sdfType. This MUST be consistent with the type.

Reference: A more detailed specification of meaning and use of sdfType.

sdfType values are intended to be registered to enable modeling additional SDF-specific types (see Section 4.7.1).

The registration policy is Specification Required as per Section 4.6 of [BCP26]. The instructions to the Expert are to ascertain that the specification provides enough detail to enable interoperability between implementations of the sdfType being registered, and that names are chosen with enough specificity that ecosystem-specific sdfTypes will not be confused with more generally applicable ones.

The initial set of registrations is described in Table 13.

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

Table 13: Initial set of sdfType values

8. Security Considerations

Some wider security considerations applicable to Things are discussed in [RFC8576]. Section 5 of [RFC8610] gives an overview over security considerations that arise when formal description techniques are used to govern interoperability; analogs of these security considerations can apply to SDF.

The security considerations of underlying building blocks such as those detailed in Section 10 of [RFC3629] apply. SDF uses JSON as a representation language; for a number of cases [RFC8259] indicates that implementation behavior for certain constructs allowed by the JSON grammar is unpredictable. Implementations need to be robust against invalid or unpredictable cases on input, preferably by rejecting input that is invalid or that would lead to unpredictable behavior, and need to avoid generating these cases on output.

Implementations of model languages may also exhibit performance-related availability issues when the attacker can control the input, see Section 4.1 of [I-D.ietf-jsonpath-base] for a brief discussion.

SDF may be used in two processes that are often security relevant: model-based _validation_ of data that is intended to be described by SDF models, and model-based _augmentation_ of these data with information obtained from the SDF models that apply.

Implementations need to ascertain the provenance and applicability of the SDF models they employ operationally in such security relevant ways. Implementations that make use of the composition mechanisms defined in this document need to do this for each of the components they combine into the SDF models they employ. Essentially, this process needs to undergo the same care and scrutiny as any other introduction of source code into a build environment; the possibility of supply-chain attacks on the modules imported needs to be considered.

Specifically, implementations might rely on model-based input validation for enforcing certain properties of the data structure ingested (which, if not validated, could lead to malfunctions such as crashes and remote code execution). These implementations need to be particularly careful about the data models they apply, including their provenance and potential changes of these properties that upgrades to the referenced modules may (inadvertently or as part of an attack) cause. More generally speaking, implementations should strive to be robust against expected and unexpected limitations of the model-based input validation mechanisms and their implementations.

Similarly, implementations that rely on model-based augmentation may generate false data from their inputs; this is an integrity violation in any case but also can possibly be exploited for further attacks.

In applications that dynamically acquire models and obtain modules from module references in these, the security considerations of dereferenceable identifiers apply (see [I-D.bormann-t2trg-deref-id] for a more extensive discussion of dereferenceable identifiers).

There may be confidentiality requirements on SDF models, both on their content and on the fact that a specific model is used in a particular Thing or environment. The present specification does not discuss model discovery or define an access control model for SDF models, nor does it define a way to obtain selective disclosure where that may be required. It is likely that these definitions require additional context from underlying ecosystems and the characteristics of the protocols employed there; this is therefore left as future work (e.g., for documents such as [I-D.bormann-asdf-sdf-mapping]).

9. References

9.1. Normative References

[BCP26] Cotton, M., Leiba, B., and T. Narten, "Guidelines for Writing an IANA Considerations Section in RFCs", BCP 26, RFC 8126, DOI 10.17487/RFC8126, June 2017, <https://www.rfc-editor.org/rfc/rfc8126>.

[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/rfc/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/rfc/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/rfc/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/rfc/rfc3553>.

[RFC3629] Yergeau, F., "UTF-8, a transformation format of ISO 10646", STD 63, RFC 3629, DOI 10.17487/RFC3629, November 2003, <https://www.rfc-editor.org/rfc/rfc3629>.

[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/rfc/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/rfc/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/rfc/rfc6901>.

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

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

[RFC8259] Bray, T., Ed., "The JavaScript Object Notation (JSON) Data Interchange Format", STD 90, RFC 8259, DOI 10.17487/RFC8259, December 2017, <https://www.rfc-editor.org/rfc/rfc8259>.

[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/rfc/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/rfc/rfc8610>.

[RFC8798] Bormann, C., "Additional Units for Sensor Measurement Lists (SenML)", RFC 8798, DOI 10.17487/RFC8798, June 2020, <https://www.rfc-editor.org/rfc/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/rfc/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/rfc/rfc9165>.

[RFC9193] Keränen, A. and C. Bormann, "Sensor Measurement Lists (SenML) Fields for Indicating Data Value Content-Format", RFC 9193, DOI 10.17487/RFC9193, June 2022, <https://www.rfc-editor.org/rfc/rfc9193>.

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

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

9.2. Informative References

[CamelCase]
"Camel Case", December 2014, <http://wiki.c2.com/?CamelCase>.

[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-asdf-sdf-mapping]
Bormann, C. and J. Romann, "Semantic Definition Format (SDF): Mapping files", Work in Progress, Internet-Draft, draft-bormann-asdf-sdf-mapping-03, 3 December 2023, <https://datatracker.ietf.org/doc/html/draft-bormann-asdf-sdf-mapping-03>.

[I-D.bormann-asdf-sdftype-link]
Bormann, C., "An sdfType for Links", Work in Progress, Internet-Draft, draft-bormann-asdf-sdftype-link-02, 3 December 2023, <https://datatracker.ietf.org/doc/html/draft-bormann-asdf-sdftype-link-02>.

[I-D.bormann-t2trg-deref-id]
Bormann, C. and C. Amsüss, "The "dereferenceable identifier" pattern", Work in Progress, Internet-Draft, draft-bormann-t2trg-deref-id-02, 19 December 2023, <https://datatracker.ietf.org/doc/html/draft-bormann-t2trg-deref-id-02>.

[I-D.ietf-jsonpath-base]
Gössner, S., Normington, G., and C. Bormann, "JSONPath: Query expressions for JSON", Work in Progress, Internet-Draft, draft-ietf-jsonpath-base-21, 24 September 2023, <https://datatracker.ietf.org/doc/html/draft-ietf-jsonpath-base-21>.

[I-D.irtf-t2trg-rest-iot]
Keränen, 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-13, 25 January 2024, <https://datatracker.ietf.org/doc/html/draft-irtf-t2trg-rest-iot-13>.

[JSO4] Galiegue, F., Zyp, K., and G. Court, "JSON Schema: core definitions and terminology", Work in Progress, Internet-Draft, draft-zyp-json-schema-04, 31 January 2013, <https://datatracker.ietf.org/doc/html/draft-zyp-json-schema-04>. This is the base specification for json-schema.org "draft 4".

[JSO4V] Zyp, K. and G. Court, "JSON Schema: interactive and non interactive validation", Work in Progress, Internet-Draft, draft-fge-json-schema-validation-00, 31 January 2013, <https://datatracker.ietf.org/doc/html/draft-fge-json-schema-validation-00>. This is the validation specification for json-schema.org "draft 4".

