ASDF
Internet-Draft
Intended status: Informational
Expires: 23 October 2026

H. Lee, Ed.
J. Hong
ETRI

21 April 2026

Semantic Definition Format (SDF) Modeling for Digital Twin
draft-ietf-asdf-digital-twin-04

Abstract

This memo specifies SDF modeling for digital twins, i.e. a digital twin systems, and their things. An SDF is a format that is used to create and maintain data and interaction, and to represent the various kinds of data that is exchanged for these interactions. The SDF format can be used to model the characteristics, behavior and interactions of things, i.e. physical objects, in digital twins that contain things as components.

Status of This Memo

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

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

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

This Internet-Draft will expire on 23 October 2026.

Copyright Notice

Table of Contents

1. Introduction

A digital twin is defined as a digital representation of an object of interest and may require different capabilities, for example, synchronization and real-time support, according to the specific domain of application. [Y.4600]. Digital twin help organizations improve important functional objectives, including real-time control, off-line analytics, and predictive maintenance, by modeling and simulating objects in the real world. Therefore, it is important for a digital twin to represent as much real-world information about the object as possible when digitally representing the object.

Nowadays, digital twin technologies are applied in various domains including manufacturing, energy, medical, farm, transportation, etc. And a common format is needed to represent the objects in the domains as digital twins. SDF [I-D.ietf-asdf-sdf] can be used for modeling objects as digital twins.

This document specifies the modeling and guidance on how to use SDF to represent objects as digital twins.

2. Terminology

This specification uses the terminology specified in [I-D.ietf-asdf-sdf] in particular "Class Name Keyword", "Object", and "Affordance".

The key words "MUST", "MUST NOT", "REQUIRED", "SHALL", "SHALL NOT", "SHOULD", "SHOULD NOT", "RECOMMENDED", "NOT RECOMMENDED", "MAY", and "OPTIONAL" in this document are to be interpreted as described in BCP 14 [RFC2119] [RFC8174] when, and only when, they appear in all capitals, as shown here.

3. SDF structure for digital twin

This section describes SDF structure to represent a thing or an object as a digital twin. The architecture of a digital twin based on the SDF model is illustrated in Figure 1, following the guidelines of [ISO23247-3].

The physical layer comprises affordance and non-affordance objects. From the real-world objects, only those deemed relevant are selected for representation as digital twins.

The digital twin sublayer is structured into three sublayers: the device communication sublayer, the digital twin sublayer, and the application sublayer.

The device communication sublayer is responsible for monitoring and collecting data from both affordance and non-affordance objects. This sublayer provides the necessary data to synchronize the physical objects with their digital twin counterparts.

The digital twin sublayer ensures synchronization between the affordance and non-affordance objects and their respective digital twins using the data provided by the Device Communication Sublayer.

The Application sublayer presents the synchronized values of the digital twins to users to facilitate informed decision making.

