The Open Group Standard

Open Data Format (O-DF), The Open Group Standard for the Internet of Things (IoT), Version 2.0

The Open Group Standard

Open Data Format (O-DF), The Open Group Standard for the Internet of Things (IoT), Version 2.0

ISBN: 1-947754-41-6

Document Number: C19D

Published by The Open Group, December 2019.

Comments relating to the material contained in this document may be submitted to:

The Open Group, Apex Plaza, Forbury Road, Reading, Berkshire, RG1 1AX, United Kingdom

or by electronic mail to:

ogspecs@opengroup.org

Contents

List of Figures

List of Examples

Preface

The Open Group

The Open Group is a global consortium that enables the achievement of business objectives through technology standards. Our diverse membership of more than 700 organizations includes customers, systems and solutions suppliers, tools vendors, integrators, academics, and consultants across multiple industries.

The mission of The Open Group is to drive the creation of Boundaryless Information Flow™ achieved by:

- Working with customers to capture, understand, and address current and emerging requirements, establish policies, and share best practices
- Working with suppliers, consortia, and standards bodies to develop consensus and facilitate interoperability, to evolve and integrate specifications and open source technologies
- Offering a comprehensive set of services to enhance the operational efficiency of consortia
- Developing and operating the industry's premier certification service and encouraging procurement of certified products

Further information on The Open Group is available at www.opengroup.org.

The Open Group publishes a wide range of technical documentation, most of which is focused on development of Standards and Guides, but which also includes white papers, technical studies, certification and testing documentation, and business titles. Full details and a catalog are available at www.opengroup.org/library.

This Document

This document is The Open Group Internet of Things (IoT) Standard for the Open Data Format (O-DF), Version 2.0. It has been developed and approved by The Open Group.

This document specifies Version 2.0 of the O-DF standard. It is based on Version 1.0 that was published in 2014 with Technical Corrigenda published in May 2016 and September 2017. Particular care has been taken to, as far as possible, keep Version 2.0 backwards-compatible with earlier versions.

This document is structured as follows:

- Chapter 1 provides an introduction to the standard and describes conformance requirements, normative references, and terminology
- Chapter 2 contains Definitions

- Chapter 3 describes the O-DF elements and attributes and how they are expected to be used
- Chapter 4 provides detailed instructions on how the *type* attribute should be used for referring to external vocabularies
- Chapter 5 describes how O-DF elements can be accessed directly using the HTTP GET method
- Chapter 6 explains how to create domain-specific data models that are extensions of the O-DF standard
- Appendix A contains the XML Schema file
- Appendix B contains example O-DF structures
- Appendix C shows possible JSON mappings

Trademarks

Acknowledgements

The Open Group gratefully acknowledges the contribution of the following people in the development of this standard.

Contributing Members of The Open Group IoT Work Group

- Kary Främling, Aalto University* (IoT Work Group Chair)
- Chris Harding, Lacibus
- Robert Hellbach, BIBA
- Lauri Isojärvi, Aalto University
- Tuomas Kinnunen, Aalto University
- Prodromos Kolyvakis, EPFL
- Jussi Pirilä, Aalto University
- Edward Roberts, Elparazim
- Ron Schuldt, Data-Harmonizing LLC
- Min-Jung Yoo, EPFL

* Primary Contributor

The Open Group Staff

- Michelle Supper, Director, The Open Group IoT Work Group

Referenced Documents

(Please note that the links below are good at the time of writing but cannot be guaranteed for the future.)

Normative References

Normative references for this standard are defined in Section 1.4.

Informative References

The following document is referenced in this standard:

- An Introduction to Internet of Things (IoT) and Lifecycle Management, White Paper (W167), May 2016, published by The Open Group; refer to: www.opengroup.org/library/w167

1 Introduction

1.1 Objective

The Open Group Standards for the Internet of Things (IoT) have been developed to fill an interoperability gap identified in the context of the IoT, as explained in the White Paper: An Introduction to Internet of Things and Lifecycle Management.

A great number of useful standards exist on the level of communications within a local network, within a specific domain, or for a limited purpose such as remote management of computers. However, at the time of writing Version 2.0 (as when writing Version 1.0 in 2014), we have not been able to identify an appropriate standard that would address the higher-level requirements of the IoT. Such requirements are notably the need for any data sources (devices, machines, server-based systems, etc.) to be able to publish their available data and provide access to it in a generic way, independently of the domain. As pointed out in the aforementioned White Paper, the O-DF standard has been specified for requirements of so-called System-of-Systems (SoS)-level interoperability, where it should be possible to request, use, and combine information from multiple information sources, even when that information comes from multiple organizations, domains, clouds, devices, or any other connected systems.

The O-DF standard can be used for publishing the available data using ordinary URL (Uniform Resource Locator) addresses. O-DF structures can also be used for requesting and sending published data between systems, notably when used together with The Open Group Open Messaging Interface (O-MI) standard.

In the IoT, information about a product or a "Thing" is often distributed over many different devices, systems, and organizations. The O-DF standard is intended to represent information about things in a standardized way that can be understood and *exchanged* in a universal way by all information systems that need to manage such IoT-related data. In the O-DF standard, "Things" are represented by the *Object* data structure. Every *Object* has at least one identifier. The visibility and the access to O-DF represented data may depend on the object identifier used, as well as on the identity of the requesting party and on the context of the request. This is why the object identifier is of particular importance in any universal IoT standard.

1.1.1 Compatibility with Earlier Versions of the O-DF Standard

Version 2.0 of the O-DF standard has the following new features compared to Version 1.0:

- *InfoItem* now has a *type* attribute
- A *prefix* attribute has been added for *Objects*, *Object*, and *InfoItem*
- Updated and clearer instructions for the use of *type* attribute with different vocabularies
- A section describing the use of the Open Data Element Framework (O-DEF™) standard in O-DF notation

- Consistently allowing the use of *anyAttribute* where appropriate, which makes it possible to add attributes that are not covered by the O-DF standard but are needed for other purposes
- Specification of how to request for a specific O-DF version with HTTP GET

The following changes have been made to the XML Schema file that may (but most probably will not) cause compatibility issues with Version 1.0:

- The <*name*> sub-element of *InfoItem* has been changed to <*altname*>

  The main reason for this change is that the Java® Architecture for XML Binding (JAXB) tool (and potentially others) did not deal graciously with having a "name" attribute and "name" sub-elements.

- The type *QlmID* has been renamed *IoTIdType*

  This should not cause any compatibility issues between implementations using different O-DF versions. The name change is mainly motivated by feedback about inconsistency with other type names. *QlmID* was used mainly for historical reasons.