[JSO7] Wright, A. and H. Andrews, "JSON Schema: A Media Type for Describing JSON Documents", Work in Progress, Internet-Draft, draft-handrews-json-schema-01, 19 March 2018, <https://datatracker.ietf.org/doc/html/draft-handrews-json-schema-01>. This is the base specification for json-schema.org "draft 7".

[JSO7V] Wright, A., Andrews, H., and G. Luff, "JSON Schema Validation: A Vocabulary for Structural Validation of JSON", Work in Progress, Internet-Draft, draft-handrews-json-schema-validation-01, 19 March 2018, <https://datatracker.ietf.org/doc/html/draft-handrews-json-schema-validation-01>. This is the validation specification for json-schema.org "draft 7".

[KebabCase]
"Kebab Case", August 2014, <http://wiki.c2.com/?KebabCase>.

[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/rfc/rfc8576>.

[RFC9485] Bormann, C. and T. Bray, "I-Regexp: An Interoperable Regular Expression Format", RFC 9485, DOI 10.17487/RFC9485, October 2023, <https://www.rfc-editor.org/rfc/rfc9485>.

[ZCL] "The ZigBee Cluster Library", Elsevier, 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].

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]. Features whose names end in "-ext" indicate extension points for further evolution.

```
start = sdf-syntax

sdf-syntax = {
 ; info will be required in most process policies
 ? info: sdfinfo
 ? namespace: named<text>
 ? defaultNamespace: text
 ; Thing is a composition of objects that work together in some way
 ? sdfThing: named<thingqualities>
 ; Object is a set of Properties, Actions, and Events that together
 ; perform a particular function
 ? sdfObject: named<objectqualities>
 ; Includes Properties, Actions, and Events as well as sdfData
 paedataqualities
 EXTENSION-POINT<"top-ext">
}
```

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

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

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

sdf-pointer = global / same-object / true
global = text .regexp ".*[:#].*" ; rough CURIE or JSON Pointer syntax
same-object = referenceable-name
referenceable-name = text .regexp "[^:#]*"

; per se no point in having an empty list, but used for sdfRequired
; in odmobject-multiple_axis_joystick.sdf.json
pointer-list = [* sdf-pointer]

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

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

arraydefinitionqualities = (
 ? "minItems" => uint
 ? "maxItems" => uint
)
```

```
   paedataqualities = (
    ; Property represents the state of an instance of an object
    ? sdfProperty: named<propertyqualities>
    ; Action invokes an application layer verb associated with an object
    ? sdfAction: named<actionqualities>
    ; Event represents an occurrence of event associated with an object
    ? sdfEvent: named<eventqualities>
    ; Data represents a piece of information that can be the state of a
    ; property or a parameter to an action or a signal in an event
    ? sdfData: named<dataqualities>

   )

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

   ; for single objects, or for arrays of objects
   objectqualities = {
    commonqualities
    paedataqualities
    arraydefinitionqualities
    EXTENSION-POINT<"object-ext">
   }

   parameter-list = dataqualities

   actionqualities = {
    commonqualities
    ? sdfInputData: parameter-list   ; sdfRequiredInputData applies here
    ? sdfOutputData: parameter-list  ; sdfRequired applies here
    ; zero or more named data type definitions that might be used above
    ? sdfData: named<dataqualities>
    EXTENSION-POINT<"action-ext">
   }

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

```
   }

   sdftype-name = text .regexp "[a-z][-a-z0-9]*"       ; EXTENSION-POINT

   dataqualities = {
    commonqualities
    jsonschema
    ? "unit" => text
    ? nullable: bool
    ? "sdfType" => "byte-string" / "unix-time"
               / (sdftype-name .feature "sdftype-ext") ; EXTENSION-POINT
    ? contentFormat: text
    EXTENSION-POINT<"data-ext">
   }

   propertyqualities = {
    ? observable: bool
    ? readable: bool
    ? writable: bool
    ~dataqualities
   }

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

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

   optional-choice = (
    ? (("sdfChoice" => named<dataqualities>)
     // ("enum" => [+ text])) ; limited to text strings
   )

   jsonschema = (
    ? (("type" => "number" / "string" / "boolean" / "integer" / "array")
       // compound-type
       // (type: text .feature "type-ext")             ; EXTENSION-POINT
      )
    ; if present, all other qualities apply to all choices:
    optional-choice
    ; the next three should validate against type:
    ? const: allowed-types
    ? default: allowed-types
```

```
    ; number/integer constraints
    ? minimum: number
    ? maximum: number
    ? exclusiveMinimum: number
    ? exclusiveMaximum: number
    ? multipleOf: number
    ; text string constraints
    ? minLength: uint
    ? maxLength: uint
    ? pattern: text                ; regexp
    ? format: "date-time" / "date" / "time"
              / "uri" / "uri-reference" / "uuid"
              / (text .feature "format-ext")          ; EXTENSION-POINT
    ; array constraints
    ? minItems: uint
    ? maxItems: uint
    ? uniqueItems: bool
    ? items: jso-items
   )

   jso-items = {
        ? sdfRef: sdf-pointer ; import limited to subset allowed here...
        ? description: text   ; long text (no constraints)
        optional-comment
        ; leave commonqualities out for non-complex data types,
        ; but need the above three.
        ; no further nesting: no "array"
        ? ((type: "number" / "string" / "boolean" / "integer")
           // compound-type
           // (type: text .feature "itemtype-ext")     ; EXTENSION-POINT
          )
        ; if present, all other qualities apply to all choices
        optional-choice
        ; jso subset
        ? minimum: number
        ? maximum: number
        ? format: text
        ? minLength: uint
        ? maxLength: uint
        EXTENSION-POINT<"items-ext">
      }

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

   ; RFC 3339 sans time-numoffset, slightly condensed
   rfc3339z = '
      date-fullyear   = 4DIGIT
```

```
      date-month      = 2DIGIT  ; 01-12
      date-mday       = 2DIGIT  ; 01-28, 01-29, 01-30, 01-31 based on
                                ; month/year
      time-hour       = 2DIGIT  ; 00-23
      time-minute     = 2DIGIT  ; 00-59
      time-second     = 2DIGIT  ; 00-58, 00-59, 00-60 based on leap sec
                                ; rules
      time-secfrac    = "." 1*DIGIT
      DIGIT           =  %x30-39 ; 0-9

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

      modified-dt     = full-date ["T" partial-time "Z"]
   '
```

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 [JSO7] [JSO7V].