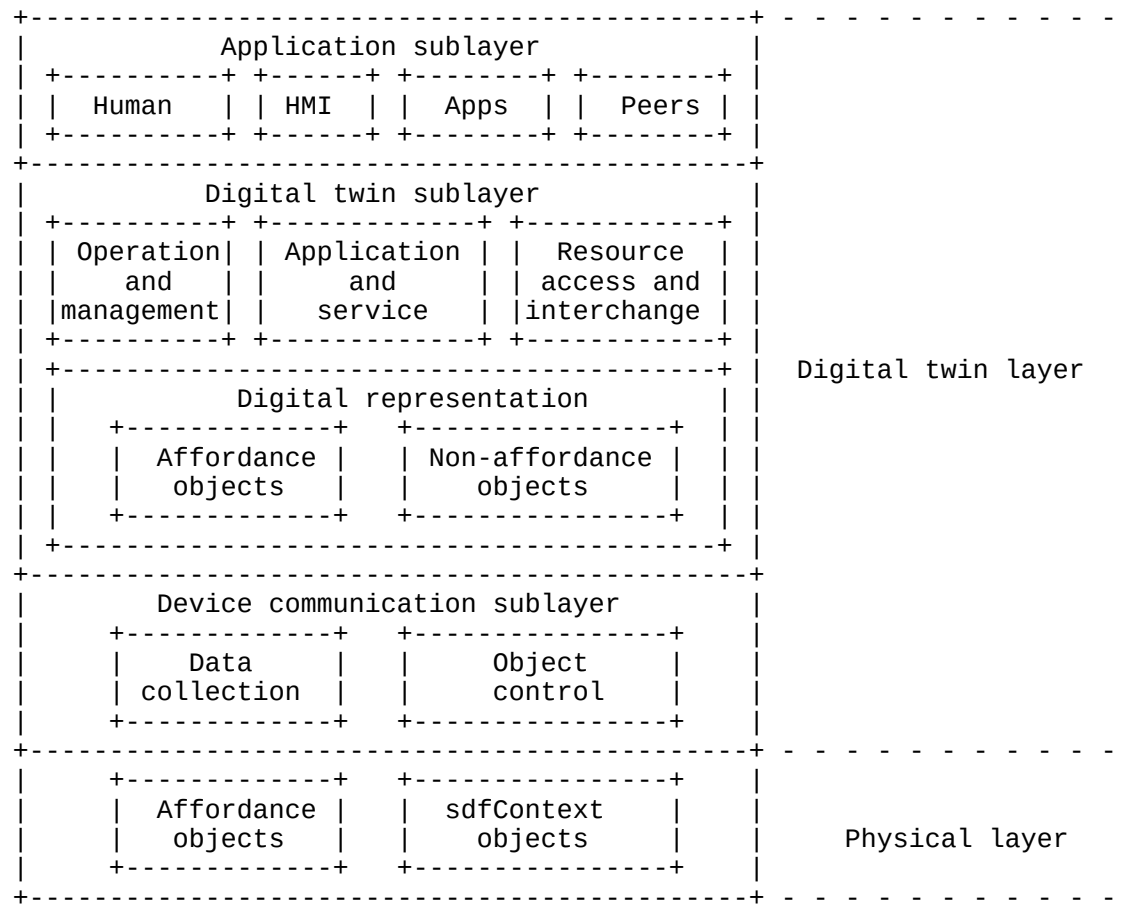

Figure 1: Basic Architecture of digital twin

4. Motivation and design rationale

The document is based on the underlying structure defined in [I-D.ietf-asdf-sdf], which which standardizes the semantic definition format (SDF) for representing IoT affordance. This specification provides a strong basis for representing individual devices and their features (sdfProperty, sdfAction, sdfEvent, etc.), but additional mechanisms are needed to address the unique requirements of digital twin modeling.

Digital twin systems defined in [ISO23247-3] often have to describe virtual representations of various physical objects, including metadata, identity, contextual relationships, historical data, as well as device interfaces.

4.1. Introduction to sdfContext

A new SDF keyword sdfContext described in [I-D.draft-ietf-asdf-sdf-nonaffordance] is introduced to represent non-functional or metadata elements that describe a device or component without implying direct interaction:

* Identifier (e.g., UUID, URN)

* Location (e.g. site, zone, GPS tag)

* Owner (e.g., representative, ,anufacturer)

These field can appear in both sdfObject and sdfThing contexts and follow the same structural pattern as sdfData and is designed for scalability.

4.2. Digital twin modeling using SDF elements

To support hierarchical representations (e.g., a boat composed of heater, GPS, and battery subsystems), this document encourages use of sdfThing to aggregate related sdfObject components, along with metadata.

The example mapping of digital twin attributes to SDF elements is shown in Table 1.

