

ASDF  
Internet-Draft  
Intended status: Informational  
Expires: 24 July 2026

H. Lee, Ed.  
J. Hong  
ETRI  
20 January 2026

Semantic Definition Format (SDF) modeling for Digital Twin  
draft-ietf-asdf-digital-twin-03

## Abstract

This memo specifies SDF modeling for a digital twin, i.e. a digital twin system, and its 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 a digital twin 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 24 July 2026.

## Copyright Notice

Copyright (c) 2026 IETF Trust and the persons identified as the document authors. All rights reserved.

This document is subject to BCP 78 and the IETF Trust's Legal Provisions Relating to IETF Documents (<https://trustee.ietf.org/license-info>) in effect on the date of publication of this document. Please review these documents carefully, as they describe your rights and restrictions with respect to this document. Code Components extracted from this document must include Revised BSD License text as described in Section 4.e of the Trust Legal Provisions and are provided without warranty as described in the Revised BSD License.

## Table of Contents

1. Introduction . . . . .	2
2. Terminology . . . . .	3
3. SDF structure for digital twin . . . . .	3
4. Motivation and design rationale . . . . .	4
4.1. Introduction of sdfContext . . . . .	5
4.2. Digital Twin Modeling using SDF elements . . . . .	5
4.3. Relationship modeling . . . . .	6
5. Examples of digital twin system . . . . .	8
5.1. Overview . . . . .	9
5.2. Marine system . . . . .	9
5.3. Healthcare system . . . . .	13
5.4. Smart building system . . . . .	15
6. Requirements for implementing digital twin . . . . .	17
7. Procedure for digital twin implementation . . . . .	18
7.1. Overview . . . . .	18
7.2. Procedure . . . . .	18
8. Security Considerations . . . . .	20
9. IANA Considerations . . . . .	20
10. References . . . . .	20
10.1. Normative References . . . . .	20
10.2. Informative References . . . . .	21
Acknowledgements . . . . .	21
Contributors . . . . .	21
Authors' Addresses . . . . .	22