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 -- Generated: 2024-02-29T07:42:35Z",
+     "title": "sdf-framework.cddl -- Generated: 2024-02-29T07:42:52Z",
      "$schema": "http://json-schema.org/draft-07/schema#",
      "$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": {
                "$ref": "#/definitions/thingqualities"
              }
            },
            "sdfObject": {
              "type": "object",
              "additionalProperties": {
                "$ref": "#/definitions/objectqualities"
              }
            },
            "sdfProperty": {
              "$ref": "#/definitions/sdfProperty-"
            },
            "sdfAction": {
              "$ref": "#/definitions/sdfAction-"
            },
            "sdfEvent": {
              "$ref": "#/definitions/sdfEvent-"
            },
            "sdfData": {
              "$ref": "#/definitions/sdfData-sdfChoice-properties-"
            }
          },
+         "patternProperties": {
+           "^(?:[a-z][a-z0-9]*:)?[a-z$][A-Za-z$0-9]*$": {
+           }
+         },
          "additionalProperties": false
        },
        "sdfinfo": {
          "type": "object",
          "properties": {
            "title": {
              "type": "string"
            },
            "description": {
              "type": "string"
            },
            "version": {
              "type": "string"
            },
            "copyright": {
              "type": "string"
            },
```

```
            "license": {
              "type": "string"
            },
            "modified": {
              "$ref": "#/definitions/modified-date-time"
            },
            "features": {
-             "type": "array",
-             "maxItems": 0
+             "type": "array"
            },
            "$comment": {
              "type": "string"
            }
          },
+       "patternProperties": {
+         "^(?:[a-z][a-z0-9]*:)?[a-z$][A-Za-z$0-9]*$": {
+         }
+       },
          "additionalProperties": false
        },
        "modified-date-time": {
          "type": "string"
        },
        "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": {
                "$ref": "#/definitions/thingqualities"
              }
            },
            "sdfProperty": {
              "$ref": "#/definitions/sdfProperty-"
            },
            "sdfAction": {
              "$ref": "#/definitions/sdfAction-"
            },
            "sdfEvent": {
              "$ref": "#/definitions/sdfEvent-"
            },
            "sdfData": {
              "$ref": "#/definitions/sdfData-sdfChoice-properties-"
            },
            "minItems": {
              "$ref": "#/definitions/uint"
            },
            "maxItems": {
              "$ref": "#/definitions/uint"
            }
          },
+         "patternProperties": {
+           "^(?:[a-z][a-z0-9]*:)?[a-z$][A-Za-z$0-9]*$": {
+           }
+         },
          "additionalProperties": false
        },
        "sdf-pointer": {
          "anyOf": [
            {
              "$ref": "#/definitions/global"
            },
            {
              "$ref": "#/definitions/same-object"
            },
            {
              "$ref": "#/definitions/true"
            }
          ]
        },
        "global": {
          "type": "string",
          "pattern": "^[^\\n\\r]*[:#][^\\n\\r]*$"
        },
```

```
        "same-object": {
          "$ref": "#/definitions/referenceable-name"
        },
        "referenceable-name": {
          "type": "string",
          "pattern": "^[^:#]*$"
        },
        "true": {
          "type": "boolean",
          "const": true
        },
        "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"
            },
            "sdfProperty": {
              "$ref": "#/definitions/sdfProperty-"
            },
            "sdfAction": {
              "$ref": "#/definitions/sdfAction-"
            },
            "sdfEvent": {
              "$ref": "#/definitions/sdfEvent-"
            },
            "sdfData": {
              "$ref": "#/definitions/sdfData-sdfChoice-properties-"
            },
            "minItems": {
```

```
              "$ref": "#/definitions/uint"
            },
            "maxItems": {
              "$ref": "#/definitions/uint"
            }
          },
+         "patternProperties": {
+           "^(?:[a-z][a-z0-9]*:)?[a-z$][A-Za-z$0-9]*$": {
+           }
+         },
          "additionalProperties": false
        },
        "propertyqualities": {
          "anyOf": [
            {
              "type": "object",
+             "patternProperties": {
+               "^(?:[a-z][a-z0-9]*:)?[a-z$][A-Za-z$0-9]*$": {
+               }
+             },
              "properties": {
                "type": {
                  "$ref": "#/definitions/type-"
                },
                "sdfChoice": {
                  "$ref": "#/definitions/sdfData-sdfChoice-properties-"
                },
                "observable": {
                  "type": "boolean"
                },
                "readable": {
                  "type": "boolean"
                },
                "writable": {
                  "type": "boolean"
                },
                "description": {
                  "type": "string"
                },
                "label": {
                  "type": "string"
                },
                "$comment": {
                  "type": "string"
                },
                "sdfRef": {
                  "$ref": "#/definitions/sdf-pointer"
                },
```

```
                "sdfRequired": {
                  "$ref": "#/definitions/pointer-list"
                },
                "const": {
                  "$ref": "#/definitions/allowed-types"
                },
                "default": {
                  "$ref": "#/definitions/allowed-types"
                },
                "minimum": {
                  "type": "number"
                },
                "maximum": {
                  "type": "number"
                },
                "exclusiveMinimum": {
                  "type": "number"
                },
                "exclusiveMaximum": {
                  "type": "number"
                },
                "multipleOf": {
                  "type": "number"
                },
                "minLength": {
                  "$ref": "#/definitions/uint"
                },
                "maxLength": {
                  "$ref": "#/definitions/uint"
                },
                "pattern": {
                  "type": "string"
                },
                "format": {
                  "$ref": "#/definitions/format-"
                },
                "minItems": {
                  "$ref": "#/definitions/uint"
                },
                "maxItems": {
                  "$ref": "#/definitions/uint"
                },
                "uniqueItems": {
                  "type": "boolean"
                },
                "items": {
                  "$ref": "#/definitions/jso-items"
                },
```

```
                "unit": {
                  "type": "string"
                },
                "nullable": {
                  "type": "boolean"
                },
                "sdfType": {
                  "$ref": "#/definitions/sdfType-"
                },
                "contentFormat": {
                  "type": "string"
                }
              },
              "additionalProperties": false
            },
            {
              "type": "object",
+             "patternProperties": {
+               "^(?:[a-z][a-z0-9]*:)?[a-z$][A-Za-z$0-9]*$": {
+               }
+             },
+             "properties": {
+               "type": {
+                 "type": "string",
+                 "const": "object"
+               },
+               "required": {
+                 "type": "array",
+                 "items": {
+                   "type": "string"
+                 },
+                 "minItems": 1
+               },
+               "properties": {
+                 "$ref": "#/definitions/sdfData-sdfChoice-properties-"
+               },
+               "sdfChoice": {
+                 "$ref": "#/definitions/sdfData-sdfChoice-properties-"
+               },
+               "observable": {
+                 "type": "boolean"
+               },
+               "readable": {
+                 "type": "boolean"
+               },
+               "writable": {
+                 "type": "boolean"
+               },
```

```
+            "description": {
+              "type": "string"
+            },
+            "label": {
+              "type": "string"
+            },
+            "$comment": {
+              "type": "string"
+            },
+            "sdfRef": {
+              "$ref": "#/definitions/sdf-pointer"
+            },
+            "sdfRequired": {
+              "$ref": "#/definitions/pointer-list"
+            },
+            "const": {
+              "$ref": "#/definitions/allowed-types"
+            },
+            "default": {
+              "$ref": "#/definitions/allowed-types"
+            },
+            "minimum": {
+              "type": "number"
+            },
+            "maximum": {
+              "type": "number"
+            },
+            "exclusiveMinimum": {
+              "type": "number"
+            },
+            "exclusiveMaximum": {
+              "type": "number"
+            },
+            "multipleOf": {
+              "type": "number"
+            },
+            "minLength": {
+              "$ref": "#/definitions/uint"
+            },
+            "maxLength": {
+              "$ref": "#/definitions/uint"
+            },
+            "pattern": {
+              "type": "string"
+            },
+            "format": {
+              "$ref": "#/definitions/format-"
+            },
```