Attribute	Recommended Mapping	Description
Identifier	sdfContext	Globally unique digital twin ID (e.g., URN)
Characteristic	sdfProperty or sdfData	General description or domain properties
Schedule	sdfEvent or sdfData	Time-based actions, availability, or maintenance
Status	sdfAction or sdfProperty	Actual or calculated operating conditions
Location	sdfContext	Physical or logical location information
Report	sdfData	Measurement summaries, analytics, or logs
owner	sdfContext	Organization or entity responsible for the digital twin
Relationship	sdfRelation	Inter-object/inter-twin relationships

Table 1: Digital twin modeling using elements of SDF model

4.3. Relationship modeling

The sdfRelation, defined in [I-D.draft-laari-asdf-relations], is a structure for specifying logical or physical relationships between objects within an SDF model. If conventional sdfThing, sdfObject, and sdfProperty focus on defining the properties of individual digital twins, sdfRelation is a means of expressing interactions and structural links between them. Since these relationships go beyond a single digital twin definition, they must be managed in a separate structure, where sdfRelation is used. The sdfRelation keyword allows describing complex relationships beyond just the parent-child hierarchy. These relationships can include:

* Physical relations (e.g., "inside", "next to")

* Functional relations (e.g., "controls", "is controlled by")

* Semantic relations (e.g., "similar to", "same as")

The sdfRelation definition can include the following fields as defined in [I-D.draft-laari-asdf-relations]:

* relType: Specifies the type of relationship that can an external ontologies (e.g., SAREF[saref4bldg]) can refer to.

* target: Points to the SDF object or an external ontology term that is the target of the relationship.

* description: Provides a detailed textual explanation of the relationship.

* label: A short human-readable label for the relationship.

* property: Additional properties describing the relationship context.

* $comment: Optional properties including implementers notes.

An example of sdfRelation is shown in Figure 2. The sdfProtocolMap in this example is described in [I-D.draft-ietf-asdf-nipc] and [I-D.draft-ietf-asdf-protocol-mapping]

```
   {
      "sdfThing": {
         "Room001": {
             "description": "Contains lightbult and thermostat"
              "sdfObject": {
                  "lightbulb": {
                    "description": "A smart lightbulb",
                    "sdfProperty": {
                        "adjacent-node": { "type": "object", "sdfType": "link"}
                     },
                     "sdfRelation": {
                        "sameRoomAsThermostat": {
                           "relType": "saref:isLocatedIn",
                           "target": "#/sdfObject/thermostat",
                           "description": "This lightbulb is located in the same room as
the thermostat.",
                           "label": "Located together"
                        }
                     }
                },
                "thermostat": {
                  "description": "A thermostat is in the same room as the lightbulb",
                  "sdfProperty": {
                     "adjacent-node": {"type": "object","sdfType": "link"}
                   }
                 },
                 "sdfProtocolMap": {
                   "description": "Protocol between the lightbulb and thermostat",
                   "ble": {
                     "services":
                         [{"serviceID": "361c9c4f-22d7-4a1e-824b-8b61045a566a"}],
                     "cached": false,
                     "cacheIdlePurge": 3600,
                     "unit": "Second",
                     "autoUpdate": true,
                     "bonding": "default"
                  }
               }
            }
         }
      }
   }
```

Figure 2: An example of sdfRelation

5. Protocol considerations for digital twin realization

5.1. Motivation

Digital twins require continuous and reliable communication with physical objects. To support synchronization, monitoring, control, and event notification, appropriate network protocols should be selected and semantically bound to modeled elements in SDF structures. This clause outlines the main protocol types, roles in digital twin operations, and guidelines for representing these bindings using [I-D.draft-ietf-asdf-protocol-mapping].

5.2. Supported protocol types

Digital twin applications can use different types of protocols depending on device performance, data volume, latency sensitivity, and network topology:

Protocol	Role in digital twin	Characteristics
MQTT	Sensor data publishing, event reporting	Lightweight, publish-subscribe, suitable for IoT
CoAP	REST-like access to constrained devices	UDP-based, compact, supports observe/notify
BLE	Local data exchange, control for wearables or embedded devices	Low energy, short range, uses characteristics/ services
HTTP/REST	Enterprise integration, cloud API	Rich semantics, widely supported, heavier overhead
WebSocket	Bi-directional low-latency updates	State synchronization, real-time commands
NIPC	Standardized control of non-IP devices	Useful for industrial, air-gapped, or legacy systems

