ASDF
Internet-Draft
Intended status: Standards Track
Expires: 24 July 2026

J. Hong, Ed.
H. Lee
ETRI

20 January 2026

Semantic Definition Format (SDF) Extension for Non-Affordance Information
draft-ietf-asdf-sdf-nonaffordance-03

Abstract

This document describes an extension to the Semantic Definition Format (SDF) for representing non-affordance information of Things, such as physical, contextual, and descriptive metadata. This extension introduces a new class keyword, sdfContext, that enables comprehensive modeling of Things and improves semantic clarity.

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.

Table of Contents

1. Introduction

The Semantic Definition Format (SDF) standardizes the representation of affordances of Things, namely Properties, Actions, and Events [I-D.ietf-asdf-sdf]. However, SDF does not currently define a way to represent non-affordance information, such as location, contextual metadata, identifiers, and other descriptive elements that are not directly related to device interactions. The absence of such constructs limits the ability to model devices and systems in use cases that require both interactive behavior and descriptive metadata.

This document specifies an extension to SDF to represent non-affordance information in a consistent and interoperable way. The extension is introduced as a new class keyword, defined alongside the existing affordance classes, and provides a mechanism for expressing descriptive Information that complements interactive definitions.

2. Terminology and Conventions

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.

* Non-Affordance: information about a Thing that is not directly related to its interactive capabilities. Non-affordance information does not define how external entities can act upon the Thing, but instead provides descriptive metadata useful for interpretation, management, or integration. Examples include location, manufacturer details, calibration parameters, or deployment context.

3. Motivation and Use Cases

The integration of non-affordance information into the Semantic Definition Format (SDF) addresses several critical needs in the modeling of Internet of Things (IoT) devices. The key motivations and corresponding use cases in the following subsections illustrate the importance of this extension.

3.1. Motivation

In the current SDF framework, the primary focus is on defining affordances - interactive elements such as Properties, Actions, and Events. While this approach effectively captures the dynamic capabilities of a Thing, it overlooks essential non-interactive attributes that are vital for a comprehensive device representation. These non-affordance attributes encompass contextual information and descriptive metadata, including dimensions, weight, location, manufacturer details, and operational constraints. The absence of a standardized representation for such static information can lead to fragmented device models, hindering interoperability and seamless integration across diverse IoT ecosystems. Although it is technically possible to model such information using 'sdfProperty', this approach introduces several forms of semantic confusion:

1. Users may misinterpret the field as observable or interactive: When interactive properties (e.g. sensor readings or actuators) and fixed attributes (e.g. a device' s physical dimensions or serial number) are all represented as properties, it becomes unclear which elements are meant to change or be acted upon and which are immutable context. This ambiguity forces developers and tools to infer intent manually, making models harder to interpret and maintain. Over time, such models require extra documentation and care to ensure that static fields are not mistakenly treated as dynamic, adding to the maintenance burden.

2. Developers may implement unnecessary runtime I/O interfaces: Many IoT frameworks and tools automatically create API handlers (e.g., REST endpoints or CoAP resources) for each defined Property. If static metadata like a device's model name or install location is modeled as a property, a tool might erroneously generate read/

write accessors for it. This is problematic because such metadata is meant to be read-only context, not an interactive affordance. The result is superfluous or misleading interface endpoints that do not reflect the device's real capabilities, potentially causing confusion or security issues. In short, using sdfProperty for static fields violates the expectation that those fields remain non-interactive, since the default affordance treatment would imply they can be polled or even written to.

3. Tools and UIs may treat static metadata as operational data: SDF is meant to clearly convey a Thing's interactive capabilities versus its contextual attributes. When both are blended under the same construct, developers and automated tools may misinterpret the purpose of a given field. For example, a field representing location or manufacturer might be misconstrued as an operational parameter rather than informative metadata. This blurring of semantics makes it harder to build consistent tooling and to map SDF models to platform implementations, since one cannot reliably distinguish which elements require interactive handling. In essence, the lack of separation between affordances and non-affordances dilutes the semantic clarity of SDF models , undermining the SDF goal of an unambiguous, self-descriptive device model.

To address these concerns, this document introduces 'sdfContext' as a dedicated top-level keyword to define static, descriptive, and non-interactive metadata. This construct enables a clear semantic distinction from 'sdfProperty', making the data model more expressive, machine-readable, and robust to implementation assumptions.