```
+            "minItems": {
+              "$ref": "#/definitions/uint"
+            },
+            "maxItems": {
+              "$ref": "#/definitions/uint"
+            },
+            "uniqueItems": {
+              "type": "boolean"
+            },
+            "items": {
+              "$ref": "#/definitions/jso-items"
+            },
+            "unit": {
+              "type": "string"
+            },
+            "nullable": {
+              "type": "boolean"
+            },
+            "sdfType": {
+              "$ref": "#/definitions/sdfType-"
+            },
+            "contentFormat": {
+              "type": "string"
+            }
+          },
+          "additionalProperties": false
+        },
+        {
+          "type": "object",
+          "patternProperties": {
+            "^(?:[a-z][a-z0-9]*:)?[a-z$][A-Za-z$0-9]*$": {
+            }
+          },
+          "properties": {
+            "type": {
+              "type": "string"
+            },
+            "sdfChoice": {
+              "$ref": "#/definitions/sdfData-sdfChoice-properties-"
+            },
+            "observable": {
+              "type": "boolean"
+            },
+            "readable": {
+              "type": "boolean"
+            },
+            "writable": {
+              "type": "boolean"
```

```
+            },
+            "description": {
+              "type": "string"
+            },
+            "label": {
+              "type": "string"
+            },
+            "$comment": {
+              "type": "string"
+            },
+            "sdfRef": {
+              "$ref": "#/definitions/sdf-pointer"
+            },
+            "sdfRequired": {
+              "$ref": "#/definitions/pointer-list"
+            },
+            "const": {
+              "$ref": "#/definitions/allowed-types"
+            },
+            "default": {
+              "$ref": "#/definitions/allowed-types"
+            },
+            "minimum": {
+              "type": "number"
+            },
+            "maximum": {
+              "type": "number"
+            },
+            "exclusiveMinimum": {
+              "type": "number"
+            },
+            "exclusiveMaximum": {
+              "type": "number"
+            },
+            "multipleOf": {
+              "type": "number"
+            },
+            "minLength": {
+              "$ref": "#/definitions/uint"
+            },
+            "maxLength": {
+              "$ref": "#/definitions/uint"
+            },
+            "pattern": {
+              "type": "string"
+            },
+            "format": {
+              "$ref": "#/definitions/format-"
```

```
+            },
+            "minItems": {
+              "$ref": "#/definitions/uint"
+            },
+            "maxItems": {
+              "$ref": "#/definitions/uint"
+            },
+            "uniqueItems": {
+              "type": "boolean"
+            },
+            "items": {
+              "$ref": "#/definitions/jso-items"
+            },
+            "unit": {
+              "type": "string"
+            },
+            "nullable": {
+              "type": "boolean"
+            },
+            "sdfType": {
+              "$ref": "#/definitions/sdfType-"
+            },
+            "contentFormat": {
+              "type": "string"
+            }
+          },
+          "additionalProperties": false
+        },
+        {
+          "type": "object",
+          "patternProperties": {
+            "^(?:[a-z][a-z0-9]*:)?[a-z$][A-Za-z$0-9]*$": {
+            }
+          },
+          "properties": {
+            "type": {
+              "$ref": "#/definitions/type-"
+            },
+            "enum": {
+              "type": "array",
+              "items": {
+                "type": "string"
+              },
+              "minItems": 1
+            },
+            "observable": {
+              "type": "boolean"
+            },
```

```
+            "readable": {
+              "type": "boolean"
+            },
+            "writable": {
+              "type": "boolean"
+            },
+            "description": {
+              "type": "string"
+            },
+            "label": {
+              "type": "string"
+            },
+            "$comment": {
+              "type": "string"
+            },
+            "sdfRef": {
+              "$ref": "#/definitions/sdf-pointer"
+            },
+            "sdfRequired": {
+              "$ref": "#/definitions/pointer-list"
+            },
+            "const": {
+              "$ref": "#/definitions/allowed-types"
+            },
+            "default": {
+              "$ref": "#/definitions/allowed-types"
+            },
+            "minimum": {
+              "type": "number"
+            },
+            "maximum": {
+              "type": "number"
+            },
+            "exclusiveMinimum": {
+              "type": "number"
+            },
+            "exclusiveMaximum": {
+              "type": "number"
+            },
+            "multipleOf": {
+              "type": "number"
+            },
+            "minLength": {
+              "$ref": "#/definitions/uint"
+            },
+            "maxLength": {
+              "$ref": "#/definitions/uint"
+            },
```

```
+            "pattern": {
+              "type": "string"
+            },
+            "format": {
+              "$ref": "#/definitions/format-"
+            },
+            "minItems": {
+              "$ref": "#/definitions/uint"
+            },
+            "maxItems": {
+              "$ref": "#/definitions/uint"
+            },
+            "uniqueItems": {
+              "type": "boolean"
+            },
+            "items": {
+              "$ref": "#/definitions/jso-items"
+            },
+            "unit": {
+              "type": "string"
+            },
+            "nullable": {
+              "type": "boolean"
+            },
+            "sdfType": {
+              "$ref": "#/definitions/sdfType-"
+            },
+            "contentFormat": {
+              "type": "string"
+            }
+          },
+          "additionalProperties": false
+        },
+        {
+          "type": "object",
+          "patternProperties": {
+            "^(?:[a-z][a-z0-9]*:)?[a-z$][A-Za-z$0-9]*$": {
+            }
+          },
+          "properties": {
+            "type": {
+              "type": "string",
+              "const": "object"
+            },
+            "required": {
+              "type": "array",
+              "items": {
+                "type": "string"
```

```
+              },
+              "minItems": 1
+            },
+            "properties": {
+              "$ref": "#/definitions/sdfData-sdfChoice-properties-"
+            },
+            "enum": {
+              "type": "array",
+              "items": {
+                "type": "string"
+              },
+              "minItems": 1
+            },
+            "observable": {
+              "type": "boolean"
+            },
+            "readable": {
+              "type": "boolean"
+            },
+            "writable": {
+              "type": "boolean"
+            },
+            "description": {
+              "type": "string"
+            },
+            "label": {
+              "type": "string"
+            },
+            "$comment": {
+              "type": "string"
+            },
+            "sdfRef": {
+              "$ref": "#/definitions/sdf-pointer"
+            },
+            "sdfRequired": {
+              "$ref": "#/definitions/pointer-list"
+            },
+            "const": {
+              "$ref": "#/definitions/allowed-types"
+            },
+            "default": {
+              "$ref": "#/definitions/allowed-types"
+            },
+            "minimum": {
+              "type": "number"
+            },
+            "maximum": {
+              "type": "number"
```