- The type of the attribute *unixTime* has been changed from xs:long into xs:double

  This signifies that Version 2.0 implementations will interpret Version 1.0 UNIX® times correctly. However, a Version 1.0 implementation may not interpret a Version 2.0 UNIX time correctly. The main reason for this change is to avoid future overflow with some programming language implementations, as well as to allow the use of fractions of a second.

1.2 Overview

The O-DF standard is specified using XML Schema. It defines a simple and extensible ontology which allows the creation of information structures that are similar to those of objects and properties in object-oriented programming. It is intended to be generic enough for the representation of any object and information that is needed for information exchange in domains such as the IoT, lifecycle information management, etc.

An O-DF structure is a hierarchy with an *Objects* element as its top element, as illustrated in Figure 1. The *Objects* element can contain any number of *Object* sub-elements. *Object* elements are identified by at least one *id* sub-element. An *Object* may also have an optional *description* sub-element. *Object* elements usually have properties, which are sub-elements called *InfoItem*, as well as *Object* sub-elements. *InfoItems* are identified by a *name* attribute. The resulting *Object* tree can contain any number of levels. Chapter 3 provides detailed, normative descriptions of these elements.

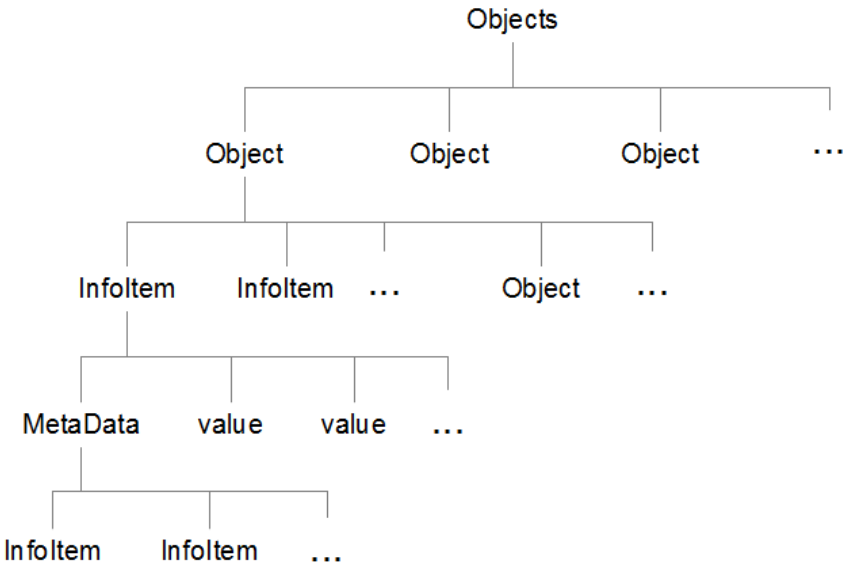

Figure 1: Illustration of O-DF Element Hierarchy

The O-DF standard is intended to be used for expressing information about "any" identifiable object (products, services, humans, etc.). How the information is communicated is not part of this standard. The most common communication media are presumably HTTP, HTTPS, WebSockets, and other so-called Web standards. However, the communication media could also be a file sent as an email attachment, on a USB stick, or any other kind of media. O-DF content can also be sent using REST-based services, SOAP, Java Message Service (JMS), the O-MI standard, and other kinds of messaging protocols.

Especially when used together with the O-MI standard, the O-DF standard can be used as a query and response format; for instance, the O-MI standard specifies that a *"read"* request with an O-DF structure should be responded to with the next level in the hierarchy shown in Figure 1. As an example, a request with only an *"Objects"* element should return an O-DF response with the list of *Object* elements available, including at least the compulsory attributes and sub-elements (notably at least one *id* element).

When O-DF data structures are exchanged between different information systems, it is recommended to send only the "interesting" parts, such as specifying what information about an *Object* is requested or what information about an *Object* has been updated. The *Object id* and *InfoItem name* specify what *InfoItem* of what *Object* the operation refers to. For instance, if a sensor reading has been updated, then an O-DF structure with only one *Object*, one *InfoItem*, and one *value* would be needed.

1.3 Conformance

This standard specifies conformance requirements for use of the O-DF standard. Chapters 3, 4, and 5 and Appendix A are normative. Appendices B and C are informative.

1.4 Normative References

The following standards contain provisions which, through references in this standard, constitute provisions of the O-DF standard. At the time of publication, the editions indicated were valid. All standards are subject to revision, and parties to agreements based on this standard are encouraged to investigate the possibility of applying the most recent editions of the standards listed below. Members of IEC and ISO maintain registers of currently valid International Standards.

- IETF RFC 2045: Multipurpose Internet Mail Extensions (MIME) Part One: Format of Internet Message Bodies, November 1996; refer to: https://www.ietf.org/rfc/rfc2045.txt
- IETF RFC 3986: Uniform Resource Identifier (URI): Generic Syntax, January 2005; refer to: https://www.ietf.org/rfc/rfc3986.txt
- ISO 639: Language Codes; refer to: https://www.iso.org/iso-639-language-codes.html
- Open Data Element Framework (O-DEF™), Version 1.0, The Open Group Standard (C163), May 2016, published by The Open Group; refer to: www.opengroup.org/library/c163
- Open Data Format (O-DF), The Open Group Internet of Things (IoT) Standard (C14A), October 2014 (with Technical Corrigenda updates in May 2016 and September 2017), published by The Open Group; refer to: www.opengroup.org/library/c14a
- Open Messaging Interface (O-MI), The Open Group Internet of Things (IoT) Standard (C14B), October 2014 (with updates in May 2016 and September 2017), published by The Open Group; refer to: www.opengroup.org/library/c14b
- Open Messaging Interface (O-MI), Version 2.0, The Open Group Internet of Things (IoT) Standard (C19E), December 2019, published by The Open Group; refer to: www.opengroup.org/library/c19e
- RDFa Core 1.1 – Third Edition, Syntax and Processing Rules for Embedding RDF through Attributes, W3C Recommendation, March 2015; refer to: https://www.w3.org/TR/rdfa-core/
- XML Schema Part 2: Datatypes Second Edition, Eds. P.V. Biron, A. Malhotra, W3C Recommendation, October 2004; refer to: www.w3.org/TR/xmlschema-2

1.5 Terminology

For the purposes of this standard, the following terminology definitions apply. When used in this way, these terms will always be shown in ALL CAPS; when these words appear in ordinary typeface, they are intended to have their ordinary English meaning.

Can — Describes a permissible optional feature or behavior available to the user or application. The feature or behavior is mandatory for an implementation that conforms to this document. An application can rely on the existence of the feature or behavior.

