भारतीय मानक

IS/ISO/IEC 21823-4 : 2022

Indian Standard

इंटरनेट ऑफ थिंगस (आई ओ टी) — आई ओ टी सिस्टम के लिए इंटरऑपरेबिलिटी

भाग 4 वाक्यात्मक इंटरऑपरेबिलिटी

Internet of Things (IoT) — Interoperability for IoT Systems

Part 4 Syntactic Interoperability

ICS 35.020

भारतीय मानक ब्यूरो
BUREAU OF INDIAN STANDARDS
मानक भवन, 9 बहादुर शाह ज़फर मार्ग, नई दिल्ली - 110002
MANAK BHAVAN, 9 BAHADUR SHAH ZAFAR MARG
NEW DELHI - 110002
www.bis.gov.in www.standardsbis.in

July 2024 **Price Group 13**

Internet of Things and Digital Twin Sectional Committee, LITD 27

NATIONAL FOREWORD

This Indian Standard (Part 4) which is identical to ISO/IEC 21823-4 : 2022 ‘Internet of things (IoT) Interoperability for IoT systems — Part 4: Syntactic interoperability’ issued by the International Organization for Standardization (ISO) and International Electrotechnical Commission (IEC) jointly was adopted by the Bureau of Indian Standards (BIS) on the recommendation of the Internet of Things and Digital Twin Sectional Committee and approval of the Electronics and Information Technology Division Council.

This standard (Part 4) is one of the parts of a series of standards on ‘Internet of Things (IoT) Interoperability for IoT Systems’. The other parts in this series are:

Part 1 Framework

Part 2 Transport interoperability

Part 3 Semantic interoperability

The text of ISO/IEC standard has been approved as suitable for publication as an Indian Standard without deviations. Certain conventions are, however, not identical to those used in Indian Standards. Attention is particularly drawn to the following:

a) Wherever the words ‘International Standard’ appears referring to this standard, they should be read as ‘Indian Standard’; and

b) Comma (,) has been used as a decimal marker while in Indian Standards, the current practice is to use a point (.) as the decimal marker.

The Committee has reviewed the provisions of the following International Standards referred in this adopted standard and has decided that they are acceptable for use in conjunction with this standard. For undated references, the latest edition of the referenced document applies, including any corrigenda and amendment. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies:

International Standard	*Title*
ISO/IEC 20924	Internet of Things (IoT) — Vocabulary

For the purpose of deciding whether a particular requirement of this standard is complied with, the final value, observed or calculated, expressing the result of a test or analysis, shall be rounded off in accordance with IS 2 : 2022 ‘Rules for rounding off numerical values (*second revision*)’. The number of significant places retained in the rounded off value should be same as that of the specified value in this standard.

CONTENTS

INTRODUCTION

In the world of the Internet of Things (IoT), heterogeneous systems and devices need to be connected and exchange data with others. How data exchange can be implemented becomes a key issue of interoperability among IoT industries. Information models (IMs), which can well represent specifications of data, are adopted and utilized to solve the interoperability problem. Meanwhile, as systems and devices in IoT can have different information models with different modelling methodologies and formats, interoperability based on different information models is recognized as an urgent problem. The IoT interoperability related systems and applications have an 11 trillion market potentially [1][1].

The ISO/IEC 21823 series standards address issues that relate to interoperability both between different IoT systems and within a single IoT system. ISO/IEC 21823-1 [2] describes a general framework for interoperability for IoT systems. It includes a five facet model for interoperability that includes transport, syntactic, semantic, behavioural, and policy viewpoints.

Different parts of ISO/IEC 21823, based on one of the facets, provide specifications from their corresponding viewpoints. Each of the parts can refer to others but is independent. Currently, ISO/IEC 21823-2 [3] defines specifications from the transport viewpoint, ISO/IEC 21823-3 [4] defines requirements, provides guidance, etc. from the semantic viewpoint, and ISO/IEC 21823-4 specifies the syntactic interoperability.

Syntactic interoperability means that exchanged information can be understood by the participating IoT systems which contain IoT devices. In more detail, the syntactic interoperability is related to the information models' representing formats, structures, and grammar of their modelling languages such as a length of a data string, constraints on data types, and forbidden characters.

This document first provides the principle of how to achieve syntactic interoperability based on metamodel-driven approaches. In other words, the reason why the information exchange rules based on metamodels can support syntactic interoperability among different IoT systems will be elaborated. Secondly, requirements on information models such as metamodels and models of IoT systems including IoT devices are described. Features related to IoT devices such as the identifier, device type, setup environments, and functions need to be considered to accomplish syntactic interoperability among different information models utilized in IoT systems. Thirdly, a framework for processes on developing information exchange rules related to IoT devices from the syntactic viewpoint is provided. For example, the kinds of metamodels, and the types of entities and relationships that shall be selected are specified, and the procedure of how to build the information exchange rules from different information models is provided.

In Annex A, possible intrinsic and extrinsic properties of IoT devices are listed as additional information of Clause 6. In Annex B, a use case of how the syntactic interoperability in accordance with specifications in this document among industrial IoT systems and IoT devices is described.

With this document, system and device vendors, who need to improve and/or develop their products to comply with IoT requirements, can implement specifications of this document to their products for an automatic or semi-automatic realization of IoT syntactic interoperability.

