SDF for Instance Information

draft-bormann-asdf-instance-information-05

— four new revisions in three weeks

— (Comparison -01 ⟺ -05)

```
info:
  title: 'A proofshot example for heater #1 on boat #007'
  version: '2025-04-08'
  copyright: Copyright 2025. All rights reserved.
  proofshotId: 026c1f58-7bb9-4927-81cf-1ca0c25a857b
namespace:
  models: https://example.com/models
  boats: https://example.com/boats
defaultNamespace: boats
sdfInstance:
  boat007:
    sdfInstanceOf: models:#/sdfThing/boat
    $context:
      scimObjectId: a2e06d16-df2c-4618-aacd-490985a3f763
    sdfProperty:
      identifier: urn:boat:007:heater:1
      location:
        wgs84:
          latitude: 35.2988233791372
          longitude: 129.25478376484912
          altitude: 0
        postal:
          city: Ulsan
          post-code: '44110'
          country: South Korea
      owner: urn:ietf:spice:glue:pen:30810
```

Instance Information

Information about:

- **Identity** of the thing (on Thing's **digital side**)
  + any Supplementary Identity Info (with **twin**/asset management/…)
- **Purpose** in Life, links, ...
- Information about **physical side**
  (location, manual electrical settings, ...)
  — lives with twin, *possibly* with thing itself

Describing messages ("data in flight")

- SDF describes **information model** of messages that go with interactions (inputdata/outputdata)
- **Ecosystem** may or may not provide rules mapping message IM to message **data model** and/or **serialization**
  - (Also applies to structural aspects of interaction)
- Possibly add ecosystem-specific information about message mapping
  - (Add by derivation? How does a class find a subclass?)

Overview (-01 vs. -05)

- Clarified structure of proofshots
  - Now following the nested structure of the referenced SDF model
  - Created a CDDL model to codify the current structure
- Included a first example for an `eventHistory` in proofshots
- Expanded the section on SDF constructors
  - New example reuses `sdfAction` instead of introducing new keyword
- Included a first example of a proofshot delta
- Cleanup:
  - obsolete examples ➔ appendix
  - Updated examples from digital-twin revision -07 to -08

What do we standardize

- Instance representation format (~ SDF, but instance)
- Modeling the Contents
  - In English 🤔
  - Model-based (SDF!)
    - Leaving models to ecosystems — the OneDM way
    - Kickstart with some pre-standardized basis?
    - Maybe mainly add to the data types and units?

Insights from the Hackathon

- NIPC helped better understand instance vs. model
  - NIPC's `protocolMap` contains ecosystem-specific information that is model-related
  - However: Where to put the instance-specific protocol information (e.g., the MAC address of a BLE device)?
- Also: better understand what SDF constructor should do
  - ➔ initialize the device itself
    (output should not encompass instance description)

New Proofshot Structure

- now aligned with SDF base (qualityName/givenName/...)
  - sdfThing/sdfObject → sdfInstance

Proofshot Example


```
{
  "info": {
    "title": "A proofshot example for heater #1 on boat #007",
    "version": "2025-04-08",
    "copyright": "Copyright 2025. All rights reserved.",
    "proofshotId": "026c1f58-7bb9-4927-81cf-1ca0c25a857b"
  },
  "namespace": {
    "models": "https://example.com/models",
    "boats": "https://example.com/boats"
  },
  "defaultNamespace": "boats",
  "sdfInstance": {
    "boat007": {
      "sdfInstanceOf": "models:#/sdfThing/boat",
      "$context": {
        "scimObjectId": "a2e06d16-df2c-4618-aacd-490985a3f763"
      },
      "sdfProperty": {
        "identifier": "urn:boat:007:heater:1",
        "location": {
          "wgs84": {
            "latitude": 35.2988233791372,
            "longitude": 129.25478376484912,
            "altitude": 0
          }
        }
      }
    }
  }
}
```


Corresponding SDF Model


```
{
  "info": {
    "title": "An example model of a heater on a boat",
    "version": "2025-01-27",
    "copyright": "Copyright 2025. All rights reserved."
  },
  "namespace": {
    "models": "https://example.com/models"
  },
  "defaultNamespace": "models",
  "sdfThing": {
    "boat": {
      "description": "A boat equipped with heating and navigation systems",
      "sdfProperty": {
        "location": {
          "offdevice": true,
          "type": "object",
          "properties": {
            "wgs84": {
              "type": "object",
              "properties": {
                "latitude": {
                  "type": "number"
                },
                "longitude": {
                  "type": "number"
                },
                "altitude": {
                  "type": "number"
                }
              }
            }
          }
        }
      }
    }
  }
}
```

Proofshot Deltas

- How to refer to the previous proofshot?
  - One new approach: `previousProofshot` quality
  - Example overrides the property `status` with `operational`
  - (Might still be quite verbose)

```
{
  "info": {
    "title": "Example SDF delta proofshot",
    "previousProofshot": "026c1f58-7bb9-4927-81cf-1ca0c25a857b",
    "proofshotId": "75532020-8f64-4daf-a241-fcb0b6dc4a42",
    "version": "2025-04-08",
    "copyright": "Copyright 2025. All rights reserved."
  },
  "namespace": {
    "models": "https://example.com/models",
    "boats": "https://example.com/boats"
  },
  "defaultNamespace": "boats",
  "sdfInstance": {
    "boat007": {
      "sdfInstanceOf": "models:#/sdfThing/boat",
      "sdfInstance": {
        "heater": {
          "sdfInstanceOf": "models:#/sdfThing/boat/sdfObject/heater",
          "sdfProperty": {
            "status": "operational"
          }
        }
      }
    }
  }
}
```

Instantiation is an Affordance (!)