## 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.

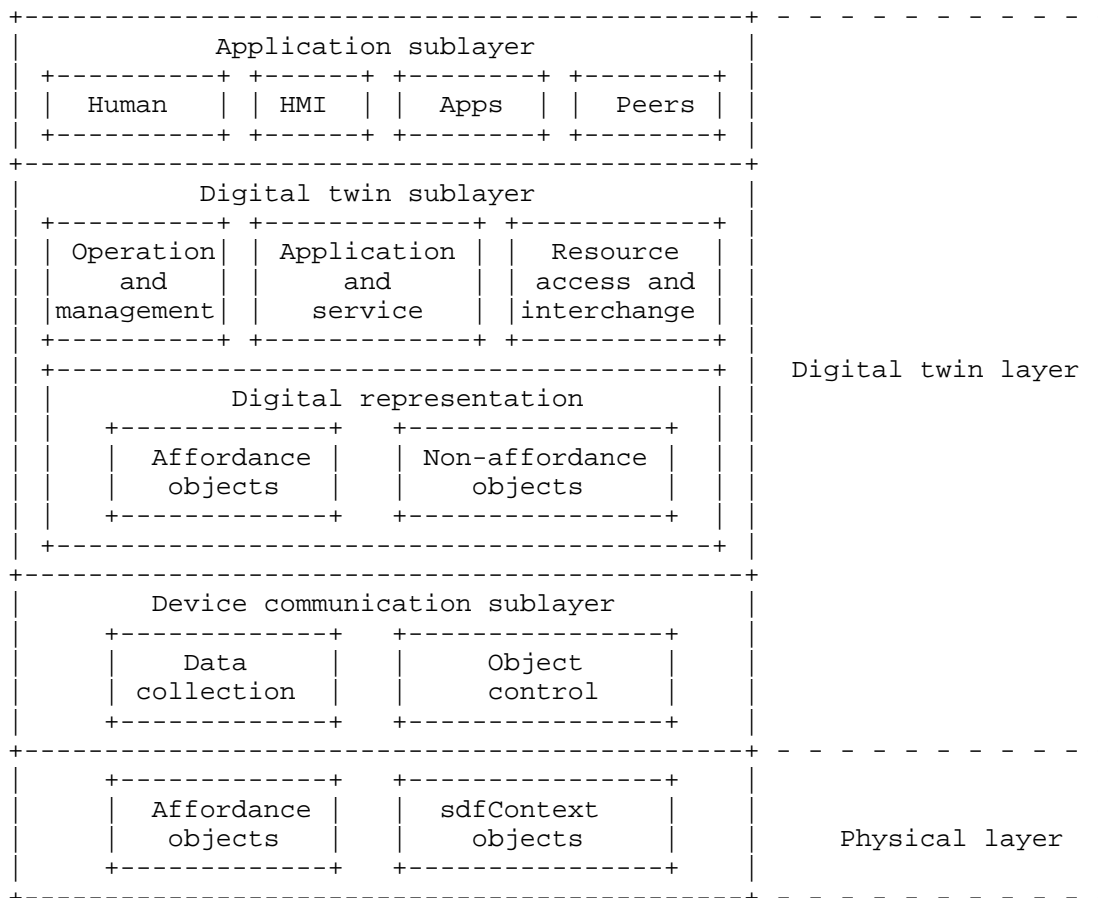


Figure 1: Basic Architecture of digital twin

#### 4. Motivation and design rationale

The document is based on the underlying structure defined in [I-D.ietf-asdf-sdf], which 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 assets, including metadata, identity, contextual relationships, historical data, as well as device interfaces.

#### 4.1. Introduction of 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 an 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 defined 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. Examples of digital twin system



### 5.1. Overview

Domain-specific examples are provided to illustrate how SDF-based Digital Twin representations can be modeled across various application areas. Each example is structured using the `sdfThing` construct to represent the physical entity, with associated `sdfObject`, `sdfProperty`, `sdfAction`, `sdfEvent`, and optional `sdfContext` or `sdfRelation` entries. These examples cover multiple domains such as maritime, healthcare, smart building and energy systems, enabling standardized modeling and interoperability across diverse use cases.

### 5.2. Marine system

Table 2 illustrates how various components of a maritime vessel—specifically Boat007—can be represented as a structured Digital Twin using the Semantic Definition Format (SDF) model. Each physical component, such as a heater or battery, is abstracted as an `sdfObject`, while the overall vessel is modeled as an `sdfThing`.

For each component, key SDF elements such as `sdfProperty`, `sdfAction`, or `sdfEvent` are defined to describe the operational and contextual aspects of the system. Relationships such as `sdfRelation` are used to express functional connections (e.g., the battery is `connectedTo` the controller), enabling richer modeling of interactions between components.

This structure allows developers and systems integrators to:

- \* Seamlessly capture and communicate device capabilities and states.
- \* Integrate operational data for real-time monitoring and analysis.
- \* Enable interoperability with other domains through standardized semantics.

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 2: Components and SDF elements of a marine system

In the context of Boat007, shown in Figure 3, 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 3: An example of marine system

### 5.3. 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 3.

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 3: Components and SDF elements of a healthcare system

A digital twin example of a patient monitoring system with ECG and infusion pump components is illustrated in Figure 4. 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 monitored by the alert system, which triggers a warning event when a high heart rate is detected.",
          "source": "#/sdfObject/biosensor/sdfProperty/heartRate",
          "target": "#/sdfObject/alertSystem/sdfEvent/highHeartRateAlert",
          "relationType": "monitoredBy",
          "directionality": "unidirectional"
        }
      }
    }
  }
}

```

Figure 4: An example of healthcare

#### 5.4. 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 4.

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 4: Components and SDF elements of a smart building system

A digital twin representation of the smart building example is shown in Figure 5. 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 5: An example of smart building lighting system

## 6. Requirements for implementing 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.

## 7. Procedure for digital twin implementation

### 7.1. 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

### 7.2. Procedure

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

- \* Identifying and scoping physical assets: The first step is to clearly identify the physical assets 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 assets in space can be represented by digital twins, it is cost-effective to select assets 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 assets in a smooth and reliable manner.
- \* **Verification and compliance:** Once an asset 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 assets are no longer available.

## 8. 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.

## 9. IANA Considerations

This document has no IANA actions.

## 10. References

### 10.1. 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, "Protocol Mapping for SDF", Work in Progress, Internet-Draft, I-D.draft-ietf-asdf-protocol-mapping-01, 16 October 2025, <<https://datatracker.ietf.org/doc/html/I-D.draft-ietf-asdf-protocol-mapping-01>>.

[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. Kernén, "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.

## 10.2. Informative References

[saref4bldg]

Poveda-Villalón, M. and R. García-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