May — Describes a feature or behavior that is optional for an implementation that conforms to this document. An application should not rely on the existence of the feature or

behavior. An application that relies on such a feature or behavior cannot be assured to be portable across conforming implementations. To avoid ambiguity, the opposite of “may” is expressed as “need not”, instead of “may not”.

Must — Describes a feature or behavior that is mandatory for an application or user. An implementation that conforms to this document shall support this feature or behavior.

Shall — Describes a feature or behavior that is mandatory for an implementation that conforms to this document. An application can rely on the existence of the feature or behavior.

Should — For an implementation that conforms to this document, describes a feature or behavior that is recommended but not mandatory. An application should not rely on the existence of the feature or behavior. An application that relies on such a feature or behavior cannot be assured to be portable across conforming implementations. For an application, describes a feature or behavior that is recommended programming practice for optimum portability.

Will — Same meaning as “shall”; “shall” is the preferred term.

1.6 Future Directions

This standard will continue to be maintained by the IoT Work Group of The Open Group Open Platform 3.0™ Forum. It may be revised to correct errors or to incorporate changes based on implementation experience.

2 Definitions

Merriam-Webster's Collegiate Dictionary should be referenced for terms used in this standard.

3 O-DF Elements

The XML Schema in Appendix A provides a formal specification for the O-DF standard. If there is any conflicting information between the schema and other parts of this standard, then the information in Appendix A is to be used.

The following units or formats SHALL be used to express the listed quantities. The *xs* namespace is defined in XML Schema Part 2: Datatypes Second Edition.

Table 1: O-DF Units and Formats

Value	Unit/Format
Date	Use xs:Date
Time	Use xs:dateTime
Duration	Use xs:duration It is defined in the schema where xs:duration should be used. For other durations, such as Time-To-Live (TTL), xs:double may be used and then the unit is seconds.
UNIX Time	Number of seconds elapsed since midnight proleptic Coordinated Universal Time (UTC) of January 1, 1970, not counting leap seconds. Because most programming languages provide millisecond accuracy, the type of the *unixTime* attribute has been changed into xs:double since Version 2.0 of the O-DF standard.
Other Values	SI units used by default.

3.1 The *Objects* Element

As shown in Figure 1, all O-DF structures SHALL have *Objects* as their root element, which is of type *ObjectsType*. An *Objects* element SHALL only have *Object* sub-elements. Example 1 shows a simple example of an O-DF structure.

Example 1: O-DF Structure

```
<Objects>
  <Object type="Air Handling Unit">
    <id>AHU123456789</id>
    <InfoItem name="Outside air temperature">
      <description>Outside air temperatures with timestamp.</ description >
      <value dateTime="2018-10-26T15:33:21">15.5</value>
      <value dateTime="2018-10-26T15:33:50">15.7</value>
    </InfoItem>
  </Object>
</Objects>
```

Objects MAY have a *version* attribute that corresponds to the O-DF version used.

Objects MAY have a prefix attribute that is used for expressing compact URI expressions as specified in the RDFa standard. The following example is given in the RDFa standard:

```
prefix="foaf: http://xmlns.com/foaf/0.1/ dc: http://purl.org/dc/terms/"
```

The prefix mechanism is needed because XML namespaces only apply to element names, not values. A prefix SHALL be used with attribute values like this: *type=”foaf:topic”*, which is then to be interpreted as *type=“http://xmlns.com/foaf/0.1/topic*”.

Other attributes for *Objects* MAY be included but their use is not specified.

3.2 The *Object* Element

An *Object* MAY have a type, which is indicated by the value of the *type* attribute. If possible, the value of the *type* attribute SHOULD be indicated as a URI that points to the definition of the type, such as https://schema.org/Place or http://www.somewhere.com/taxonomy#theType. The use of URIs for the *Object type* attribute is explained in further detail in Chapter 4.

Other attributes for *Object* MAY be included but their use is not specified.

Every *Object* SHALL have at least one sub-element called *id* that identifies the *Object*. An Object MAY have additional *id* elements. Such an identifier is known as an *object identifier*. An *object identifier* SHOULD be globally unique or at least unique for the specific application, domain, or network of the organizations involved.

An optional *description* sub-element MAY be included for providing a description of the *Object*, usually intended for human users.

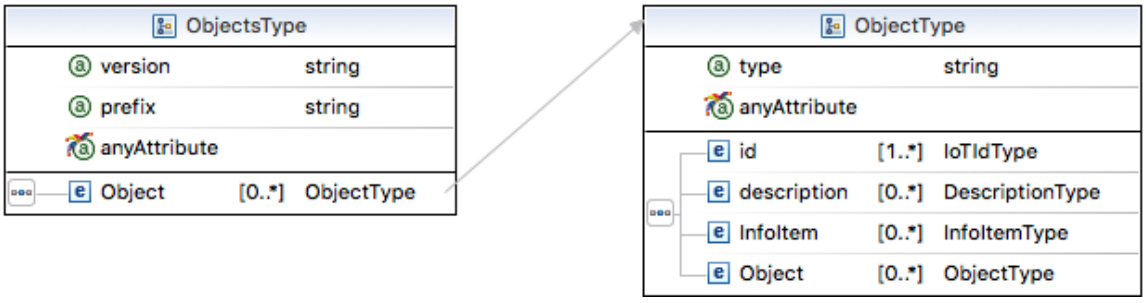

Figure 2: ObjectsType and ObjectType Element

The *Object*’s properties are included with an arbitrary number of *InfoItem* elements. *Object* elements MAY also include an arbitrary number of *Object* sub-elements (see Figure 2).

3.3 Object Identifiers (*id*)

Object identifiers are specified using the *IoTIdType* type (Figure 3). The attributes of *IoTIdType* MAY express what standard or coding scheme the *id*/*name* uses, on what kind of media it is written (e.g., RFID tag, barcode, stamped, etc.), and the beginning and end of validity of the identifier. Other information CAN be conveyed using attributes that are not defined in this standard.

Especially for *Objects*, the possibility to use more than one *id* is a common real-life requirement because the same "Thing" can often carry several different identifiers. For instance, a postal package may end up with several company-specific tracking numbers, a Serial Shipping Container Code (SSCC), and other identifiers before reaching its destination. Similarly, a vehicle may need to be identified by its Vehicle Identification Number (VIN) in one context, whereas it needs to be identified by its license number in another context.

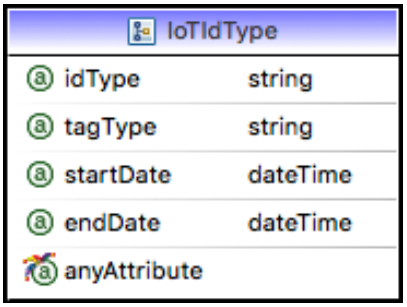

Figure 3: IoTIdType Element

Other attributes for *IoTIdType* elements MAY be included but their use is not specified.