While the primary focus of this document is the introduction of a static model extension via 'sdfContext', practical use of such metadata in real-world deployments often requires runtime mechanisms for metadata exchange. Use cases such as device onboarding, dynamic environment configuration, and regulatory audits benefit from the ability to transmit static context data as part of operational protocols. To that end, this draft also introduces optional runtime messages -'contextSnapshot', 'identityManifest', and 'contextPatch'- that can convey non-affordance attributes at appropriate times.

These messages are not the core of the SDF extension but are essential for practical interoperability, especially in systems where device metadata needs to be programmatically discovered, validated, or synchronized. Their inclusion supports real-world use cases that rely on the seamless integration of descriptive metadata into operational contexts. Thus, the proposal reflects both a modeling advancement for SDF and a runtime integration pattern to enable

widespread adoption. This design accommodates ecosystem-specific metadata such as regulatory certifications, deployment regions, or vendor-specific constraints by allowing flexible attributes to be included and mapped according to the needs of each ecosystem.

3.2. Use Cases

This section illustrates how non-affordance information modeled via sdfContext, contextSnapshot, and identityManifest supports interoperability, lifecycle traceability, and accurate interpretation of data across domains.

3.2.1. Asset Management and Tracking

* *Scenario:* In logistics and warehouse systems, physical containers and pallets are often equipped with IoT sensors to monitor conditions such as temperature, vibration, and location. In addition, each container possesses immutable physical attributes—such as size, weight, manufacturer, and maximum load capacity—that directly influence deployment, transportation, and regulatory compliance.

* *Concrete Example:* For instance, a “20ft shipping container manufactured in August 2025” would be registered with length, width, height, manufacturer name, and maximum permitted load, which never change throughout its lifecycle. These non-affordance data fields are recorded using sdfContext and are referenced whenever containers are assigned to shipping routes, loaded with cargo, or inspected by authorities.

* *Data Flow:* During onboarding, these static attributes are retrieved from the device’s digital representation or management registry and stored in the asset management system, enabling automated planning and compliance checks. Because they are immutable, they prevent errors in cargo assignment and support maintenance audits. If placement or ownership changes are required, an update can be provided using a contextPatch message without altering the interactive model.

* *Value of Separation:* By distinguishing non-affordance metadata from dynamic sensor properties, the SDF model ensures that static context is protected against accidental modification, enhancing system reliability and regulatory integrity.

3.2.2. Environmental Context Awareness

* *Scenario:* Environmental sensors in smart buildings, such as temperature sensors or CO2 monitors, are installed across diverse locations—such as underground ventilation ducts or third-floor windows—which dramatically affect how sensor values should be interpreted.

* *Concrete Example:* A motion sensor installed on the ceiling of a conference room on floor B1 would declare floor: B1, mountType: ceiling, and indoorOutdoor: indoor in its sdfContext. This static context enables analytics tools to filter and adjust real-time readings, for instance distinguishing between typical fluctuations due to conference room activity versus anomalies that indicate faults.

* *Data Flow:* At installation, the deployment technician registers this context information; during operation, a contextSnapshot message provides complete context to facility management systems and enables energy optimization or fault detection algorithms to adjust thresholds and logic based on known placement. This aligns with the instance-level "proofshot" structure, limited to static contextual data.

* *Value of Separation:* Separating static environmental context from dynamic sensor data streamlines calibration, reduces errors in data interpretation, and enables automated system adaptation to environmental changes without manual oversight.

3.2.3. Regulatory Compliance and Certification

* *Scenario:* In regulated environments such as healthcare, aviation, or industrial automation, each device must report immutable identification (manufacturer, model, production date) and traceable certification information to governing authorities and compliance systems.

* *Concrete Example:* Upon commissioning a hospital-grade infusion pump, its sdfContext includes manufacturer, model, firmwareVersion, and a set of certification records (for example, FDA certification ID, CE mark, and regulatory region). These remain unchanged during operational use and appear in device management dashboards, audit logs, and automated compliance reporting workflows.

* *Data Flow:* identityManifest messages are sent during device onboarding and updates—often as the output of a construction or commissioning process—verifying compliance status. Regulatory systems query these static records to check policy enforcement, recall campaigns, and security patch eligibility.

* *Value of Separation:* Non-affordance identity and certification data, managed as immutable context, guarantee regulatory and operational traceability and prevent unauthorized device modifications, forming the backbone of trustworthy automated compliance.

By integrating non-affordance information into SDF as described in these use cases, a more holistic device model can be achieved—one that enhances interoperability, operational efficiency, and compliance across diverse IoT applications.