Table 2: Roles and characteristics for each protocol

5.3. Protocol binding in SDF

To model the way digital twins communicate with their physical objects, sdfProtocolMap can be defined within the sdfProperty, sdfAction or sdfEvent levels. Each protocol entry can specify a communication topic, path, security mechanism, QoS settings, and timing parameters.

```
{
"sdfProtocolMap": {
   "mqtt": {
     "topic": "boat007/heater1/status",
     "qos": 1,
     "updateInterval": 10,
     "unit": "seconds"
   },
   "ble": {
     "characteristicUUID": "00002a6e-0000-1000-8000-00805f9b34fb",
     "serviceUUID": "00001809-0000-1000-8000-00805f9b34fb",
     "read": true,
     "write": true,
     "notify": true
   }
}
```

Figure 3: An example of protocol binding

5.4. QoS and Synchronization Semantics

Quality of service (QoS) settings help define how reliable and frequently data is transferred between digital twins and physical objects, which are particularly important for telemetry (e.g., sdfProperty updates) and command response flows (sdfAction).

QoS level	Meaning	Recommended Usage
0	At most once (best effort)	Periodic sensor data
1	At least once	State updates, events
2	Exactly once	Control command, AI action

Table 3: Meaning and usage of QoS levels

5.5. Security and access considerations

When binding protocols, sdfSecurityMap can be used to include security parameters (e.g., authentication tokens, OSCORE for CoAP, BLE pairing status). Role-based access control for specific protocol endpoints is also recommended.

5.6. Implementation guidelines

The protocol is essential to realizing the digital twin in operation, and it is recommended that the following considerations are taken into account:

* Use protocol mapping appropriate for device classes and network environments

* Provide both push-based (e.g., MQTT) and pool-based (e.g., HTTP) bindings, if applicable

* Reuse standardized subject structures, UUIDs, or endpoint URIs to ensure interoperability

* Avoid duplicate or conflicting bindings, and define protocol preferences via metadata if necessary.

6. Examples of digital twin system

6.1. Overview

Various examples are included to show how SDF-based digital twin models can be applied to real-world scenarios. These examples show how to represent physical objects using SDF elements. In addition, sdfContext and sdfRelation are used to describe additional information such as the context of components, and the relationship between components (sdfRelation). The examples include several domains such as marine systems, healthcare, smart buildings, and energy environments. This consistent modeling approach can support interoperability among applications.

6.2. Marine system

Table 4 describes an example of how a maritime vessel, referred to as Boat007, can be described as a digital twin using the SDF model. In this example, individual physical parts such as heaters and batteries are treated as separate sdfObjects, while the entire vessel is represented as a single sdfThing that groups these components together.

In a vessel modeled with SDF, each component is described using elements such as attributes, actions, and events. These elements are used to represent how the component behaves and what state it is in at a given time. The connections between components are also included in the model. For example, a battery may be linked to a controller, which helps show how different parts are related and interact with each other. This kind of representation makes it easier to follow the condition of devices over time. It also allows operational data to be used more effectively for monitoring, while keeping the model compatible with other systems built in a similar way.

This structure enable developers and systems integrators to:

* Seamlessly capture and communicate the features and state of the device

* Consolidate operational data for real-time monitoring and analysis

* Standardized semantics enables interoperability with other domains.

Attribute	SDF element	Example properties
Boat007	sdfThing	id, name, model, includes heater1 and battery1
Heater1	sdfObject	status (sdfProperty), temperature (sdfProperty), turnOn (sdfAction)
Thermostat1	sdfObject	setPoint, mode (sdfProperty)
Battery1	sdfObject	voltage (sdfProperty), chargeLevel (sdfProperty), battery-to-controller (sdfRelation)
Controller	sdfObject	status (sdfProperty), controlMode (sdfProperty)
Temp-to-Thermostat	sdfRelation	source: heater1.temperature, target: thermostat1.setPoint, relationType: regulatedBy
Battery-to-Controller	sdfRelation	source: batterySensor, target: powerController, relationType: connectedTo
Location	sdfContext	latitude, longitude, dockedAt (e.g., port007)