3.4 The *description* Element

The *description* element is intended for being read and understood by humans. Experience has shown the utility of including a simple-to-use "free-form" text element for user interface and debugging purposes. A *description* element MAY have a *lang* attribute that specifies the language of the description, where the *lang* value SHALL be one of the language codes specified in the ISO 639 standard. There MAY be an arbitrary number of *description* elements.

3.5 The *InfoItem* Element

The reason for calling properties "*InfoItem*", rather than using the name "*Property*" (or some similar concept that is familiar from object-oriented programming), is that in the IoT an *InfoItem* can also be an event of some kind, a URL that points to an external entity, a method that can be called, etc., rather than just a simple value.

The *name* attribute SHALL be used for defining the name of the *InfoItem*, which MUST be unique for the *Object* in question. Additional names – that is, synonyms or alternative names in different systems – MAY be provided as *altname* sub-elements.

The *type* attribute of *InfoItem* SHOULD be used for indicating what kind of value(s) the *InfoItem* represents. As for *Object*, if possible, the value of the *type* attribute SHOULD be indicated as a URI that points to the definition of the type, such as https://schema.org/latitude or http://www.somewhere.com/taxonomy#theType.

Other attributes for *InfoItem* MAY be included but their use is not specified.

InfoItem elements MAY contain sub-elements:

- *altname*: additional names for the same *InfoItem*

This feature may be used, for instance, if the same *InfoItem* is known under different names in different organizations or software.

- *description*: text that explains what the *InfoItem* represents, mainly intended for human users in the same way as for *Object*
- *MetaData*: provides metadata information about the *InfoItem*, such as units, latency, and other similar information that is not specified by the value of the *type* attribute of *InfoItem* or that is provided for other reasons
- *value*: arbitrary number of values for the *InfoItem* (Figure 4)

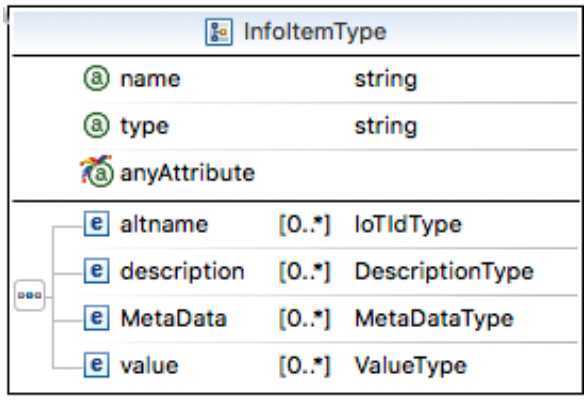

Figure 4: InfoItemType Element

Even though it is possible to include *description*, *MetaData*, and *value* element(s) simultaneously for an *InfoItem*, it is usually not practical to do so. *MetaData* and *description* information typically needs to be exchanged only once for knowing what the values signify. Values are exchanged whenever they are updated or unknown for some other reason.

MetaData and *description* elements MAY also be empty (<*MetaData*/>, <*description*/>), which is used, for instance, with the O-MI standard to query for those elements.

InfoItems MAY be used for expressing information that could be considered *MetaData* of the *Object*, such as the web pages of an organization, a link to user instructions of a machine, etc., whose values might change rarely or never. What properties are considered to be *MetaData* tend to be domain, application, and organization-specific.

3.6 The *MetaData* Element

MetaData specifies what the *value*(s) of an *InfoItem* represent and how to use them. Therefore, *MetaData* is not typically meant to be modified over time, even though practice has shown that some *MetaData* such as "last accessed" and similar can be useful in certain applications.

In most applications, *MetaData* is **sent** only when publishing a new *Object* or *InfoItem* to a new system. *MetaData* is typically **requested** only once when encountering a previously unknown *Object* or *InfoItem*. The *MetaData* element MAY contain an arbitrary number of *InfoItem* elements. *MetaData* sub-elements are of type *InfoItem* because they are syntactically similar to *Object InfoItem* sub-elements, even though the **intended use of** *MetaData InfoItems* is different from *Object InfoItems*.

There SHOULD be only one *MetaData* element per *InfoItem* for practical reasons, even though there MAY be several *MetaData* elements according to the O-DF schema.

3.7 The *value* Element

The signification of a *value* SHOULD be expressed by the *type* attribute and *MetaData* of the corresponding *InfoItem*. The *type* attribute of *ValueType* is for expressing the specific type of the actual value, such as *xs:integer, xs:anyURI*, or similar. If the type of *value* is clear from the *type* attribute of the *InfoItem* and *MetaData*, then the *type* attribute of *ValueType* SHOULD NOT be used because it adds more payload and requires extra processing.

The attributes *dateTime* and *unixTime* are alternative and complementary ways of indicating the time when the *value* was last updated, when an event occurred, or whatever is appropriate for the *InfoItem* in question. How these timestamps are used is application-specific. However, it is recommended to provide timestamps for all *value* updates. In practice, most implementations will create and associate a timestamp with all *value* updates if a timestamp is not provided. If both *dateTime* and *unixTime* are provided, then they SHOULD indicate the same moment in time in order to avoid ambiguity.

Other attributes for *value* elements MAY be included but their use is not specified.

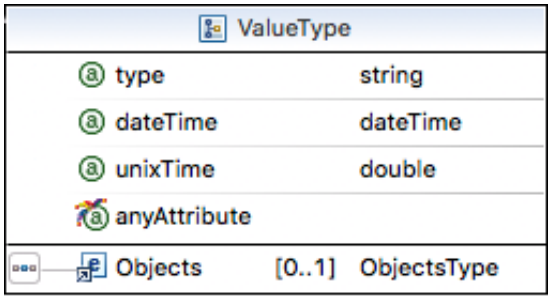

Figure 5: ValueType Element

The actual value of a *value* element can be anything allowed by the O-DF XML Schema. The most typical case is that it is a numeric value for sensor readings or similar, a text string that indicates a new state of the object, or similar.

It is also possible to use a Comma-Separated Value (CSV) or similar strings for sending many *value* updates with one single *value* element. In that case, the signification of the CSV columns SHOULD be indicated by the *MetaData* of that *InfoItem*. The *MetaData* element SHOULD then have an *InfoItem* with *name="labels"* and *type="CSV"*. The actual labels SHALL be indicated as the *value* of that *MetaData InfoItem*; i.e., using CSV in this case. The same principle SHOULD be used for other encodings, such as Tab-Separated Values (TSV), etc.