```
+            },
+            "exclusiveMinimum": {
+              "type": "number"
+            },
+            "exclusiveMaximum": {
+              "type": "number"
+            },
+            "multipleOf": {
+              "type": "number"
+            },
+            "minLength": {
+              "$ref": "#/definitions/uint"
+            },
+            "maxLength": {
+              "$ref": "#/definitions/uint"
+            },
+            "pattern": {
+              "type": "string"
+            },
+            "format": {
+              "$ref": "#/definitions/format-"
+            },
+            "minItems": {
+              "$ref": "#/definitions/uint"
+            },
+            "maxItems": {
+              "$ref": "#/definitions/uint"
+            },
+            "uniqueItems": {
+              "type": "boolean"
+            },
+            "items": {
+              "$ref": "#/definitions/jso-items"
+            },
+            "unit": {
+              "type": "string"
+            },
+            "nullable": {
+              "type": "boolean"
+            },
+            "sdfType": {
+              "$ref": "#/definitions/sdfType-"
+            },
+            "contentFormat": {
+              "type": "string"
+            }
+          },
+          "additionalProperties": false
```

```
+         },
+         {
+           "type": "object",
+           "patternProperties": {
+             "^(?:[a-z][a-z0-9]*:)?[a-z$][A-Za-z$0-9]*$": {
+             }
+           },
             "properties": {
               "type": {
-                "type": "string",
-                "const": "object"
+                "type": "string"
               },
-              "required": {
+              "enum": {
                 "type": "array",
                 "items": {
                   "type": "string"
                 },
                 "minItems": 1
               },
-              "properties": {
-                "$ref": "#/definitions/sdfData-sdfChoice-properties-"
-              },
-              "sdfChoice": {
-                "$ref": "#/definitions/sdfData-sdfChoice-properties-"
-              },
               "observable": {
                 "type": "boolean"
               },
               "readable": {
                 "type": "boolean"
               },
               "writable": {
                 "type": "boolean"
               },
               "description": {
                 "type": "string"
               },
               "label": {
                 "type": "string"
               },
               "$comment": {
                 "type": "string"
               },
               "sdfRef": {
                 "$ref": "#/definitions/sdf-pointer"
               },
```

```
                "sdfRequired": {
                  "$ref": "#/definitions/pointer-list"
                },
                "const": {
                  "$ref": "#/definitions/allowed-types"
                },
                "default": {
                  "$ref": "#/definitions/allowed-types"
                },
                "minimum": {
                  "type": "number"
                },
                "maximum": {
                  "type": "number"
                },
                "exclusiveMinimum": {
                  "type": "number"
                },
                "exclusiveMaximum": {
                  "type": "number"
                },
                "multipleOf": {
                  "type": "number"
                },
                "minLength": {
                  "$ref": "#/definitions/uint"
                },
                "maxLength": {
                  "$ref": "#/definitions/uint"
                },
                "pattern": {
                  "type": "string"
                },
                "format": {
                  "$ref": "#/definitions/format-"
                },
                "minItems": {
                  "$ref": "#/definitions/uint"
                },
                "maxItems": {
                  "$ref": "#/definitions/uint"
                },
                "uniqueItems": {
                  "type": "boolean"
                },
                "items": {
                  "$ref": "#/definitions/jso-items"
                },
```

```
                "unit": {
                  "type": "string"
                },
                "nullable": {
                  "type": "boolean"
                },
                "sdfType": {
                  "$ref": "#/definitions/sdfType-"
                },
                "contentFormat": {
                  "type": "string"
                }
              },
              "additionalProperties": false
-           },
+           }
+         ]
+       },
+       "dataqualities": {
+         "anyOf": [
            {
              "type": "object",
+             "patternProperties": {
+               "^(?:[a-z][a-z0-9]*:)?[a-z$][A-Za-z$0-9]*$": {
+               }
+             },
              "properties": {
                "type": {
                  "$ref": "#/definitions/type-"
                },
-               "enum": {
-                 "type": "array",
-                 "items": {
-                   "type": "string"
-                 },
-                 "minItems": 1
-               },
-               "observable": {
-                 "type": "boolean"
-               },
-               "readable": {
-                 "type": "boolean"
-               },
-               "writable": {
-                 "type": "boolean"
+               "sdfChoice": {
+                 "$ref": "#/definitions/sdfData-sdfChoice-properties-"
                },
```

```
"description": {
  "type": "string"
},
"label": {
  "type": "string"
},
"$comment": {
  "type": "string"
},
"sdfRef": {
  "$ref": "#/definitions/sdf-pointer"
},
"sdfRequired": {
  "$ref": "#/definitions/pointer-list"
},
"const": {
  "$ref": "#/definitions/allowed-types"
},
"default": {
  "$ref": "#/definitions/allowed-types"
},
"minimum": {
  "type": "number"
},
"maximum": {
  "type": "number"
},
"exclusiveMinimum": {
  "type": "number"
},
"exclusiveMaximum": {
  "type": "number"
},
"multipleOf": {
  "type": "number"
},
"minLength": {
  "$ref": "#/definitions/uint"
},
"maxLength": {
  "$ref": "#/definitions/uint"
},
"pattern": {
  "type": "string"
},
"format": {
  "$ref": "#/definitions/format-"
},
```

```
                "minItems": {
                  "$ref": "#/definitions/uint"
                },
                "maxItems": {
                  "$ref": "#/definitions/uint"
                },
                "uniqueItems": {
                  "type": "boolean"
                },
                "items": {
                  "$ref": "#/definitions/jso-items"
                },
                "unit": {
                  "type": "string"
                },
                "nullable": {
                  "type": "boolean"
                },
                "sdfType": {
                  "$ref": "#/definitions/sdfType-"
                },
                "contentFormat": {
                  "type": "string"
                }
              },
              "additionalProperties": false
            },
            {
              "type": "object",
+             "patternProperties": {
+               "^(?:[a-z][a-z0-9]*:)?[a-z$][A-Za-z$0-9]*$": {
+               }
+             },
              "properties": {
                "type": {
                  "type": "string",
                  "const": "object"
                },
                "required": {
                  "type": "array",
                  "items": {
                    "type": "string"
                  },
                  "minItems": 1
                },
                "properties": {
                  "$ref": "#/definitions/sdfData-sdfChoice-properties-"
                },
```

```
-             "enum": {
-               "type": "array",
-               "items": {
-                 "type": "string"
-               },
-               "minItems": 1
-             },
-             "observable": {
-               "type": "boolean"
-             },
-             "readable": {
-               "type": "boolean"
-             },
-             "writable": {
-               "type": "boolean"
+             "sdfChoice": {
+               "$ref": "#/definitions/sdfData-sdfChoice-properties-"
              },
              "description": {
                "type": "string"
              },
              "label": {
                "type": "string"
              },
              "$comment": {
                "type": "string"
              },
              "sdfRef": {
                "$ref": "#/definitions/sdf-pointer"
              },
              "sdfRequired": {
                "$ref": "#/definitions/pointer-list"
              },
              "const": {
                "$ref": "#/definitions/allowed-types"
              },
              "default": {
                "$ref": "#/definitions/allowed-types"
              },
              "minimum": {
                "type": "number"
              },
              "maximum": {
                "type": "number"
              },
              "exclusiveMinimum": {
                "type": "number"
              },
```

