T2T Research Group
Internet-Draft
Intended status: Informational
Expires: 29 December 2022

M. Milenkovic, Ed.
IoTsense LLC

27 June 2022

IoT Information-Model Standards Description ("Nutrition Label")
draft-t2trg-iot-standards-description-00

Abstract

Implementation of IoT systems imposes a requirement of M2M semantic interoperability. This problem is addressed by a number of ongoing attempts to define and standardize IoT information and data models. At present this work is fragmented across several standards with different approaches, scope, objectives, terminology, and assumptions that makes them difficult to understand and compare. This document is part of the effort to alleviate that problem by means of more streamlined presentations and descriptions.

We are advocating for clear articulation of the intent and implicit or explicit assumptions in SDO specifications using the common terminology. Such clarifications would aid IoT practitioners and potential adopters to make meaningful comparisons and facilitate selection of IoT specifications that are the best fit for their intended purpose. To that end, we propose that creators of IoT standards address a list of questions that would characterize their work in comparable ways, somewhat akin to the intent of nutrition labels for food.

This paper describes the basic design principles and practices of IoT information and data model definitions, and proposes an initial list of questions that SDOs may address to facilitate understanding and comparisons. Our intent is to evolve and refine this list over time by actively soliciting and incorporating feedback and suggestions of IoT community.

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/.

Copyright Notice

Table of Contents

1. Whom is This Document For?

This document is intended to encourage Standard Development Organizations (SDOs) involved in definition of IoT data and information models to provide clarifications in their documents that would aid better understanding of their purpose, objectives, and approaches to the problem. We believe that this would be useful to the technical community of readers and potential implementors who are not directly involved in SDO work and thus do not share the common frame of reference and understanding of terminology that the group members tend to evolve and adopt in the course of their meetings and interactions. When used in specifications without clarifications, implied concepts and assumptions as well as peculiar use of terminology can result in specifications that appear obtuse and difficult to read and understand to the "uninitiated".

We are advocating for clear articulation of the intent and implicit or explicit assumptions in specifications using the common terminology. Such clarifications would aid readers and potential adopters to make comparisons and facilitate selection of IoT standards that are the best fit for their intended purpose.

In this document we offer some specific suggestions for the questions to be covered. Some categories of users who would benefit from such clarifications are listed below.

IoT researchers and practitioners may want to understand the landscape of relevant IoT standards and principles of information modeling and interoperability in IoT systems.

An IoT system designer/architect who has requirements for her specific IoT application and is looking to select an IoT specification with compatible architectural style and suitable usage domain. Such users are often concerned with being able to future-proof their design in the sense of being able to establish and retain the ownership and use of their collected data even if the system needs to be modified, extended, or migrated to another IoT platform.

An IoT system implementer who is considering adoption of a potential IoT standard and would like to assess potential implementation complexity in terms of required node behaviors and in providing the assumed environmental requirements - such as protocols and security - for them to operate. This type of user may be interested in the availability of reference implementations, tools, and restrictions such as specific language bindings.

2. Introduction and Problem Statement

Internet of Things (IoT) systems deploy smart connected things that sense their environment and generate quantitative data about the physical world. IoT systems essentially interface the Internet to the physical world.

ITU defines IoT as ... "the extension of Internet connectivity into physical devices and everyday objects. Embedded with electronics, Internet connectivity, and other forms of hardware (such as sensors), these devices can communicate and interact with others over the Internet, and they can be remotely monitored and controlled. The IoT adds the ability for IoT devices to interoperate with the existing Internet infrastructure." [ITU]

In IoT systems, communication is primarily between machines that generate and consume the data. They exchange message conveying data measurements and actuation commands. This machine-to-machine (M2M) mode of operation requires semantic interoperability in the sense that receiving nodes need to be able to "understand" the meaning of data. Defining meaning and understanding is generally a philosophical undertaking. In practical terms as applied to M2M, this means that the communicating machines need a common agreement on how to model and semantically annotate the behaviors and properties of the physical things in order to represent, communicate, and interpret the data that they generate. In other words, all communicating parties need to use the same conceptual model of things that exist in their domain. One way to accomplish this is by having all parties use the same specification of an IoT information model.