If the value is encoded using the Base64 algorithm described in IETF RFC 2045, then the *value* element should have *type=xs:base64binary*. The same principle applies to other encodings, such as *xs:hexBinary*.

A *value* element MAY have another O-DF structure as its sub-element. This possibility is used, for instance, with the *call* operation of the O-MI standard.

4 Use of the *type* Attribute

The O-DF notation is primarily meant for keeping track of events and *value* updates related to unique instances of some kind in an IoT and lifecycle management context. It is not intended to be used as a generic information modeling tool. Rather, O-DF *Object* hierarchies, *InfoItems*, and identifiers SHOULD use existing standards and well-known information models whenever possible. Examples of such information models are Schema.org and the O-DEF standard.

The information model used is expressed through the *type* attribute. The value of the *type* attribute MAY be absolute or relative in the same way as URLs as specified in IETF RFC 3986. As for URLs, an *absolute type value* SHALL be indicated by the presence of a scheme at the beginning of the value; i.e., a string that ends with the scheme component delimiter (":").

A *relative type value* SHALL NOT include a scheme. A relative *type* SHALL automatically inherit any scheme specified by the *type* attribute of the parent element. Other parts of the parent element *type* MAY be inherited depending on the scheme.

For clarity, the *idType* attribute of <*id*> elements SHALL NOT inherit the scheme or other parts of the *type* attribute. The reason for this is that object identifiers are often specified by different standards and standardization bodies than object types.

4.1 The *type* Attribute using Common Ontologies

Many taxonomies and ontologies exist for different domains, which may be standards or not. For instance, indicating *type="https://schema.org/Place"* for an *Object* signifies that the *Object* in question SHALL have the *InfoItems* (properties) and sub-objects that correspond to the specification found at https://schema.org/Place. In this case, the URI scheme is https, so relative types MAY be used for sub-elements.

The *prefix* attribute is another way of shortening type specifications, as shown in Figure 6. Schema.org specifies that there can be a property called *address*, which can be of *type schema.org/PostalAddress* or *schema.org/Text*. *schema.org/Text* has been used for simplicity in this example.

```
<Object type="schema:Place" prefix="schema: http://www.schema.org/">
  <id>Aalto_CS_ParkingPlace</id>
    <InfoItem name="address" type="schema:Text" >
      <value unixTime="1527758862" dateTime="2018-05-31T12:27:42.977+03:00">
        Konemiehentie 2, 02150 Espoo, Finland
      </value>
    </InfoItem>
</Object>
```

Figure 6: Use of the Prefix Attribute for the *type* Specification

4.2 The *type* Attribute using the O-DEF Standard

The Open Data Element Framework (O-DEF) is a standard of The Open Group that provides unique identifiers for *Object* types, properties, units, etc. It also supports a concept of *plug-ins* that can be used for referring to definitions in other standards, such as the UNSPSC and UNECE standards.

An O-DEF type SHALL be indicated by a URI scheme (see IETF RFC 3986). If the type is a numeric O-DEF code, then that scheme SHALL be *odef*. It the type is indicated using one of the O-DEF language mappings, then the scheme SHALL be *odef-lang*, where "*lang*" SHALL be one of the language codes specified in the ISO 639 standard. As an example, *odef-en* indicates that the type definition is using the English language mapping of the O-DEF standard, as shown in Figure 8.

A numeric O-DEF code SHOULD be used as the type whenever possible. If, for some reason, it makes sense to indicate the type using different O-DEF language mappings, then the *Object* SHALL have an *InfoItem* per language mapping with a *name* attribute that corresponds to the scheme of that language mapping, as in Figure 8.

When the O-DEF standard is used to indicate the *type* of an *Object*, that *type* SHALL be an O-DEF object class or an O-DEF object class plus an O-DEF role.

InfoItem types SHALL be indicated using the *type* attribute, as shown in the example in Figure 7. When the O-DEF standard is used to indicate the *type* of an *InfoItem*, the *type* information provided SHALL be an O-DEF property. The *type* of the *InfoItem* SHALL then be the O-DEF data element concept identified by the O-DEF object class of the enclosing object and the O-DEF property.

When the type of an O-DF element is using the *odef* scheme, then *odef* SHALL be the only scheme allowed until the end of that element. There SHALL NOT be any other scheme for any sub-elements of that element.

For an element whose type uses the *odef* scheme, all sub-elements SHALL inherit the entire O-DEF type of the parent element. In the example in Figure 7, the O-DEF type of the *Object* is *odef:10;4:[40101709]*. Since the *Object* O-DEF type is inherited by the *Outside InfoItem*, the absolute *type* of the *FreshAir InfoItem* is *odef:10;4:[40101709]_5;6:[CEL]*. The O-DEF standard provides a plug-in concept that makes it possible to use existing standards as such. In these examples both the UNSPSC and UNECE Rec 20 plug-ins are used.

```
<Objects xmlns="http://www.opengroup.org/xsd/odf/2.0/" version="2.0">
  <Object type="odef:10;4:[40101709]">
    <id idType="id@uri">2013001000@enervent.fi</id>
    <InfoItem name="OutsideAirTemperature" type="_5;6:[CEL]">
      <value unixTime="1518169710">10.5</value>
    </InfoItem>
  </Object>
</Objects>
```

Figure 7: Example of a Partial O-DF Description of an Air Handling Unit Object using O-DEF *types*

(Note that this includes one *InfoItem* with the corresponding O-DEF *type* and a *value* element.)

```
<Objects xmlns="http://www.opengroup.org/xsd/odf/2.0/" version="2.0">
  <Object type="odef:10;4:[40101709]">
    <id idType="id@uri">2013001000@enervent.fi</id>
    <description lang="EN">Enervent Air Handling Unit</description>
    <InfoItem name="odef-en">
      <value>
        Product;UNSPSCv18:[Air handling unit]
      </value>
    </InfoItem>
    <InfoItem name="OutsideAirTemperature" type="_5;6:[CEL] ">
      <description lang="EN">"TE01 (outside air) temperature."</description>
      <MetaData>
        <InfoItem name="odef-en">
          <value>
            _Measure;UNECE-Rec20:[Heat.Temperature.Degree-Celsius]
          </value>
        </InfoItem>
      </MetaData>
      <value unixTime="1518169710">10.5</value>
    </InfoItem>
  </Object>
</Objects>
```

Figure 8: Example of Figure 7 Extended with O-DEF English Language Types

5 Use of the O-DF Structures with HTTP GET

Due to the hierarchical nature of O-DF structures, they CAN be used for performing URL-based information discovery and queries through the HTTP GET operation. An example of how this can be done using the UNIX "wget" utility is shown in Example 2, with a corresponding example response in Example 3. The response SHALL contain all compulsory attributes and sub-elements. It SHOULD include all other attributes. It MAY contain other sub-elements.