```
                "exclusiveMaximum": {
                  "type": "number"
                },
                "multipleOf": {
                  "type": "number"
                },
                "minLength": {
                  "$ref": "#/definitions/uint"
                },
                "maxLength": {
                  "$ref": "#/definitions/uint"
                },
                "pattern": {
                  "type": "string"
                },
                "format": {
                  "$ref": "#/definitions/format-"
                },
                "minItems": {
                  "$ref": "#/definitions/uint"
                },
                "maxItems": {
                  "$ref": "#/definitions/uint"
                },
                "uniqueItems": {
                  "type": "boolean"
                },
                "items": {
                  "$ref": "#/definitions/jso-items"
                },
                "unit": {
                  "type": "string"
                },
                "nullable": {
                  "type": "boolean"
                },
                "sdfType": {
                  "$ref": "#/definitions/sdfType-"
                },
                "contentFormat": {
                  "type": "string"
                }
              },
              "additionalProperties": false
-         }
-       ]
-     },
-     "dataqualities": {
```

```
-       "anyOf": [
+         },
          {
            "type": "object",
+           "patternProperties": {
+             "^(?:[a-z][a-z0-9]*:)?[a-z$][A-Za-z$0-9]*$": {
+             }
+           },
            "properties": {
              "type": {
-               "$ref": "#/definitions/type-"
+               "type": "string"
              },
              "sdfChoice": {
                "$ref": "#/definitions/sdfData-sdfChoice-properties-"
              },
              "description": {
                "type": "string"
              },
              "label": {
                "type": "string"
              },
              "$comment": {
                "type": "string"
              },
              "sdfRef": {
                "$ref": "#/definitions/sdf-pointer"
              },
              "sdfRequired": {
                "$ref": "#/definitions/pointer-list"
              },
              "const": {
                "$ref": "#/definitions/allowed-types"
              },
              "default": {
                "$ref": "#/definitions/allowed-types"
              },
              "minimum": {
                "type": "number"
              },
              "maximum": {
                "type": "number"
              },
              "exclusiveMinimum": {
                "type": "number"
              },
              "exclusiveMaximum": {
                "type": "number"
```

```
                },
                "multipleOf": {
                  "type": "number"
                },
                "minLength": {
                  "$ref": "#/definitions/uint"
                },
                "maxLength": {
                  "$ref": "#/definitions/uint"
                },
                "pattern": {
                  "type": "string"
                },
                "format": {
                  "$ref": "#/definitions/format-"
                },
                "minItems": {
                  "$ref": "#/definitions/uint"
                },
                "maxItems": {
                  "$ref": "#/definitions/uint"
                },
                "uniqueItems": {
                  "type": "boolean"
                },
                "items": {
                  "$ref": "#/definitions/jso-items"
                },
                "unit": {
                  "type": "string"
                },
                "nullable": {
                  "type": "boolean"
                },
                "sdfType": {
                  "$ref": "#/definitions/sdfType-"
                },
                "contentFormat": {
                  "type": "string"
                }
              },
              "additionalProperties": false
            },
            {
              "type": "object",
+             "patternProperties": {
+               "^(?:[a-z][a-z0-9]*:)?[a-z$][A-Za-z$0-9]*$": {
+               }
```

```
+           },
            "properties": {
              "type": {
-               "type": "string",
-               "const": "object"
+               "$ref": "#/definitions/type-"
              },
-             "required": {
+             "enum": {
                "type": "array",
                "items": {
                  "type": "string"
                },
                "minItems": 1
              },
-             "properties": {
-               "$ref": "#/definitions/sdfData-sdfChoice-properties-"
-             },
-             "sdfChoice": {
-               "$ref": "#/definitions/sdfData-sdfChoice-properties-"
-             },
              "description": {
                "type": "string"
              },
              "label": {
                "type": "string"
              },
              "$comment": {
                "type": "string"
              },
              "sdfRef": {
                "$ref": "#/definitions/sdf-pointer"
              },
              "sdfRequired": {
                "$ref": "#/definitions/pointer-list"
              },
              "const": {
                "$ref": "#/definitions/allowed-types"
              },
              "default": {
                "$ref": "#/definitions/allowed-types"
              },
              "minimum": {
                "type": "number"
              },
              "maximum": {
                "type": "number"
              },
```

```
            "exclusiveMinimum": {
              "type": "number"
            },
            "exclusiveMaximum": {
              "type": "number"
            },
            "multipleOf": {
              "type": "number"
            },
            "minLength": {
              "$ref": "#/definitions/uint"
            },
            "maxLength": {
              "$ref": "#/definitions/uint"
            },
            "pattern": {
              "type": "string"
            },
            "format": {
              "$ref": "#/definitions/format-"
            },
            "minItems": {
              "$ref": "#/definitions/uint"
            },
            "maxItems": {
              "$ref": "#/definitions/uint"
            },
            "uniqueItems": {
              "type": "boolean"
            },
            "items": {
              "$ref": "#/definitions/jso-items"
            },
            "unit": {
              "type": "string"
            },
            "nullable": {
              "type": "boolean"
            },
            "sdfType": {
              "$ref": "#/definitions/sdfType-"
            },
            "contentFormat": {
              "type": "string"
            }
          },
          "additionalProperties": false
        },
```

```
            {
              "type": "object",
    +         "patternProperties": {
    +           "^(?:[a-z][a-z0-9]*:)?[a-z$][A-Za-z$0-9]*$": {
    +           }
    +         },
              "properties": {
                "type": {
    -             "$ref": "#/definitions/type-"
    +             "type": "string",
    +             "const": "object"
    +           },
    +           "required": {
    +             "type": "array",
    +             "items": {
    +               "type": "string"
    +             },
    +             "minItems": 1
    +           },
    +           "properties": {
    +             "$ref": "#/definitions/sdfData-sdfChoice-properties-"
                },
                "enum": {
                  "type": "array",
                  "items": {
                    "type": "string"
                  },
                  "minItems": 1
                },
                "description": {
                  "type": "string"
                },
                "label": {
                  "type": "string"
                },
                "$comment": {
                  "type": "string"
                },
                "sdfRef": {
                  "$ref": "#/definitions/sdf-pointer"
                },
                "sdfRequired": {
                  "$ref": "#/definitions/pointer-list"
                },
                "const": {
                  "$ref": "#/definitions/allowed-types"
                },
                "default": {
```

```
      "$ref": "#/definitions/allowed-types"
    },
    "minimum": {
      "type": "number"
    },
    "maximum": {
      "type": "number"
    },
    "exclusiveMinimum": {
      "type": "number"
    },
    "exclusiveMaximum": {
      "type": "number"
    },
    "multipleOf": {
      "type": "number"
    },
    "minLength": {
      "$ref": "#/definitions/uint"
    },
    "maxLength": {
      "$ref": "#/definitions/uint"
    },
    "pattern": {
      "type": "string"
    },
    "format": {
      "$ref": "#/definitions/format-"
    },
    "minItems": {
      "$ref": "#/definitions/uint"
    },
    "maxItems": {
      "$ref": "#/definitions/uint"
    },
    "uniqueItems": {
      "type": "boolean"
    },
    "items": {
      "$ref": "#/definitions/jso-items"
    },
    "unit": {
      "type": "string"
    },
    "nullable": {
      "type": "boolean"
    },
    "sdfType": {
```