4. SDF Extension for Non-Affordance Information

In the SDF, the primary focus has been on defining affordances - interactive elements such as Properties, Actions, and Events - that specify how external entities can interact with a Thing. However, SDF does not provide a construct to represent non-affordance information, which covers attributes not directly related to interaction but needed to describe a Thing’s context and characteristics.

To address this need, this document introduces a new class keyword, "sdfContext". The "sdfContext" keyword is a peer to "sdfProperty", "sdfAction", and "sdfEvent", and is used to represent non-affordance information when authoring the SDF model (i.e., at design time). Definitions under "sdfContext" are read-only in any generated interface. Examples include location, manufacturer details, calibration parameters, installation attributes, and static identifiers. It is important to note that ecosystem-specific details can alternatively be layered onto an SDF model via an external mapping mechanism; SDF mapping files [Mapping] provide a way to augment a base model with additional qualities or metadata. The sdfContext extension defined in this document complements that approach by providing a first-class construct within the model for static context, ensuring that such metadata is uniformly available and semantically distinguished in the core specification of a Thing.

The qualities of an "sdfContext" definition include the common qualities defined in Section 4.6 ("Common Qualities") of [I-D.ietf-asdf-sdf]. Additional qualities are shown in Table 1. None of these qualities are required or have default values that are assumed if the quality is absent.

Quality	Type	Description
(common)	-	See Section 4.6 ("Common Qualities") of [I-D.ietf-asdf-sdf]
type	string/ object	Data type of the context item (e.g., string, number, object)
description	string	Human-readable explanation of the context item
required	array	List of mandatory context entries

Table 1: SDF-defined Qualities of sdfContext

Practical scenarios where "sdfContext" may be applied include:

* Asset Management: associating static metadata such as manufacturing date, supplier, or warranty information with a device in a fleet.

* Commissioning Tools: storing parameters injected during deployment (e.g., room assignment, installation coordinates) that are not otherwise observable through interactive affordances.

* Calibration Metadata: describing accuracy classes or calibration factors used by the device when reporting measurements. These values are typically read-only and not exposed via a protocol interface.

* Identity and Traceability: capturing serial numbers, SKU identifiers, or web resource references that uniquely characterize a device but remain constant throughout its lifecycle.

The following example defines an sdfObject that contains both an affordance and several non-affordance attributes. The affordance is represented as a temperature property, which is a numeric value in degrees Celsius. The non-affordance attributes are grouped under sdfContext. These include a serialNumber, which is a factory-assigned identifier; an installationLocation, which indicates where the device is deployed within a building; and a calibrationOffset, which specifies a correction value applied to raw sensor measurements. Together, these definitions illustrate how sdfContext can capture descriptive metadata that is not directly interactive but still essential for interpretation and integration.

```
{
  "sdfObject": {
    "deviceWithContext": {
      "sdfProperty": {
        "temperature": {
          "type": "number",
          "unit": "°C"
        }
      },
      "sdfContext": {
        "serialNumber": {
          "type": "string",
          "description": "Unique factory-assigned identifier"
        },
        "installationLocation": {
          "type": "string",
          "description": "Location within the building where the device is in
stalled"
        },
        "calibrationOffset": {
          "type": "number",
          "unit": "°C",
          "description": "Offset applied to raw measurements for calibration"
        }
      }
    }
  }
}
```

Figure 1: SDF Object containing a Property and a Context entry

Section 4 specifies how non-affordance information is added to an SDF model and how that information can be exchanged at runtime. We deliberately split the description into two complementary subsections:

* 4.1 Static model Definition shows how contextual metadata is embedded directly in the SDF document under the new sdfContext class. These definitions are authored once, validated like any other SDF schema fragment, and travel with the model wherever it is stored or published.

* 4.2 Run-Time Context Messaging introduces three JSON envelopes (contextSnapshot, identityManifest, contextPatch) that let a deployed device or its digital twin report changes to that metadata over time. Keeping these messages outside the affordance model preserves the core principle that non-affordance data is descriptive, not interactive, while still allowing systems to keep the context up-to-date.

To align with the use cases described in Section 3.2, this section introduces detailed examples of non-affordance information modeling using the ’sdfContext’ construct. Each subsection demonstrates how contextual metadata can be represented in an SDF model corresponding to a specific use case.

4.1. Static Model Definition (sdfContext)

The sdfContext class is intended for non-affordance information that is static or metadata-oriented. These are values that are expected to be known at design time, provisioned at deployment time, or extracted from static documentation or identity stores. Examples include manufacturer name, model number, location of deployment, or contact email for support.