Note that machine-level semantic interoperability is not required in the common usage of worldwide web where humans interpret the meaning of the presented web pages and thus achieve semantic interoperability with data (content) creators [NB add interop harder ref later].

3. IoT Information Model

[NB: this is a context setting section, but not the thrust of the paper, so may need to be trimmed??..]

An IoT information model is an abstract, formal representation of IoT thing types that often include their properties, relationships, and the operations that can be performed on them. A thing information model needs to specify what the thing is, what it can do, thing properties and their values, and ways to interact with it. Some specifications provide only definition of thing types and their properties and they tend to be referred to as data models. Distinctions between information and data models are discussed in [RFC3444].

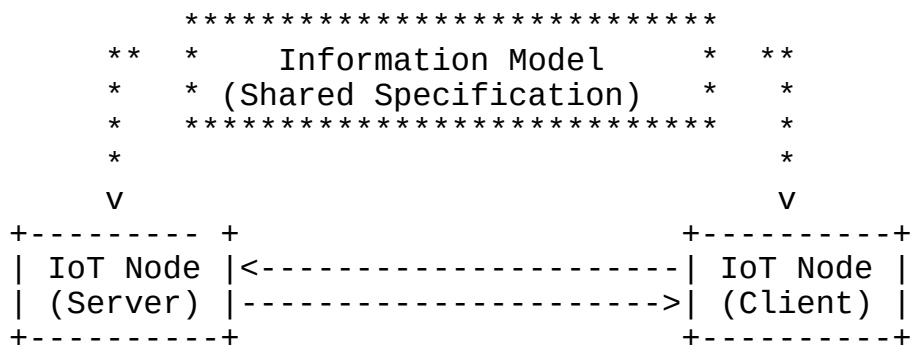

Figure 1: Use of a Shared Information Model

Figure 1 illustrates two endpoints in an IoT system that intend to engage in data and control exchanges([iotbook]). As indicated earlier, all communicating parties need to use the same conceptual model of things that exist in their domain. A common way to accomplish this is by having all parties use the same specification of an IoT information model.

One of the primary tasks of an IoT information model is to facilitate semantic interoperability, i.e. a shared understanding of what the data means. It is used by the IoT server to provide the compliant software abstraction of the thing, and by IoT clients to properly interpret it and be able to process the data accordingly.

Following the Internet principles and practice, clients and servers exchange payloads and rely on an information-model specification for proper encoding and interpretation of the exchanged messages. The two endpoints are otherwise decoupled, i.e. they may be developed independently of each other and operate on different platforms and runtime environments. In this context, endpoint refers to software agents that are acquiring (producing) or processing (consuming) data.

The IoT server internally implements the thing description as a cyber-world touch point for each particular instance using the format of the information model defined by the shared specification. It also needs to implement the actions associated with the device when requested by the authorized parties. Depending on the system configuration and hardware capability, an IoT server may physically reside on the thing or on a proxy device, such as a directly attached gateway.

IoT clients are consumers of thing/server data and generators of actuation commands. IoT clients are commonly software agents associated with or acting on behalf of services and applications. A client can access server thing descriptions that it is authorized to and use them to issue queries and optionally command messages. The client uses the shared information-model definition to properly format requests and interpret responses that it receives. IoT clients may reside at in at the edge - such as a peer thing node or edge gateway, at some intermediate point in the system hierarchy such as a fog node, or in the cloud.

Since objects are modeling real-world things, there are some implied semantics defined by the intrinsic nature of the thing that is being modeled. For example, a temperature sensor is commonly understood to measure temperature of something based on its characteristics and placement, such as the ambient air temperature or water temperature in a heating/cooling pipe.

3.1. Structure of an IoT Information Model

Software objects that represent and model IoT things are often structured in ways that follow the common practices and terminology in object-oriented (OO) programming. They are also referred to by other names, such as thing descriptions and smart objects. Table 1 illustrates general the form of object-oriented IoT information model representations.

Element	Description
Object Type	Physical thing being modeled (device)
Properties, Attributes	Object attributes, data, metadata
Interactions	Ways to interact with object, actions, events
Links	To other objects, compositions and collections

Table 1: Structure of IoT Information Model

Some IoT data models specify only object types and properties of modeled things and leave out definition of interactions and links.