Table 4: Components and SDF elements of a marine system

In the context of Boat007, shown in Figure 4, such a Digital twin can support various applications, including predictive maintenance, energy optimization, and fleet-level coordination, demonstrating the practicality and scalability of SDF-based Digital twin modeling for mobility and transportation systems.

```
{
  "sdfThing": {
      "boat007": {
      "label": "Boat #007 with a heater",
      "description": "Contains heaters, fans, battery, etc."
      "sdfProperty": {
         "status": {
            "type": "boolean",
            "description": "Indicates if the boat is powered"
          }
      },
      "sdfObject": {
         "heater1": {
          "description": "A heater ",
          "identityManifest": {
             "manufacturer": "HeaterTech Inc.",
              "model": "HEATER-2025-V1",
              "firmwareVersion": "1.4.3",
              "dateOfManufacture": "2025-04-20T09:00:00Z",
              "certifications": [
                { "scheme": "KS", "certId": "KS123", "region": "KR" } ]
            },
            "contextSnapshot": {
              "thingId": "heater:unit5689",
              "timestamp": "2025-05-23T10:20:00Z",
              "installationInfo": {
                 "room": "kitchen",
                  "floor": 1,
                  "mountType": "freestanding",
                  "installationDate": "2025-06-01"
            },
            "usageProfile": {
                "type": "residential",
                "powerCircuit": "230V@60Hz",
                "energyRating": "A++"
            },
            "location": {"lat": 35.1796, "lon": 129.0756 }
          },
          "sdfProperty": {
              "status": {
                "type": "boolean"
                "description":"Whether the heater is powered"
              },
              "temperature": {
                 "type": "number",
                 "unit": "degreeCelsius",
                 "description": "Temperature of the heater"
                }
```

```
                        },
                        "sdfAction": {
                            "turnOn": { "description": "Activate the heater" },
                            "turnOff": { "description": "Deactivate the heater" }
                        },
                        "contextPatch": {
                            "thingId": "heater:unit5689",
                            "timestamp": "2025-06-20T09:00:00Z",
                            "location": {"lat": "35.2988", "lon": "129.2547" },
                            "installationInfo": {"floor": 1, "mountType": "wall" }
                        }
                      },
                      "thermostat": {
                           "maintenanceSchedule": {
                              "timestamp": "2025-05-20T10:00:00Z"
                              "description": "Last maintained date"
                            }
                      },
                      "batterySensor1": {
                        "sdfProperty": {
                            "chargeLevel": {
                              "type": "number",
                              "unit": "percent",
                              "description": "Battery charge level"
                            },
                            "voltage": {
                              "type": "number",
                              "unit": "volt",
                              "description": "Battery voltage"
                            }
                        }
                      },
                      "powerController1": {
                         "sdfAction": {
                            "connect": {"description": "Connect power from the battery"
},
                            "disconnect": {"description": "Disconn power from the batter
y"}
                            }
                        }
                    },
                    "sdfRelation": {
                      "temperature-control": {
                         "source": "#/sdfObject/heater1/sdfProperty/temperature",
                         "target": "#/sdfObject/thermostat1/sdfProperty/setPoint",
                         "relationType": "regulatedBy",
                         "directionality": "unidirectional",
                         "description": "The current temperature of the heater is regula
ted by the thermostat’s setPoint value."
                      },
                      "battery-to-controller": {
```

```
                        "source": "#/sdfObject/batterySensor",
                        "target": "#/sdfObject/powerController",
                        "relationType": "connectedTo",
                        "directionality": "unidirectional"
                    }
                }
            }
        }
    }
```

Figure 4: An example of marine system

6.3. Healthcare system

This example represents a digital twin for a patient health monitor system (patientMonitor001) assigned to a patient. The system reports real-time health properties while referencing contextual patient information with the components and elements shown in Table 5.