4.1.1. Comparison Between sdfContext and sdfProperty

Although both sdfProperty and sdfContext may describe characteristics of a Thing, they differ fundamentally in purpose, mutability, and system interaction behavior. Table 2 summarizes these differences.

Aspect	sdfProperty	sdfContext
Purpose	Represents observable or modifiable affordances	Represents static descriptive information
Mutability	Typically mutable or changeable over time	Immutable after deployment
Protocol Mapping	Generates REST/CoAP/ other interface endpoints	Does not generate interactive endpoints
UI Representation	Interactive elements (sliders, inputs)	Read-only fields (labels, metadata panels)
Tooling Behavior	Polled, observed, or updated via APIs	Displayed statically; excluded from interaction logic
Migration Criteria	-	Use if: (1) immutable, (2) non-pollable, (3) descriptive

Table 2: Comparison of sdfProperty and sdfContext

The distinction between sdfProperty and sdfContext is important for toolchain behavior, interface generation, and user interface clarity. While both may contain descriptive data, sdfProperty is intended for values that are dynamic, observable, or settable at runtime. In contrast, sdfContext is reserved for metadata that is immutable after deployment and does not participate in interaction protocols.

For example, a sensor’s temperature reading should be modeled as a sdfProperty since it changes over time and clients may want to poll or subscribe to it. However, a serial number or installation date should be modeled as sdfContext, as these are assigned once and do not change or require polling.

From an implementation perspective, tools MUST NOT generate GET/PUT endpoints for sdfContext, and UI frameworks SHOULD render these values as static labels or metadata panels. If an existing model contains sdfProperty entries that are read-only and represent metadata, they MAY be migrated to sdfContext for semantic clarity.

4.1.2. Practical Use Cases for sdfContext

This subsubsection presents SDF examples that statically describe non-affordance information associated with different types of devices and use cases. Each example corresponds to a use case described in Section 3.2.

4.1.2.1. Example: Asset Management and Tracking

The following example corresponds to the use case in Section 3.2.1. It models static metadata for a shipping container, including physical dimensions and capacity.

```
{
  "sdfObject": {
    "assetContainer": {
      "description": "Shipping container with embedded sensors",
      "sdfContext": {
        "physicalSpecs": {
          "description": "Physical dimensions and capacity",
          "type": "object",
          "properties": {
            "length": { "type": "number", "unit": "m" },
            "width": { "type": "number", "unit": "m" },
            "height": { "type": "number", "unit": "m" },
            "maxLoadKg": { "type": "number", "unit": "kg" }
          },
          "required": ["length", "width", "height", "maxLoadKg"]
        },
        "location": {
          "type": "object",
          "properties": {
            "lat": { "type": "number" },
            "lon": { "type": "number" }
          }
        }
      }
    }
  }
}
```

Figure 2: Example: Asset Management and Tracking

4.1.2.2. Example: Environmental Context Awareness

This example corresponds to Section 3.2.2. It describes installation-related metadata for a building-mounted environmental sensor.

```
{
  "sdfObject": {
    "envSensor": {
      "description": "Environmental sensor unit",
      "sdfContext": {
        "installationInfo": {
          "type": "object",
          "properties": {
            "floor": { "type": "integer" },
            "mountType": {
              "type": "string",
              "enum": ["wall", "ceiling", "window"]
            },
            "indoorOutdoor": {
              "type": "string",
              "enum": ["indoor", "outdoor"]
            }
          },
          "required": ["floor", "mountType"]
        }
      }
    }
  }
}
```

Figure 3: Example: Environmental Context Awareness

4.1.2.3. Example: Regulatory Compliance and Certification

Aligned with Section 3.2.3, the following SDF model defines immutable identity and certification data of a regulated device.

```
{
  "sdfThing": {
    "deviceMetadata": {
      "sdfContext": {
        "identity": {
          "type": "object",
          "properties": {
            "manufacturer": { "type": "string" },
            "model": { "type": "string" },
            "firmwareVersion": { "type": "string" }
          }
        },
        "certifications": {
          "type": "array",
          "items": {
            "type": "object",
            "properties": {
              "scheme": { "type": "string" },
              "certId": { "type": "string" },
              "region": { "type": "string" }
            }
          }
        }
      }
    }
  }
}
```

Figure 4: Example: Regulatory Compliance and Certification

4.1.3. Using sdfType: link for External Resource References

Some context entries may refer to external documentation or digital artifacts. In such cases, it is beneficial to annotate the field with sdfType: "link", as defined in [I-D.ietf-asdf-sdftype-link]. This signals that the field's value is a resolvable URI or hyperlink.