```
                  "$ref": "#/definitions/sdfType-"
                },
                "contentFormat": {
                  "type": "string"
                }
              },
              "additionalProperties": false
            },
            {
              "type": "object",
    +         "patternProperties": {
    +           "^(?:[a-z][a-z0-9]*:)?[a-z$][A-Za-z$0-9]*$": {
    +           }
    +         },
              "properties": {
                "type": {
    -             "type": "string",
    -             "const": "object"
    -           },
    -           "required": {
    -             "type": "array",
    -             "items": {
    -               "type": "string"
    -             },
    -             "minItems": 1
    -           },
    -           "properties": {
    -             "$ref": "#/definitions/sdfData-sdfChoice-properties-"
    +             "type": "string"
                },
                "enum": {
                  "type": "array",
                  "items": {
                    "type": "string"
                  },
                  "minItems": 1
                },
                "description": {
                  "type": "string"
                },
                "label": {
                  "type": "string"
                },
                "$comment": {
                  "type": "string"
                },
                "sdfRef": {
                  "$ref": "#/definitions/sdf-pointer"
```

```
    },
    "sdfRequired": {
      "$ref": "#/definitions/pointer-list"
    },
    "const": {
      "$ref": "#/definitions/allowed-types"
    },
    "default": {
      "$ref": "#/definitions/allowed-types"
    },
    "minimum": {
      "type": "number"
    },
    "maximum": {
      "type": "number"
    },
    "exclusiveMinimum": {
      "type": "number"
    },
    "exclusiveMaximum": {
      "type": "number"
    },
    "multipleOf": {
      "type": "number"
    },
    "minLength": {
      "$ref": "#/definitions/uint"
    },
    "maxLength": {
      "$ref": "#/definitions/uint"
    },
    "pattern": {
      "type": "string"
    },
    "format": {
      "$ref": "#/definitions/format-"
    },
    "minItems": {
      "$ref": "#/definitions/uint"
    },
    "maxItems": {
      "$ref": "#/definitions/uint"
    },
    "uniqueItems": {
      "type": "boolean"
    },
    "items": {
      "$ref": "#/definitions/jso-items"
```

```
            },
            "unit": {
              "type": "string"
            },
            "nullable": {
              "type": "boolean"
            },
            "sdfType": {
              "$ref": "#/definitions/sdfType-"
            },
            "contentFormat": {
              "type": "string"
            }
          },
          "additionalProperties": false
        }
      ]
    },
    "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": {
              }
+           },
+           {
            }
          ]
        },
        "uint": {
          "type": "integer",
          "minimum": 0
        },
        "jso-items": {
          "anyOf": [
            {
              "type": "object",
+             "patternProperties": {
+               "^(?:[a-z][a-z0-9]*:)?[a-z$][A-Za-z$0-9]*$": {
+               }
+             },
              "properties": {
                "type": {
                  "type": "string",
                  "enum": [
                    "number",
                    "string",
                    "boolean",
                    "integer"
                  ]
                },
                "sdfChoice": {
                  "$ref": "#/definitions/sdfData-sdfChoice-properties-"
                },
                "sdfRef": {
                  "$ref": "#/definitions/sdf-pointer"
                },
                "description": {
                  "type": "string"
                },
                "$comment": {
                  "type": "string"
                },
                "minimum": {
                  "type": "number"
                },
```

```
                "maximum": {
                  "type": "number"
                },
                "format": {
                  "type": "string"
                },
                "minLength": {
                  "$ref": "#/definitions/uint"
                },
                "maxLength": {
                  "$ref": "#/definitions/uint"
                }
              },
              "additionalProperties": false
            },
            {
              "type": "object",
+             "patternProperties": {
+               "^(?:[a-z][a-z0-9]*:)?[a-z$][A-Za-z$0-9]*$": {
+               }
+             },
              "properties": {
                "type": {
                  "type": "string",
                  "const": "object"
                },
                "required": {
                  "type": "array",
                  "items": {
                    "type": "string"
                  },
                  "minItems": 1
                },
                "properties": {
                  "$ref": "#/definitions/sdfData-sdfChoice-properties-"
                },
                "sdfChoice": {
                  "$ref": "#/definitions/sdfData-sdfChoice-properties-"
                },
                "sdfRef": {
                  "$ref": "#/definitions/sdf-pointer"
                },
                "description": {
                  "type": "string"
                },
                "$comment": {
                  "type": "string"
                },
```

```
                "minimum": {
                  "type": "number"
                },
                "maximum": {
                  "type": "number"
                },
                "format": {
                  "type": "string"
                },
                "minLength": {
                  "$ref": "#/definitions/uint"
                },
                "maxLength": {
                  "$ref": "#/definitions/uint"
                }
              },
              "additionalProperties": false
            },
            {
              "type": "object",
+             "patternProperties": {
+               "^(?:[a-z][a-z0-9]*:)?[a-z$][A-Za-z$0-9]*$": {
+               }
+             },
+             "properties": {
+               "type": {
+                 "type": "string"
+               },
+               "sdfChoice": {
+                 "$ref": "#/definitions/sdfData-sdfChoice-properties-"
+               },
+               "sdfRef": {
+                 "$ref": "#/definitions/sdf-pointer"
+               },
+               "description": {
+                 "type": "string"
+               },
+               "$comment": {
+                 "type": "string"
+               },
+               "minimum": {
+                 "type": "number"
+               },
+               "maximum": {
+                 "type": "number"
+               },
+               "format": {
+                 "type": "string"
```

```
+             },
+             "minLength": {
+               "$ref": "#/definitions/uint"
+             },
+             "maxLength": {
+               "$ref": "#/definitions/uint"
+             }
+           },
+           "additionalProperties": false
+         },
+         {
+           "type": "object",
+           "patternProperties": {
+             "^(?:[a-z][a-z0-9]*:)?[a-z$][A-Za-z$0-9]*$": {
+             }
+           },
             "properties": {
               "type": {
                 "type": "string",
                 "enum": [
                   "number",
                   "string",
                   "boolean",
                   "integer"
                 ]
               },
               "enum": {
                 "type": "array",
                 "items": {
                   "type": "string"
                 },
                 "minItems": 1
               },
               "sdfRef": {
                 "$ref": "#/definitions/sdf-pointer"
               },
               "description": {
                 "type": "string"
               },
               "$comment": {
                 "type": "string"
               },
               "minimum": {
                 "type": "number"
               },
               "maximum": {
                 "type": "number"
               },
```

```
                "format": {
                  "type": "string"
                },
                "minLength": {
                  "$ref": "#/definitions/uint"
                },
                "maxLength": {
                  "$ref": "#/definitions/uint"
                }
              },
              "additionalProperties": false
            },
            {
              "type": "object",
+             "patternProperties": {
+               "^(?:[a-z][a-z0-9]*:)?[a-z$][A-Za-z$0-9]*$": {
+               }
+             },
              "properties": {
                "type": {
                  "type": "string",
                  "const": "object"
                },
                "required": {
                  "type": "array",
                  "items": {
                    "type": "string"
                  },
                  "minItems": 1
                },
                "properties": {
                  "$ref": "#/definitions/sdfData-sdfChoice-properties-"
                },
                "enum": {
                  "type": "array",
                  "items": {
                    "type": "string"
                  },
                  "minItems": 1
                },
                "sdfRef": {
                  "$ref": "#/definitions/sdf-pointer"
                },
                "description": {
                  "type": "string"
                },
                "$comment": {
                  "type": "string"
```