Attribute	SDF element	Example properties
Patient monitor	sdfThing	patientMonitor001 as a digital twin
ECG Module	sdfObject	heartRate, rhythmType, signalStrength
Infusion Pump	sdfObject	flowRate, volumeRemaining, alarmStatus
Property	sdfProperty	e.g., temperature, bloodPressureSystolic, oxygenSaturation
Context info	sdfContext	bedNumber, wardLocation, patientID, usageScenario
Identity info	identityManifest	systemType, firmwareVersion, hospitalAssetTag
Relations	sdfRelation	ECG → AlarmSystem (relationType: monitoredBy)

Table 5: Components and SDF elements of a healthcare system

A digital twin example of a patient monitoring system with ECG and infusion pump components is illustated in Figure 5. in the healthcare domain, where a biosensor measuring the heart rate is functionally connected to an alert system that emits a high heart rate warning. This enables real-time patient monitoring in medical environments.

```
{
  "sdfThing": {
    "patientMonitor001": {
    "sdfObject": {
        "ecg": {
            "sdfProperty": {
                "heartRate": { "type": "number", "unit": "bpm" },
                "rhythmType": { "type": "string" }
              }
          },
          "infusionPump": {
              "sdfProperty": {
                "flowRate": { "type": "number", "unit": "ml/h" },
                "volumeRemaining": { "type": "number", "unit": "ml" }
              }
          }
        },
        "sdfContext": {
            "wardLocation": { "const": "ICU-5A" },
            "patientID": { "const": "PT123456" }
        },
        "identityManifest": {
          "manufacturer": "MediTech",
          "model": "IM-500",
          "serialNumber": "MT-IM500-00789"
        },
        "sdfRelation": {
          "heartRate-to-alertSystem": {
              "description": "The heart rate data from the biosensor is moni
tored by the alert system, which triggers a warning event when a high heart rate is detec
ted.",
              "source": "#/sdfObject/biosensor/sdfProperty/heartRate",
              "target": "#/sdfObject/alertSystem/sdfEvent/highHeartRateAlert
",
              "relationType": "monitoredBy",
              "directionality": "unidirectional"
          }
      }
    }
  }
}
```

Figure 5: An example of healthcare

6.4. Smart building system

This example shows a digital twin representing a smart lighting control system within a smart building domain. The system uses both the MQTT protocol and the CoAP protocol to integrate lighting devices and occupancy-based controls. Contextual information such as room number, zone, and usage scenario is included to support location-based control and analysis.

The SDF elements and related components used in this domain are described in Table 6.

Attribute	SDF element	Example properties
Smart room	sdfThing	roomControl001 as a digital twin, including lightController and sensorUnit
Light controller	sdfObject	brightness (sdfProperty), toggle (sdfAction)
Sensor unit	sdfObject	occupancy (sdfProperty), motionDetected (sdfEvent)
Property	sdfProperty	brightness:percent, occupancy:boolean
Action	sdfAction	toggle (on/off), dimTo (level)
Context info	sdfContext	roomNumber: “101”, zone: “eastWing”, usage: “office”
Identity info	identityManifest	vendor: “SmartBuild Inc.”, firmware: “v2.1.0”
Protocol	sdfProtocolMap	MQTT + CoAP for monitoring and control
Relations	sdfRelation	sensor-to-lightController (relationType: triggers)

Table 6: Components and SDF elements of a smart building system

A digital twin representation of the smart building example is shown in Figure 6. In this configuration, occupancy sensors trigger lighting control action through functional relationships,

demonstrating real-time and context-aware behavior. Such modeling can be applicable to energy optimization, comfort control, and responsive automation in smart buildings.

```
{
  "sdfThing": {
    "roomControl001": {
      "sdfContext": {
        "roomNumber": "101",
        "zone": "eastWing",
        "usage": "office"
      },
      "sdfObject": {
        "lightController": {
          "sdfProperty": {
            "brightness": {
              "type": "integer",
              "unit": "percent",
              "description": "Current brightness level of the light"
            }
          },
          "sdfAction": {
            "toggle": {
              "description": "Turns the light on or off"
            },
            "dimTo": {
              "description": "Dims the light to the specified brightness"
            }
          },
          "sdfProtocolMap": {
            "mqtt": {
              "topic": "building/room101/light",
              "qos": 1,
              "updateInterval": 5,
              "unit": "seconds"
            },
            "coap": {
              "method": "POST",
              "href": "/room101/light/toggle"
            }
          }
        },
        "sensorUnit": {
          "sdfProperty": {
            "occupancy": {
              "type": "boolean",
              "description": "Whether the room is currently occupied"
            }
```

```
                    },
                    "sdfEvent": {
                      "motionDetected": {
                        "description": "Triggered when motion is detected"
                      }
                    }
                  }
                },
                "sdfRelation": {
                  "sensorToLight": {
                    "source": "#/sdfThing/roomControl001/sdfObject/sensorUnit",
                    "target": "#/sdfThing/roomControl001/sdfObject/lightController",
                    "relationType": "triggers",
                    "directionality": "unidirectional"
                  }
                }
              }
            }
          }
```