Example 2: Issuing an HTTP GET Request

```
wget http://dialog.hut.fi/omi/Objects/
```

Example 2 illustrates issuing an HTTP GET request to the URL "http://dialog.hut.fi/omi/"[1] for querying the available information about the data source *Objects*.

Example 3: Response using the O-DF Structure

```
<Objects>
    <Object>
        <id>Refrigerator123</id>
    </Object>
    <Object>
        <id>HeatingController321</id>
    </Object>
    <Object>
        <id>WeatherStation651</id>
    </Object>
</Objects>
```

Example 3 shows an example response to request an *Objects* list, within the "Smart Home" domain. In this case, no type or other attributes have been stored, otherwise they should be included in the response.

The elements of the retrieved O-DF structure in Example 3 can be used for drilling further down into the *Object* hierarchy where, for instance, the following URL:

```
http://dialog.hut.fi/omi/Objects/Refrigerator123/
```

would return the list of *InfoItem* sub-elements and possible sub-objects of the *Object* "*Refrigerator123*" as shown in Example 4.

The relative URL to use for *Object* elements SHALL BE, in order:

1. An <*id*> element with *idType="odfurl"*; for example:
   <id idType="odfurl">Refrigerator123</id>.
2. If no <*id*> element with *idType="odfurl"* is present, then the value of the first <*id*> element SHALL be used.

[1] The URL "http://dialog.hut.fi/omi/" is provided as an example of a valid URL of an O-MI node. The reader should not assume that a valid O-MI node would be continuously available at that address, nor that it would return the content shown in this standard.

The relative URL to use for *InfoItem* elements SHALL BE, in order:

1. A <*altname*> element with *idType="odfurl"*; for example: <*altname idType="odfurl"*>*PowerConsumption*</*altname*>.

2. If no <*altname*> element with *idType="odfurl"* is present, then the *value* of the *name* attribute SHALL be used.

If HTTP GET support is to be implemented, then the appropriate mechanism and relative URLs MUST be specified so that they can be used as a part of a valid URL.

Example 4: Result for a URL-Based Data Discovery Request using O-DF Semantics

(Note that the empty description element indicates the existence of a description that can be queried separately.)

```
<Object>
    <id>Refrigerator123</id>
    <description/>
    <InfoItem name="PowerConsumption"/>
    <InfoItem name="RefrigeratorDoorOpenWarning"/>
    <InfoItem name="RefrigeratorProbeFault"/>
</Object>
```

Further drilling down in the O-DF structure (Example 2) can be done in similar ways, as for example:

- `<URL>/Objects/Refrigerator123/id/` returns all the *id*s of *Refrigerator123*
- `<URL>/Objects/Refrigerator123/PowerConsumption/` returns the current power consumption *value* structure
- `<URL>/Objects/Refrigerator123/PowerConsumption/value/` returns the "raw" power consumption *value*
- `<URL>/Objects/Refrigerator123/PowerConsumption/description/` returns the *PowerConsumption* element including all its *description* sub-elements
- `<URL>/Objects/Refrigerator123/PowerConsumption/MetaData/` returns the *PowerConsumption* element including all *MetaData* sub-elements
- `<URL>/Objects/Refrigerator123/PowerConsumption/altname/` returns the *PowerConsumption* element including all its *altname* sub-elements

More complex queries such as querying for *values* of several *Objects* and *InfoItems* in one go, or querying for historical *values*, requires the use of other kinds of querying mechanisms, such as those specified in the O-MI standard.

If the client wants to receive a specific version of the O-DF data, it SHALL use an HTTP GET parameter called "*version*". The value of the parameter SHALL have at least the major version number (such as "2", "1", ...) and MAY include a minor version number (such as "2.0", "2.1", ...). The receiving server SHALL provide the O-DF data using the requested version, if it is supported by the server. If the requested version is not available, then an earlier version that is as close as possible to the requested version SHALL be used.

6 Inheritance Mechanism for Domain-Specific Data Models

At the time of publication, a domain-specific data model extension exists only for product and product lifecycle-related information. That data model is specified in a separate XML Schema file. Such domain-specific schema SHALL include the O-DF schema by the following line:

```
"<xs:include namespace=http://www.opengroup.org/xsd/odf/2.0/
schemaLocation="odf.xsd"/>"
```

The following lines define that a new *type* called *odfPhysicalProduct* is an extension of *Object* and can be used in the same way as *Object*:

```
<xs:element name="odfPhysicalProduct" type="PhysicalProduct"
substitutionGroup="Object"/>
<xs:complexType name="PhysicalProduct">
    <xs:complexContent>
    <xs:extension base="ObjectType">
```

This signifies that *odfPhysicalProduct* elements can be used interchangeably with *Object* elements and that they inherit all the attributes and sub-elements of *Object*. Other attributes and sub-elements can then be defined for the *PhysicalProduct* type, which are particular for that domain. Similar extensions can be created for all other data types defined in the root data model schema.

The same extension mechanism SHALL be used for all other O-DF-compliant specifications.

For the time being, the only known use of this mechanism is the Open Lifecycle Management (O-LM) work performed in the IoT Work Group of The Open Group. However, the O-LM work is not currently being pursued. The main reason is that it is possible to link to external taxonomies by using the *type* attribute of *Object* and *InfoItem*. However, this extension mechanism may turn out to be useful in the future.

A O-DF XSD Schema (Normative)