3.1.1. Object Type

Within each specification, names of thing abstractions indicate the type of the physical thing or the phenomenon that is being modeled. This field is labeled as Object Type in Table 1 and is often modeled in object-oriented systems as a class. Object and class types directly correspond to the specific real-world things that are modeled by the specification. They can include a variety of sensors and devices, such as temperature, humidity, thermostat, refrigerator, security camera, current drawn, metered units (kWh, gals or liters) and many others. It is customary to use a single object-class definition representing a category of physical devices or things, such as a temperature sensor, and to create specific object instances of that class for each individual physical instantiation of a thing in a particular IoT system.

Information-model specifications and standards explicitly name and define each type of the physical entity that they model. Those names, in effect, constitute a vocabulary of types with their related definitions. A vocabulary may be a simple list of defined names resembling a dictionary. In some specifications, the naming tree may be structured as a as a hierarchy to form a taxonomy to indicate possible classification relationships among the objects, and to guide the proper placement of future objects as they are defined.

To facilitate unambiguous machine parsing, the vocabulary provides a precise definition of how terms are to be named and used when referring to a specific object class, such as temp or temperature. By using the type names exactly as defined in the vocabulary,

communicating parties can establish an accurate correlation of references at both sending and receiving ends. This is a simple and effective way of establishing semantic interoperability at object type and naming level.

3.1.2. Properties, Attributes

Attributes generally say something about the object and its state. This field is commonly used to represent sensor readings, such as for a temperature sensor its current reading in the applicable units of measure, e.g. degrees C or F. Standards usually specify the exact format of the name fields and their meaning in the vocabulary. They also specify the data types for the values associated with the particular name fields - such as numbers, integers, or strings. Values or attributes in thing abstraction representations can be read-only, such as temperature reading, or writeable for actuation, or both. They can also include metadata, such as what the sensor is measuring, e. g. air or water temperature, minimum and maximum possible values of reading, and the like.

3.1.3. Interactions

Some information models define the types of interactions that a modeled object supports. These can include interfaces and methods that implementations may support, such as the types of requests and responses for reading data and for issuing actuation commands. They can also include designations of protocols and formalisms, usually APIs, to interact with the thing. Device supported APIs usually include retrieval of attributes and values, such as sensor readings and metadata. They may also include ways to request machine-readable descriptions of the thing, its object type and characteristics, or to activate the built-in methods that operate on the object’s internal structures and outputs, including actuations. Many specifications also define events, which are generally asynchronous signals emitted by the device to indicate some change of state under observation.

3.1.4. Links

Some sort of linking mechanism or representation is often included in IoT information models, used primarily as a mechanism to point to other items of interest or to form groupings of related objects. Groupings of interest may be formed to facilitate coordinated behaviors of multiple devices, such as home-automation things. They may also be used to reflect structural properties of the physical connections between devices as installed and configured in the field that may be of interest to applications, such as indicating which room and HVAC zone a thermostat may belong to.

Composition via linking may also be used to describe composite objects that include some more primitive basic types already defined in the specification. For example, a thermostat object type may be constructed as a composition of a temperature sensor, set point for temperature regulation and scheduling, and actuators of the heating and cooling systems.

3.2. Functional Interoperability

In addition to using the same information model, in order to interoperate IoT endpoints also need to use the common set of protocols and serialization mechanisms, compatible naming and addressing conventions and operate in the common security perimeter.

A working IoT system needs to provide a common operational environment for nodes to communicate. Some of its components may be defined by standards and other specified by convention or a particular system implementation. Environmental components needed for functional interoperability generally include:

* Data (information) model
* Payload serialization
* Protocol bindings
* Naming and addressing, discovery
* Security
* Device management and provisioning

Payload serialization refers to the common agreement on the format in which messages appear "on the wire", i.e. are formatted by the sending node and parsed by the receiving one. Variants of JSON are commonly used for this purpose, although more compact forms of serializations, such as the binary CBOR [RFC8949] may be favored for constrained nodes and networks.

Protocol bindings specify which protocols are supported and perhaps the port numbers on which they operate. In addition to the common Internet protocols, more compact versions, such as CoAP ([RFC7959]) may be favored for constrained nodes or segments of the network.