Figure 6: An example of smart building lighting system

7. Requirements for implenmenting digital twin

A digital twin is a partial representation of sdfThing or sdfObject that contains attributes such as sdfProperty, sdfAction and sdfEvent[ISO23247-1]. By representing sdfThing as a digital twin, crucial events that require appropriate action can be quickly detected and controlled. The requirements defined in [ISO23247-1] are applied to represent sdfThings and sdfObjects as digital twins.

* Identification: sdfThings and sdfObjects should contain data that uniquely identify them as digital twins.

* Data acquisition: data related to sdfThing and sdfObject, such as sdfProperty, sdfEvent, and sdfAction, should be collected from IP and non-IP systems.

* Data analysis: collected data needs to be analyzed to understand the state of sdfThing and sdfObject.

* Accuracy: The sdfThings and sdfObjects should be represented as digital twins with appropriate levels of detail and accuracy, depending on the application.

* Synchronization: sdfThings and sdfObjects should be synchronized with the digital twin at intervals appropriate to the requirements of each application. Newly added or deleted sdfThings and sdfObjects should be recognized and reflected in the digital twin.

8. Procedure for digital twin implementation

8.1. Overview

It is essential to define a standardized implementation procedure to ensure interoperability, scalability, and effective lifecycle management across digital twin systems. This section outlines a step-by-step approach aligned with the Semantic Definition Format (SDF) model and its architecture, enabling consistent modeling, integration, and operation of digital twins in IoT environments. A general principles for representing an sdfThing as a digital twin within a specific domain is outlined as follows:

* defining a purpose for expressing the observable object as a digital twin in the domain

* collecting and mapping data based on the roles of the observable object in the domain

* configuring the observable object into the digital twin based on the data for the purposes

* interworking among digital twins reflecting various roles of the observable object

* synchronizing the observable object and the digital twin

8.2. Procedure

The procedure of digitally twinning the space and the objects contained in it is described.

* Identifying and scoping physical objects: The first step is to clearly identify the physical objects that will be represented as digital twins. This step includes assigning a globally unique identifier, such as a URN or UUID, and determining the extent of modeling. It also involves deciding whether the unique identifier will cover the entire system or focus on a specific subsystem or component. Although all objects in space can be represented by digital twins, it is cost-effective to select objects for implementation purposes and configure them as digital twins.

* Defining a digital twin: A detailed digital twin should be defined using SDF structures, including sdfThing and sdfObject. This step requires specifying affordances such as sdfProperty, sdfAction, and sdfEvent, as well as non-affordance metadata like location, owner, and other descriptive elements through sdfContext.

* Metadata and contextualization: This step adds metadata that enriches the context of the digital twin, such as geographic location, ownership details, manufacturing information, and feature summaries. It can also support advanced analytics and management, including contextual attributes such as production schedules or maintenance periods.

* Binding interfaces and communications: Digital twins are bound to real-world communication interfaces and protocols such as MQTT, CoAP, and HTTP. This allows affordance of SDF models to interact with real-world data sources, APIs, and physical objects in a smooth and reliable manner.

* Verification and compliance: Once an object is defined and bound as a digital twin, it should be validated against syntax and semantic rules using tools such as JSON schema validators or CDDL definitions. Compliance with specific SDF profiles or domain-specific standards must also be verified to ensure interoperability.

* Deployment and registration: After verification, the digital twins are deployed in a digital twin registry, edge system, or cloud infrastructure. This step involves registering the model with the discovery service for integration and use by other systems or stakeholders.

* Runtime monitoring and updating: During operations, digital twins need to continuously monitor real data and update their status accordingly. Properties updates, event processing, and partial updates using contextPatch messages should be supported for efficient and lightweight synchronization.