- Interaction model may want to describe instantiation
- What parameters are needed (e.g., IP address)
- What interactions are performed during instantiation process
- What information is produced during instantiation process
- (Physical Device probably needs to exist beforehand)

SDF Constructors

- initialize a device
  - Input data: construction message
- Constructors are affordances can therefore be modelled as sdfActions
- After initialization, proofshots can describe the state of the device

old: Create New Keywords

```
{
  "info": {
    "title": "Example document for SDF (Semantic Definition Format) \
                                  with constructors for instantiation",
    "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": {
    "temperatureSensor": {
      "sdfProperty": {
        "temperature": {
          "description": "Temperature value measure by this Thing's \
                                                  temperature sensor.",
          "type": "number",
          "sdfParameters": [
            "minimum",
            {
              "targetQuality": "minimum",
              "parameterName": "minimum",
              "constructorName": "construct"
            },
            "maximum",
            {
              "targetQuality": "unit",
              "parameterName": "#/sdfObject/Switch/sdfConstructors/\
                                          construct/temperatureUnit"
            }
          ]
        }
      },
      "sdfConstructors": {
        "$comment": "TODO: Dicuss whether this should be assumed to \
                                        be the default constructor",
        "construct": {
          "parameters": {
            "minimum": {
              "required": true
            },
            "maximum": {
              "required": false,
              "$comment": "Constructors could allow for further \
             restricting values that can be assigned to affordances",
              "type": "integer"
            },
            "temperatureUnit": {
              "required": false
            }
          }
        }
      }
    }
  }
}
```

New: Using sdfAction as Constructor

```
{
  "info": {
    "title": "Example document for SDF (Semantic Definition Format) with constructors for instantiation",
    "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": {
    "temperatureSensor": {
      "sdfProperty": {
        "temperature": {
          "description": "Temperature value measure by this Thing's temperature sensor.",
          "type": "number",
          "sdfParameters": [
            "minimum",
            {
              "targetQuality": "minimum",
              "parameterName": "minimum",
              "constructorName": "construct"
            },
            "maximum",
            {
              "targetQuality": "unit",
              "parameterName": "#/sdfObject/Switch/sdfConstructors/construct/temperatureUnit"
            }
          ]
        }
      },
      "sdfConstructors": {
        "$comment": "TODO: Dicuss whether this should be assumed to be the default constructor",
        "construct": {
          "parameters": {
            "minimum": {
              "required": true
            },
            "maximum": {
              "required": false,
              "$comment": "Constructors could allow for further restricting values that can be assigned to affordances",
              "type": "integer"
            },
            "temperatureUnit": {
              "required": false
            }
          }
        }
      }
    }
  }
}
```

Constructors: Open Questions

- **Important:** `sdfNonAffordance` vs `offdevice`
  - General questions: How to converge on `instance-information`?
- How to identify constructors as such?
- How to indicate a preference of constructors?
- How to cleanly separate instance from class information in constructors
- What is the form of the instance on which the constructor is called?
  How to talk to it without context?
  - How do protocol bindings on such natal devices work?
- What exactly is the constructor output going to be?
  Just a reference to the instance (description)?

sdf-mapping **Draft**

"SDF model + SDF mapping ➔ fat SDF model" (~ IETF 112)
SDF model + N * SDF mapping ➔ fatter SDF model

- three new revisions since early June
- (Comparison of -05 and -07)

- Made editorial changes
  - Add WoT reference and updated examples
- Outlined details of augmentation mechanism
- Specified algorithm for logging the augmentation steps
- Added examples for both mechanisms/algorithms

Augmentation Mechanism

Algorithm

1. Copy target SDF file
2. For each entry in a Mapping File's `map`:
    1. Resolve JSON pointer (in entry key) against copy to acquire target
    2. Apply Merge Patch algorithm to target with entry value
3. Return augmented copy

Open Questions

- Do we need the `map` object to be an array instead to ensure deterministic augmentation?
- Are JSON pointers good enough to fulfill our requirements? Or are they too limited?
    - Potential use case: Maybe we have information that we want to add to all actions that fulfill certain criteria, for instance?

Logging Algorithm

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

— For each augmentation, record mapping file's URI in an array

— First augmentation: record original SDF model's URI

— If info block or one of the qualities do not exist, they are created

Open Questions

— Identifying the SDF model by its location?
(SDF models are currently not supposed to necessarily correspond with URIs)

Cheating: Describing Instance via Derived Model

- Make an instance by:
  - Referencing base model (sdfRef)
  - Inserting instance information as constant properties into derived model
- Problems:
  - No control
  - Instance information may not be constant ➔ need to update model
  - Not all instance information is on device (or should be obtained there)