Naming and addressing refer to the mechanisms used to access nodes in the system. They usually resolve to network IP addresses, but may also include higher-level mapping constructs such as URLs and URIs.

Discovery provides means to identify other nodes in the system to communicate with or to form groupings of interest. They can include discovery through network scanning to identify nodes. This is usually combined with mechanisms and conventions to access machine-readable descriptions of the nature and capabilities of nodes at actively used addresses. Another approach is to create directories where nodes get registered during the activation process and are discovered by queries.

A working IoT system typically implements security requirements, conventions, and protocols for node authentication, access authorization, and secure communication. In order to access each other and to communicate, all nodes must adhere to the rules and use the commonly prescribed system procedures. Security system usually maintains and updates security postures and policies of IoT nodes and monitors their activity to detect and deal with anomalous behaviors such as intrusions and probing.

Device management, like security, is part of the control plane of an IoT system. It is often implemented to keep the system operational, track states of its components - such as active or inactive - and often manages distribution and application of software and firmware updates.

3.3. IoT Frameworks

IoT frameworks are more prescriptive forms of IoT standards intended to provide full operational interoperability by extending their scope of definition beyond the data and information model to include specification of the complete operating environment for the compliant IoT nodes. In addition, frameworks may include bridges and protocol converters to facilitate interoperability with the devices and things that are using conventions and legacy protocols that may not be Internet compatible. IoT frameworks generally operate above the transport layer, but below the application layer ([IIC]).

Framework definition of data and information models typically include object type definitions, properties and interactions, with types of requests and responses, supported interfaces, methods, and access points for those.

IoT frameworks also tend to define the conventions, namespaces, and mechanisms to be used for the thing naming and addressing. Object instance name may be correlated to its addressing mechanism, so that knowing the name can be used to locate the target object directly or by consulting an appropriate thing directory.

Frameworks may also specify security requirements for node connectivity and compliance. These may include the required and sanctioned types of authentication, authorization, encryption, as well as supported security protocols, such as TLS.

Some frameworks also specify elements of device management, such as the forms of health and status reporting, software and security updating, and methods for initial device onboarding and provisioning.

As indicated, IoT frameworks can define many aspects of an IoT system, well beyond the information model. Nodes wishing to interoperate within a framework must implement all of its applicable components. The positive side of using frameworks is that they can assure complete operational interoperability of nodes in an IoT system. In addition to the framework specification, some SDOs also provide its instantions in terms of sanctioned software and middleware implementations. Some SDOs also provide tools and facilities for formal certification of vendor product compliance that should guarantee interoperability.

On the negative side, use of a framework requires adherence to all of its mandatory parts which may be a problem when its specification is at odds with the design preferences and requirements of a particular IoT system. Another practical problem is that framework compliance may limit system implementations to only those thing types that are defined in its specification.

4. Characteristics List

[NB: to be revised based on group feedback and consensus]

A partial list of recommended charateristics that documents and specifications of IoT data and information models should explicitly state (or articulate?).

* Objective
* Domain
* Environmental Assumptions
* Terminology
* Would be Nice to Know

4.1. Objective

What is the objective of the specification? What does the specification cover? It may be to define data models or information models for IoT endpoints (things) in general or in a particular domain. Or it may be a meta-model specification providing thing abstractions that the specific information and data models can be mapped to and from. Or it may be a framework, in which case it should be specified which additional aspects of the operational environment it defines, such as security, naming, discovery, communication protocols, and serialization methods.

4.2. Domain

What is the target usage domain for the specification? A specification may target or be optimized for a particular domain, such as smart home, healthcare, industrial, manufacturing, transportation, agriculture. Choice of the domain is reflected in the specific types of objects and things for which the information model is defined. While the specification charter and ambitions may make broader claims, the types of actually defined information models effectively determine what it may be used for. Domains of the specified objects should be stated. Identifying target domains for future iterations of the specification would also be helpful.

4.3. Environmental Assumptions

As indicated earlier, in order to interoperate IoT endpoints need to use not only a common information model, but also a compatible or common set of communication protocols, data serialization mechanisms, naming and addressing conventions, and operate in the common security perimeter. In effect, these are environmental conditions in which the nodes need to operate in order to achieve full interoperability.