Use of sdfType: "link" enhances clarity and enables tools to render these values as clickable links, validate their format, or connect the SDF model to a broader ecosystem of cloud or documentation platforms.

The following example illustrates how external references can be expressed within the sdfContext section using the sdfType: "link" construct. Each field in this example represents a URI or hyperlink that points to an external digital resource relevant to the device but not directly part of its runtime interface. The use of sdfType: "link" explicitly signals that the string value is a resolvable link,

which enables tools and user interfaces to render it accordingly (e.g., as a clickable URL or embedded resource reference). The optional "readable": true qualifier is included to emphasize that these fields are intended for display or documentation purposes.

* datasheetUrl: Points to the manufacturer’s datasheet, which typically contains technical specifications, electrical parameters, and physical characteristics of the device.

* certificationLink: Refers to a publicly accessible certification document such as safety approvals (e.g., CE, FCC) or compliance reports.

* digitalTwinUrl: Provides a reference to an external digital twin instance of the device, typically hosted in a cloud-based platform or digital thread system, allowing for visualization, monitoring, or synchronization.

These fields are not meant to be exposed through runtime protocol affordances such as CoAP or HTTP, but they enhance the expressiveness of the model for tooling, documentation, and system integration purposes.

```
            {
              "sdfContext": {
                "datasheetUrl": {
                  "type": "string",
                  "sdfType": "link",
                  "description": "Manufacturer datasheet for technical specifications",
                  "readable": true
                },
                "certificationLink": {
                  "type": "string",
                  "sdfType": "link",
                  "description": "Official link to product safety certification documen
t",
                  "readable": true
                },
                "digitalTwinUrl": {
                  "type": "string",
                  "sdfType": "link",
                  "description": "Reference to the digital twin instance in external pl
atform",
                  "readable": true
                }
              }
            }
```

Figure 5: Examples of using sdfType: "link"

4.2. Run-Time Context Messages

During operation, some contextual values change (e.g., a device is moved to a new room) or must be declared for audit purposes. To communicate those facts without re-classifying them as affordances, three transport-agnostic JSON envelopes for run-time context exchange are defined:

* contextSnapshot: conveys the full, current set of non-affordance fields-such as installation information or geographic coordinates-and is typically sent at boot, on request, or during periodic audits.

* identityManifest: declares immutable identity data (model, manufacturer, capability tags, certifications) and is normally issued once at commissioning or whenever a permanent attribute is added, for example after a firmware upgrade that introduces a new capability.

* contextPatch: transmits only the keys that have changed since the last snapshot, minimising bandwidth when a device is moved, re-mounted, or otherwise updated in context.

All envelopes carry a thingId and an timestamp.

4.2.1. contextSnapshot Message

The ’contextSnapshot’ message provides a complete view of a device’s static non-affordance metadata. This message is typically sent upon onboarding or registration to inform the system of all contextual properties.

```
{
  "thingId": "envSensor:abc123",
  "timestamp": "2025-07-01T12:00:00Z",
  "contextSnapshot": {
    "installationInfo": {
      "floor": 3,
      "mountType": "ceiling",
      "indoorOutdoor": "indoor"
    }
  }
```

Figure 6: Example of contextSnapshot Message

4.2.2. identityManifest Message

The 'identityManifest' message describes immutable identity attributes of a device or asset. It can be used for device authentication or registry lookup.

```
{
  "thingId": "medDevice:unit42",
  "timestamp": "2025-07-01T08:15:00Z",
  "identityManifest": {
    "manufacturer": "HealthTech Inc.",
    "model": "HT-2025-M",
    "firmwareVersion": "1.4.3",
    "certifications": [
      { "scheme": "FDA", "certId": "FDA123456", "region": "US" },
      { "scheme": "CE", "certId": "CE987654", "region": "EU" }
    ]
  }
}
```

Figure 7: Example of identityManifest Message

4.2.3. contextPatch Message

The 'contextPatch' message reports changes to specific non-affordance attributes. This allows for efficient partial updates without resending the entire snapshot.

```
{
   "thingId": "assetContainer:box001",
   "timestamp": "2025-07-01T14:23:00Z",
   "contextPatch": {
     "location": {
       "lat": 37.5665,
       "lon": 126.9780
     }
   }
 }
```

Figure 8: Example of contextPatch Message

5. Security Considerations

TBD

6. IANA Considerations

TBD

7. Normative References

[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-24, 27 July 2025, <https://datatracker.ietf.org/doc/html/draft-ietf-asdf-sdf-24>.

[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>.

Authors' Addresses

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

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