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- edited by Kary Framling (Aalto University) -->
<!-- Root Data Model Schema -->
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns="http://www.opengroup.org/xsd/odf/2.0/"
targetNamespace="http://www.opengroup.org/xsd/odf/2.0/" elementFormDefault="qualified"
attributeFormDefault="unqualified" version="2.0">
  <xs:element name="Objects" type="ObjectsType">
    <xs:annotation>
      <xs:documentation>Data Model Root Element</xs:documentation>
    </xs:annotation>
  </xs:element>
  <xs:complexType name="ObjectsType">
    <xs:sequence>
      <xs:element name="Object" type="ObjectType" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="version" type="xs:string" use="optional">
      <xs:annotation>
        <xs:documentation>Schema version used.</xs:documentation>
      </xs:annotation>
    </xs:attribute>
    <xs:attribute name="prefix" type="xs:string" use="optional">
      <xs:annotation>
        <xs:documentation>Prefix definitions according to RDFa convention.</xs:documentation>
      </xs:annotation>
    </xs:attribute>
    <xs:anyAttribute processContents="lax">
      <xs:annotation>
        <xs:documentation>Proprietary or extended attributes.</xs:documentation>
      </xs:annotation>
    </xs:anyAttribute>
  </xs:complexType>
  <xs:complexType name="ObjectType">
    <xs:sequence>
      <xs:element name="id" type="IoTIdType" minOccurs="1" maxOccurs="unbounded"/>
      <xs:element name="description" type="DescriptionType" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="InfoItem" type="InfoItemType" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="Object" type="ObjectType" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="type" type="xs:string" use="optional"/>
    <xs:anyAttribute processContents="lax">
      <xs:annotation>
        <xs:documentation>Proprietary or extended attributes.</xs:documentation>
      </xs:annotation>
    </xs:anyAttribute>
  </xs:complexType>
  <xs:complexType name="InfoItemType">
    <xs:sequence>
      <xs:element name="altname" type="IoTIdType" minOccurs="0" maxOccurs="unbounded">
        <xs:annotation>
          <xs:documentation>Optional list of other names for the same InfoItem.</xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="description" type="DescriptionType" minOccurs="0" maxOccurs="unbounded"/>
      <xs:element name="MetaData" type="MetaDataType" minOccurs="0" maxOccurs="unbounded">
```

```
        <xs:annotation>
          <xs:documentation>Meta-data about the InfoItem, such as "latency", "unit"
etc.</xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="value" type="ValueType" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="name" type="xs:string" use="required">
      <xs:annotation>
        <xs:documentation>Name of InfoItem, such as "PowerConsumption", "Diameter" or
similar.</xs:documentation>
      </xs:annotation>
    </xs:attribute>
    <xs:attribute name="type" type="xs:string" use="optional">
      <xs:annotation>
        <xs:documentation>Type definition for the InfoItem, such as "http://schema.org/latitude".
From Version 2.0 onwards.</xs:documentation>
      </xs:annotation>
    </xs:attribute>
    <xs:anyAttribute processContents="lax">
      <xs:annotation>
        <xs:documentation>Proprietary or extended attributes.</xs:documentation>
      </xs:annotation>
    </xs:anyAttribute>
  </xs:complexType>
  <xs:complexType name="MetaDataType">
    <xs:sequence>
      <xs:element name="InfoItem" type="InfoItemType" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="DescriptionType">
    <xs:annotation>
      <xs:documentation>String with some"human-readable" text.</xs:documentation>
    </xs:annotation>
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute name="lang" type="xs:string" use="optional">
          <xs:annotation>
            <xs:documentation>Language of "description" text.</xs:documentation>
          </xs:annotation>
        </xs:attribute>
        <xs:anyAttribute processContents="lax">
          <xs:annotation>
            <xs:documentation>Proprietary or extended attributes.</xs:documentation>
          </xs:annotation>
        </xs:anyAttribute>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
  <xs:complexType name="IoTIdType">
    <xs:simpleContent>
      <xs:extension base="xs:string">
        <xs:attribute name="idType" type="xs:string" use="optional">
          <xs:annotation>
            <xs:documentation>Text identifying the ID schema.</xs:documentation>
          </xs:annotation>
        </xs:attribute>
        <xs:attribute name="tagType" type="xs:string" use="optional">
          <xs:annotation>
            <xs:documentation>Text identifying the ID Tag media type. </xs:documentation>
          </xs:annotation>
        </xs:attribute>
        <xs:attribute name="startDate" type="xs:dateTime" use="optional">
```

```
          <xs:annotation>
            <xs:documentation>Start of validity for the ID</xs:documentation>
          </xs:annotation>
        </xs:attribute>
        <xs:attribute name="endDate" type="xs:dateTime" use="optional">
          <xs:annotation>
            <xs:documentation>End of validity for the ID</xs:documentation>
          </xs:annotation>
        </xs:attribute>
        <xs:anyAttribute processContents="lax">
          <xs:annotation>
            <xs:documentation>Proprietary attributes.</xs:documentation>
          </xs:annotation>
        </xs:anyAttribute>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
  <xs:complexType name="ValueType" mixed="true">
    <xs:sequence>
      <xs:element ref="Objects" minOccurs="0" maxOccurs="1"/>
    </xs:sequence>
    <xs:attribute name="type" type="xs:string" use="optional" default="xs:string"/>
    <xs:attribute name="dateTime" type="xs:dateTime" use="optional"/>
    <xs:attribute name="unixTime" type="xs:double" use="optional"/>
    <xs:anyAttribute processContents="lax"/>
  </xs:complexType>
</xs:schema>
```

B Example O-DF Structures

This appendix shows example messages for some basic cases. More examples are normally available at the website(s) where this specification is published.

All examples have been validated against the XML Schema using the UNIX `xmllint` utility. The command that was executed for validation was:

```
xmllint --noout --schema odf.xsd <XML file>
```

Example 5: Object->Object->InfoItem->Value(s)

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Example of Object->Object->InfoItem->value(s).  -->
<Objects xmlns="http://www.opengroup.org/xsd/odf/2.0/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengroup.org/xsd/odf/2.0/ odf.xsd">
  <Object type="someType">
    <id>UniqueTargetID_1</id>
    <InfoItem name="InfoItem1">
      <value>Value1</value>
      <value>Value2</value>
      <value>Value3</value>
    </InfoItem>
    <InfoItem name="InfoItem2">
      <value>Value</value>
    </InfoItem>
    <Object type="someType">
      <id>SubTarget1</id>
      <InfoItem name="SubInfoItem1"/>
      <Object type="someType">
        <id>SubSubTarget1</id>
        <InfoItem name="SubSubTarget1InfoItem1">
          <value>22.5</value>
        </InfoItem>
      </Object>
    </Object>
    <Object type="someType">
      <id>SubTarget2</id>
      <InfoItem name="SubTarget2InfoItem1">
        <value>34.6</value>
      </InfoItem>
    </Object>
  </Object>
</Objects>
```