* Lifecycle and governance management: The life cycle of the digital twin is managed through version tracking, audit logs, and compliance documents. This step ensures safe and transparent governance and enables proper disposal and deregistration when objects are no longer available.

9. Security Considerations

Only authorized users should have the authority to manage digital twins, sdfThings and sdfObjects. Also, Secure communication and metadata integrity are essential when implementing digital twins. All context messages, including contextPatch and identityManifest, must have mechanisms such as authentication and authorization applied.

10. IANA Considerations

This document has no IANA actions.

11. References

11.1. Normative References

[I-D.draft-ietf-asdf-nipc]
Brinckman, B., Mohan, R., and B. Sanford, "An Application Layer Interface for Non-IP device control (NIPC)", Work in Progress, Internet-Draft, I-D.draft-ietf-asdf-nipc-13, 20 September 2025, <https://datatracker.ietf.org/doc/html/I-D.draft-ietf-asdf-nipc-13>.

[I-D.draft-ietf-asdf-protocol-mapping]
Mohan, R., Brinckman, B., and L. Corneo, "SDF Protocol Mapping", Work in Progress, Internet-Draft, I-D.draft-ietf-asdf-protocol-mapping-06, 2 March 2026, <https://datatracker.ietf.org/doc/html/I-D.draft-ietf-asdf-protocol-mapping-06>.

[I-D.draft-ietf-asdf-sdf-nonaffordance]
Hong, J. and H. Lee, "Semantic Definition Format (SDF) Extension for Non-Affordance Information", Work in Progress, Internet-Draft, I-D.draft-ietf-asdf-sdf-nonaffordance-01, 21 September 2025, <https://datatracker.ietf.org/doc/html/I-D.draft-ietf-asdf-sdf-nonaffordance-01>.

[I-D.draft-laari-asdf-relations]
Laari, P., "Extended relation information for Semantic Definition Format (SDF)", Work in Progress, Internet-Draft, I-D.draft-laari-asdf-relations-04, 28 January 2025, <https://datatracker.ietf.org/doc/html/I-D.draft-laari-asdf-relations-04>.

[I-D.ietf-asdf-sdf]
Koster, M., Bormann, C., and A. Kernen, "Semantic Definition Format (SDF) for Data and Interactions of Things", Work in Progress, Internet-Draft, draft-ietf-asdf-sdf-25, 13 October 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-asdf-sdf-25>.

[ISO23247-1]
"Automation systems and integration Digital twin framework for manufacturing - Part 1: Overview and general principles, ISO 23247-1.", October 2021, <https://www.iso.org/standard/75066.html>.

[ISO23247-3]
"Automation systems and integration Digital twin framework for manufacturing - Part 3: Digital representation of manufacturing elements, ISO 23247-3.", October 2021, <https://www.iso.org/standard/78744.html>.

[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>.

[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>.

[Y.4600] Union, I. T., ""Recommendation ITU-T Y.4600 (2022), Requirements and capabilities of a digital twin system for smart cities.", August 2022.

11.2. Informative References

[saref4bldg]
Poveda-Villaln, M. and R. Garcia-Castro, "SAREF extension for building", 5 June 2020, <https://saref.etsi.org/saref4bldg>.

Acknowledgements

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

Contributors

Joo-Sang Youn
DONG-EUI University
176 Eomgwangno Busan_jin_gu
Busan
47340
South Korea
Phone: +82 51 890 1993
Email: joosang.youn@gmail.com

Yong-Geun Hong
Daejeon University
62 Daehak-ro, Dong-gu
Daejeon
34520
South Korea
Phone: +82 42 280 4841
Email: yonggeun.hong@gmail.com

Authors' Addresses

Hyunjeong Lee (editor)
Electronics and Telecommunications Research Institute
218 Gajeong-ro, Yuseong-gu
Daejeon
34129
South Korea
Phone: +82 42 860 1213
Email: hjlee294@etri.re.kr

Jungha Hong
Electronics and Telecommunications Research Institute
218 Gajeong-ro, Yuseong-gu
Daejeon
34129
South Korea
Phone: +82 42 860 0926
Email: jhong@etri.re.kr