```
                },
                "minimum": {
                  "type": "number"
                },
                "maximum": {
                  "type": "number"
                },
                "format": {
                  "type": "string"
                },
                "minLength": {
                  "$ref": "#/definitions/uint"
                },
                "maxLength": {
                  "$ref": "#/definitions/uint"
                }
              },
              "additionalProperties": false
+           },
+           {
+             "type": "object",
+             "patternProperties": {
+               "^(?:[a-z][a-z0-9]*:)?[a-z$][A-Za-z$0-9]*$": {
+               }
+             },
+             "properties": {
+               "type": {
+                 "type": "string"
+               },
+               "enum": {
+                 "type": "array",
+                 "items": {
+                   "type": "string"
+                 },
+                 "minItems": 1
+               },
+               "sdfRef": {
+                 "$ref": "#/definitions/sdf-pointer"
+               },
+               "description": {
+                 "type": "string"
+               },
+               "$comment": {
+                 "type": "string"
+               },
+               "minimum": {
+                 "type": "number"
+               },
```

```
+            "maximum": {
+              "type": "number"
+            },
+            "format": {
+              "type": "string"
+            },
+            "minLength": {
+              "$ref": "#/definitions/uint"
+            },
+            "maxLength": {
+              "$ref": "#/definitions/uint"
+            }
+          },
+          "additionalProperties": false
          }
        ]
      },
+    "sdftype-name": {
+      "type": "string",
+      "pattern": "^[a-z][\\-a-z0-9]*$"
+    },
      "actionqualities": {
        "type": "object",
        "properties": {
          "description": {
            "type": "string"
          },
          "label": {
            "type": "string"
          },
          "$comment": {
            "type": "string"
          },
          "sdfRef": {
            "$ref": "#/definitions/sdf-pointer"
          },
          "sdfRequired": {
            "$ref": "#/definitions/pointer-list"
          },
          "sdfInputData": {
            "$ref": "#/definitions/parameter-list"
          },
          "sdfOutputData": {
            "$ref": "#/definitions/parameter-list"
          },
          "sdfData": {
            "$ref": "#/definitions/sdfData-sdfChoice-properties-"
          }
```

```
        },
+       "patternProperties": {
+         "^(?:[a-z][a-z0-9]*:)?[a-z$][A-Za-z$0-9]*$": {
+         }
+       },
        "additionalProperties": false
      },
      "parameter-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": {
            "$ref": "#/definitions/parameter-list"
          },
          "sdfData": {
            "$ref": "#/definitions/sdfData-sdfChoice-properties-"
          }
        },
+       "patternProperties": {
+         "^(?:[a-z][a-z0-9]*:)?[a-z$][A-Za-z$0-9]*$": {
+         }
+       },
        "additionalProperties": false
      },
      "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"
+         }
+       ]
+     },
+     "sdfType-": {
+       "anyOf": [
+         {
+           "type": "string",
+           "const": "byte-string"
+         },
+         {
+           "type": "string",
+           "const": "unix-time"
+         },
+         {
+           "$ref": "#/definitions/sdftype-name"
+         }
        ]
      },
      "sdfData-sdfChoice-properties-": {
```

```
          "type": "object",
          "additionalProperties": {
            "$ref": "#/definitions/dataqualities"
          }
        },
        "type-": {
          "type": "string",
          "enum": [
            "number",
            "string",
            "boolean",
            "integer",
            "array"
          ]
        },
-       "sdfAction-": {
+       "sdfEvent-": {
          "type": "object",
          "additionalProperties": {
-           "$ref": "#/definitions/actionqualities"
+           "$ref": "#/definitions/eventqualities"
          }
        },
-       "sdfProperty-": {
+       "sdfAction-": {
          "type": "object",
          "additionalProperties": {
-           "$ref": "#/definitions/propertyqualities"
+           "$ref": "#/definitions/actionqualities"
          }
        },
-       "sdfEvent-": {
+       "sdfProperty-": {
          "type": "object",
          "additionalProperties": {
-           "$ref": "#/definitions/eventqualities"
+           "$ref": "#/definitions/propertyqualities"
          }
-       },
-       "sdfType-": {
-         "type": "string",
-         "enum": [
-           "byte-string",
-           "unix-time"
-         ]
        }
      }
    }
```

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 [JSO7] and [JSO7V]. As the vocabulary used by JSO is familiar to many JSON modelers, 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 some interaction data.

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 [JSO4V] 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 can be represented in JSON.

The length (as measured in characters) can be constrained by the additional data qualities "minLength" and "maxLength", which are inclusive bounds.

(More specifically, Unicode text strings as defined in this specification are sequences of Unicode scalar values, the number of which is taken as the length of such a text string. Note that earlier drafts of this specification 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 provided in an extension.)

The data quality "pattern" takes a string value that is interpreted as an [ECMA-262] regular expression in Unicode mode that constrains 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 I-Regexps [RFC9485], 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.

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

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.

For instance, [OCF] is based on Swagger 2.0 which appears to be based on "draft-4" [JSO4][JSO4V] (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.

Appendix D. Composition Examples

This appendix contains two examples illustrating different composition approaches using the sdfThing quality.

D.1. Outlet Strip Example

```
{
  "sdfThing": {
    "outlet-strip": {
      "label": "Outlet strip",
      "description": "Contains a number of Sockets",
      "sdfObject": {
        "socket": {
          "description": "An array of sockets in the outlet strip",
          "minItems": 2,
          "maxItems": 10
        }
      }
    }
  }
}
```

Figure 6

D.2. Refrigerator-Freezer Example

```
{
  "sdfThing": {
    "refrigerator-freezer": {
      "description": "A refrigerator combined with a freezer",
      "sdfProperty": {
        "status": {
          "type": "boolean",
          "description":
"Indicates if the refrigerator-freezer is powered"
        }
      },
      "sdfObject": {
        "refrigerator": {
          "description": "A refrigerator compartment",
          "sdfProperty": {
            "temperature": {
              "sdfRef": "#/sdfProproperty/temperature",
              "maximum": 8
            }
          }
        },
        "freezer": {
          "label": "A freezer compartment",
          "sdfProperty": {
            "temperature": {
              "sdfRef": "#/sdfProproperty/temperature",
              "maximum": -6
            }
          }
        }
      }
    },
    "sdfProperty": {
      "temperature": {
        "description": "The temperature for this compartment",
        "type": "number",
        "unit": "Cel"
      }
    }
  }
```

Figure 7

Acknowledgements

This specification is based on work by the One Data Model group.

Contributors

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

Wouter van der Beek
Cascoda Ltd.
Threefield House
Threefield Lane
Southampton
United Kingdom
Email: w.vanderbeek@cascoda.com

Authors' Addresses

Michael Koster (editor)
KTC
29415 Alderpoint Road
Blocksburg, CA, 95514
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

Ari Keränen
Ericsson
FI-02420 Jorvas
Finland
Email: ari.keranen@ericsson.com