Example 6: MetaData about Air Handling Unit Outside Temperature InfoItem

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- MetaData about Air Handling Unit outside temperature InfoItem -->
<Objects xmlns="http://www.opengroup.org/xsd/odf/2.0/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengroup.org/xsd/odf/2.0/ odf.xsd">
  <Object type="odef:3;4:[40101709]">
    <id>2013001000@enervent.fi</id>
    <InfoItem name="OutsideAirTemperature" type="_5;6:[CEL]">
      <MetaData>
        <InfoItem name="format"><value
type="xs:string">xs:double</value></InfoItem>
        <InfoItem name="latency"><value type="xs:int">10</value></InfoItem>
        <InfoItem name="readable"><value
type="xs:boolean">true</value></InfoItem>
        <InfoItem name="writable"><value
type="xs:boolean">false</value></InfoItem>
        <InfoItem name="unit"><value
type="xs:string">Celsius</value></InfoItem>
        <InfoItem name="accuracy"><value
type="xs:double">0.1</value></InfoItem>
      </MetaData>
    </InfoItem>
  </Object>
</Objects>
```

Example 7: Measurement Values for Air Handling Unit Outside Temperature

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- Example of a simple O-DF structure for outside air temperature of AHU. -->
<Objects xmlns="http://www.opengroup.org/xsd/odf/2.0/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengroup.org/xsd/odf/2.0/ odf.xsd">
  <Object>
    <id idType="id@uri">2013001000@enervent.fi</id>
    <InfoItem name="OutsideAirTemperature">
    <value dateTime="2018-10-26T15:33:21">18.5</value>
    <value dateTime="2018-10-26T16:33:50">17.7</value>
    <value dateTime="2018-10-26T17:34:15">16.5</value>
    <value dateTime="2018-10-26T18:34:35">15.4</value>
    <value dateTime="2018-10-26T19:34:52">14.3</value>
    </InfoItem>
  </Object>
</Objects>
```

Example 8: O-DF Structure Based on Schema.org Place (no values included in this example)

```
<?xml version="1.0" encoding="UTF-8"?>
<!-- This is a representation of schema.org/Place object in O-DF. -->
<Objects xmlns="http://www.opengroup.org/xsd/odf/2.0/"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
xsi:schemaLocation="http://www.opengroup.org/xsd/odf/2.0/ odf.xsd">
  <Object type="schema:Place" prefix="schema: https://schema.org/">
    <id>SomePlace</id>
    <!-- InfoItems/Objects of Place can be included as/if needed. -->
    <Object type="schema:GeoCoordinates">
      <id>geo</id>
      <InfoItem name="latitude" type="schema:latitude"/>
      <InfoItem name="longitude" type="schema:longitude"/>
      <InfoItem name="elevation" type="schema:elevation"/>
      <InfoItem name="postalCode" type="schema:postalCode"/>
      <Object type="schema:PostalAddress">
        <id>address</id>
        <InfoItem name="streetAddress" type="schema:streetAddress"/>
        <InfoItem name="postalCode" type="schema:postalCode"/>
        <InfoItem name="postOfficeBoxNumber"
type="schema:postOfficeBoxNumber"/>
        <InfoItem name="addressRegion" type="schema:addressRegion"/>
        <InfoItem name="addressLocality" type="schema:addressLocality"/>
        <InfoItem name="addressCountry" type="schema:addressCountry"/>
      </Object>
    </Object>
  </Object>
</Objects>
```

C JSON Mapping and Examples

Many XML generation and manipulation tools can convert XML messages into JavaScript Object Notation (JSON), and *vice versa*. There are also many frameworks that can do the needed conversions in "real time". The following code shows Example 7 converted into JSON by an online tool (in this case at URL http://www.utilities-online.info/xmltojson/#.XBuT6y2B1TY).

```
{
  "Objects": {
    "-xmlns": "http://www.opengroup.org/xsd/odf/2.0/",
    "-xmlns:xsi": "http://www.w3.org/2001/XMLSchema-instance",
    "-xsi:schemaLocation": "http://www.opengroup.org/xsd/odf/2.0/ odf.xsd",
    "Object": {
      "id": {
        "-idType": "id@uri",
        "#text": "2013001000@enervent.fi"
      },
      "InfoItem": {
        "-name": "OutsideAirTemperature",
        "value": [
          {
            "-dateTime": "2018-10-26T15:33:21",
            "#text": "18.5"
          },
          {
            "-dateTime": "2018-10-26T16:33:50",
            "#text": "17.7"
          },
          {
            "-dateTime": "2018-10-26T17:34:15",
            "#text": "16.5"
          },
          {
            "-dateTime": "2018-10-26T18:34:35",
            "#text": "15.4"
          },
          {
            "-dateTime": "2018-10-26T19:34:52",
            "#text": "14.3"
          }
        ]
      }
    }
  }
}
```

Another conversion tool (https://codebeautify.org/xmltojson) produces the following:

```
{
  "Objects": {
    "Object": {
      "id": {
        "_idType": "id@uri",
        "__text": "2013001000@enervent.fi"
      },
      "InfoItem": {
        "value": [
```

```
            {
              "_dateTime": "2018-10-26T15:33:21",
              "__text": "18.5"
            },
            {
              "_dateTime": "2018-10-26T16:33:50",
              "__text": "17.7"
            },
            {
              "_dateTime": "2018-10-26T17:34:15",
              "__text": "16.5"
            },
            {
              "_dateTime": "2018-10-26T18:34:35",
              "__text": "15.4"
            },
            {
              "_dateTime": "2018-10-26T19:34:52",
              "__text": "14.3"
            }
          ],
          "_name": "OutsideAirTemperature"
        }
      },
      "_xmlns": "http://www.opengroup.org/xsd/odf/2.0/",
      "_xmlns:xsi": "http://www.w3.org/2001/XMLSchema-instance",
      "_xsi:schemaLocation": "http://www.opengroup.org/xsd/odf/2.0/ odf.xsd"
    }
}
```


These two conversions are quite similar but also differ in several ways. In general, an O-DF structure that is converted from XML to JSON and then converted back to XML will not be identical to the original O-DF structure. This is because JSON does not make a difference between “attribute” and “sub-element”, so that information is lost. Other details are also handled in different ways by different conversion tools.

The need for a JSON specification of the O-DF standard was acknowledged by the IoT Work Group before Version 1.0 was published in 2014. However, that specification still remains to be done.

Abbreviations & Acronyms

CSV	Comma-Separated Value
HTTP	Hypertext Transfer Protocol
IoT	Internet of Things
JAXB	Java Architecture for XML Binding
JMS	Java Message Service
JSON	JavaScript Object Notation
O-DEF	Open Data Element Framework
O-DF	Open Data Format
O-LM	Open Lifecycle Management
O-MI	Open Messaging Interface
QLM	Quantum Lifecycle Management
REST	REpresentational State Transfer
RFID	Radio-Frequency Identification
SOAP	Simple Object Access Protocol
SoS	System-of-Systems
SSCC	Serial Shipping Container Code
TSV	Tab-Separated Values
TTL	Time-To-Live
UNECE	United Nations Economic Commission for Europe
UNSPSC	United Nations Standard Products and Services Code
URI	Uniform Resource Identifier
URL	Uniform Resource Locator
UTC	Coordinated Universal Time
VIN	Vehicle Identification Number
XML	EXtensible Markup Language

Index