IoT standard needs to indicate which of these elements are included in its specification and which elements are required for its proper implementation. It should also indicate what is assumed, implicitly or explicitly, to be provided in terms of specific protocol bindings and serialization methods, naming, and discovery of nodes.

IoT standards should also state which parts they intentionally do not cover, say security, to indicate that those need to be defined and implemented separately if required.

4.4. Terminology

Different standards groups tend to use different terminology and sometimes even use the same terms to refer to different concepts. This can cause confusion in understanding and interpreting their specifications. It is therefore advisable for specifications to define their key terms early on. This does not need to be a formal dictionary, a simple identification of the terms used to refer to elements of the IoT information model depicted in Table 1 would go a long way towards accomplishing this goal.

4.5. Would be Nice to Know

[NB: should this be a separate indented list of categories or a simple continuation of the previous one?]

While not strictly necessary for understanding the basic tenets of a specification, it would be useful to address the additional items that are generally of interest to potential adopters. These may include:

* Metadata Handling
* Creation of New Object Types
* Development Considerations and Assumptions
* Tools, Reference Implementations
* Licensing Requirements and Terms

4.5.1. Metadata Handling

Metadata, also called tags, are used in IoT systems provide contextual semantics and create more useful data for a variety of post-processing services and applications, such as analytics and device/asset management. Metadata annotations can provide enriched contextual information in the form of additional attributes of a thing’s functionality, geographic location, manufacturer, serial number, and the like.

Information and data models often specify some metadata as object properties or attributes. These are typically inherent properties of the things being modeled, such as the units of measure used by a temperature sensor when reporting its readings. Some IoT applications may benefit from or even require additional metadata, such as on the structural relationships among things that are determined later in the system lifecycle, e.g. at installation time.

For example, additional metadata may indicate that an air temperature sensor in a specific room is part of an HVAC zone whose ambient conditions are regulated by a named air-handling unit.

This kind of information may be consumed by an application to programmatically determine which HVAC components to actuate in order to reduce or increase ambient temperature as dictated by the the associated thermostat schedule or a user request. Without the metadata, knowledge of structural relations among nodes would have to be hard coded into an application and customized for each specific installation, resulting in implementations that are brittle, not portable, time consuming, and error prone.

It would be useful to know if creators of the particular information model have a mechanism for or a position on handling additional metadata beyond its basic specification. [NB: maybe mention haystack as an example of adding free-form metadata]

4.5.2. Creation of New Object Types

Data and information models provide definitions for a number of object types representing things in their target domain. Their number and variety is limited to what the defining organization sees as important, has the bandwidth to specify, and can reach a consensus on. The number and type of covered devices tends to change and grow over time, as reflected in evolving revisions of the specification.

This can present a problem to adopters who wish to use the specification, but need to incorporate IoT device types that are not (yet) defined in it. Such users need to define a device model, preferably with a prospect of being compliant or at least subject to minimal modifications as the specification evolves to include new types.

Various SDOs have different approaches to definition of new models. Some simply wait until agreeing on thing types to be defined in future iterations. Others encourage crowd-sourcing of new models, where expert groups or individual companies propose new models, often after implementing and proving them in internal use. Naturally, such proposals need to be consistent with the architectural and design principles of the models that are already defined. Some SDOs (ref OCF oneIoTa) even provide the tools for creation and evaluation of machine-readable definitions of the proposed new models. Naturally, the proposals have to be eventually approved by the SDO and modified if necessary for adoption. In general, crowdsourcing of thing models tends to accelerate the coverage and encourage experimentation and innovation by interested adopters.

Creators of specifications of IoT data models should indicate their process for creation of new object types. It would also be helpful to provide a recommendation on how to deal with them in the interim.

4.5.3. Development Considerations and Assumptions

Specifications typically define the details of the types and format of legal node behaviors and their actions and interactions, often in the form of well-formed requests and expected or permissible replies. We posit that implementation developers would be well served by summarizing the totality of what a compliant node is required and assumed to be able to do. Additional clarification may be provided by what is possible and assumed to happen at the two distinct parts of the node lifecycle - design time when the code is written and runtime when the node is in operation.

At the design time, the code is written to implement the applicable node behaviors defined by the specification. This includes all mandatory parts of the specification, and compliant implementation of the chosen subset of the optional parts of the specification that node designers deem relevant and appropriate for the purposes of their target application.

At runtime, when the node is in operation, it can probe other nodes (that it knows of and is authorized to access) to determine potential existence of optional elements of the specification that the two nodes may be able to engage in if the necessary support is provided at both ends. Defined actions of runtime behavior are commonly provided by the specification and their existence and support discovered by runtime queries.

Some specifications and frameworks go further by providing mechanisms and conventions for nodes to discover behaviors that they have not previously "seen" (programmed to deal with) and to learn how to deal with them at runtime. The idea is that ontologies and machine-readable specifications may be used and provided in the system for nodes to "learn" what to do as and when necessary. Such elements may not be explicitly defined in the specification, but they may be inferred or learned from the associated and perhaps separately evolved ontologies. This implies that the node that does the query may not have the requisite pre-programmed (at design time) interactions, but needs to have the capability to query the encodings of knowledge and construct appropriate behaviors at runtime.

Specifications that include these possibilities do not often explain their scope and assumed operation very well. On the other hand, such elements tend to be quite complex and difficult to implement and test. Clearly articulating what is intended and assumed, and how to

properly use this capability when present, would go a long way towards improving the chances of correct and interoperable implementations.

4.5.4. Tools, Reference Implementations

Provide functional description of any additional tools or reference implementations that may be available to developers. Describe language dependencies or preferences, if any, that may result from the specific bindings or reference implementations.

4.5.5. Licensing Terms

Indicate the terms and conditions for implementing a particular specification. These may include requirements for membership in the associated SDO, licensing fees, or a mandate to share any modifications and enhancements with the community. Differences, if any, for licensing research vs. commercial implementations should be pointed out.

5. Acknowledgements

Add acks, contributors etc

6. IANA Considerations

This memo includes no request to IANA RFC 5226 [RFC5226].

7. Security Considerations

All drafts are required to have a security considerations section. See RFC 3552 [RFC3552] for a guide.

8. Appendices

When available, include descriptions of contributed appendices that describe objectives and scope of specifications contributed by SDOs or volunteers.

9. Informative References

[IIC] Industry IoT Consortium, "The Industrial Internet of Things Connectivity Framework", IIC , 2022.

[iotbook] Milenkovic, M., "Internet of Things: Concepts and System Design", Springer Nature Publishing , 2020.

[ITU] International Telecommunications Union, "Overview of the internet of things", ITU-T Y.4000/Y.2600, 2016.

[RFC3444] Pras, A. and J. Schoenwaelder, "On the Difference between Information Models and Data Models", RFC 3444, DOI 10.17487/RFC3444, January 2003, <https://www.rfc-editor.org/info/rfc3444>.

[RFC3552] Rescorla, E. and B. Korver, "Guidelines for Writing RFC Text on Security Considerations", BCP 72, RFC 3552, DOI 10.17487/RFC3552, July 2003, <https://www.rfc-editor.org/info/rfc3552>.

[RFC5226] Narten, T. and H. Alvestrand, "Guidelines for Writing an IANA Considerations Section in RFCs", RFC 5226, DOI 10.17487/RFC5226, May 2008, <https://www.rfc-editor.org/info/rfc5226>.

[RFC7959] Bormann, C. and Z. Shelby, Ed., "Block-Wise Transfers in the Constrained Application Protocol (CoAP)", RFC 7959, DOI 10.17487/RFC7959, August 2016, <https://www.rfc-editor.org/info/rfc7959>.

[RFC8949] Bormann, C. and P. Hoffman, "Concise Binary Object Representation (CBOR)", STD 94, RFC 8949, DOI 10.17487/RFC8949, December 2020, <https://www.rfc-editor.org/info/rfc8949>.

Appendix A. Additional Stuff

Placeholder for SDOs or volunteers to provide their descriptions that meet the suggested guidelines

Author’s Address

Milan Milenkovic (editor)
IoTsense LLC
Dublin, CA 94568
United States of America
Phone: +1 650 431 8280
Email: milan@iotsense.com