[1] Numbers in square brackets refer to the Bibliography.

Indian Standard

INTERNET OF THINGS (IOT) — INTEROPERABILITY FOR IOT SYSTEMS

PART 4 SYNTACTIC INTEROPERABILITY

1 Scope

This part of ISO/IEC 21823 specifies the IoT interoperability from a syntactic point of view. In this document, the following specifications for IoT interoperability from a syntactic viewpoint are included:

– a principle of how to achieve syntactic interoperability among IoT systems which include IoT devices;

– requirements on information related to IoT devices for syntactic interoperability;

– a framework for processes on developing information exchange rules related to IoT devices from the syntactic viewpoint.

2 Normative references

The following documents are referred to in the text in such a way that some or all of their content constitutes requirements of this document. For dated references, only the edition cited applies. For undated references, the latest edition of the referenced document (including any amendments) applies.

ISO/IEC 20924, *Internet of Things (IoT) – Vocabulary*

3 Terms and definitions

For the purposes of this document, the terms and definitions given in ISO/IEC 20924 and the following apply.

ISO and IEC maintain terminological databases for use in standardization at the following web addresses:

- ISO Online browsing platform: available at http://www.iso.org/obp
- IEC Electropedia: available at http://www.electropedia.org/

3.1
instance
individual entity having its own value and possibly its own identity

[SOURCE: ISO 19103:2015 [5], 4.20]

3.2
metamodel
special kind of model that specifies the abstract syntax of a modelling language

Note 1 to entry: A model is an *instance* (3.1) of a metamodel

Note 2 to entry: IoT syntactic interoperability is achieved by information exchange rules through the structure, data format, and syntactic constraints using syntactic aspects of the metamodel.

[SOURCE: ISO/IEC 19506:2012 [6], modified – The description that follows the definition has been deleted. Notes to entry have been added.]

3.3
model
abstraction of some aspects of reality

[SOURCE: ISO 19109:2015 [7], 4.15]

3.4
property
particular characteristic of an object type

[SOURCE: ISO 16484-5:2017 [8], 3.2.74]

3.5
syntactic interoperability
interoperability such that the formats of the exchanged information can be understood by the participating systems

Note 1 to entry: System means IoT system.

Note 2 to entry: IoT device, IoT gateway, sensor and actuator are considered as system.

[SOURCE: ISO/IEC 19941:2017 [9], 3.1.4, modified – Notes to entry have been added.]

4 Abbreviated terms

CRS	coordinate reference system
EPIoT	extrinsic properties of physical IoT devices
IPIoT	intrinsic properties of physical IoT devices
IoT	Internet of Things
JSON	JavaScript Object Notation
MOF	Meta Object Facility
UML	Unified Modelling Language
XML	extensible markup language

5 Principle for IoT syntactic interoperability

5.1 General

In the ISO/IEC 21823 series, ISO/IEC 21823-1 [2] defines an overall framework for IoT interoperability. It specifies that IoT interoperability shall be supported by standards from five facets: transport, semantic, syntactic, behavioural, and policy. A standard based on each of the facets shall provide specifications from its corresponding viewpoint. Each of the standards can refer to or can be independent of standards based on other facets. ISO/IEC 21823-2 [3] defines specifications from the transport viewpoint. ISO/IEC 21823-3 [4] defines requirements, provides guidance, etc. from the semantic aspect. ISO/IEC 21823-4 (this document) addresses the syntactic interoperability that provides specifications from the syntax viewpoint.

5.2 Principle for IoT syntactic interoperability

In this subclause, a principle for IoT syntactic interoperability is specified. In order for an IoT system to achieve syntactic interoperability with other IoT systems and devices, the information exchange rules between their data are adopted.

The information exchange rules for syntactic interoperability provide the following types of information exchange.

a) Format exchange.
   - The term "format" is bound for a data format.
   - The "format exchange" means that information in different data formats can be exchanged.

   For example, data in the UML format can be exchanged with data in the XML format.

b) Structure exchange.
   - The term "structure" is bound for a data structure that has a hierarchy and branches.
   - The "structure exchange" means that information in different structures can be exchanged.

   For example, information defined in a hierarchical tree structure can be transformed into a flat tree structure.

c) Syntactic constraint exchange.
   - The term "constraint" is a condition related to syntax or syntactic requirements on data.
   - The "syntactic constraint exchange" means that information with different constraints can be exchanged.

   For example, data in IoT System1 have a value of one digit after the decimal point, and data in IoT System2 have a value of two digits after the decimal point. Their data accuracy exchange is classified into syntactic constraint exchange.

Furthermore, information of IoT systems is expressed with models. In each IoT system, its information can be represented with a metamodel, models, and instances [10]. In order to describe information exchange rules between IoT systems for their syntactic interoperability, syntactic aspects in their metamodels and models are utilized. In addition, specific requirements for metamodels, models, and information exchanges in the IoT domain are included in this document.

5.3 Relevant technologies for syntactic interoperability

5.3.1 Metamodel and syntactic interoperability

A metamodel, as the model's model, consists of statements about models. Especially in the UML as described in [10], the metamodel specifies the abstract syntax of the UML. The abstract syntax defines the set of UML modelling concepts, attributes, relationships as well as rules for combining concepts to construct partial UML models.

There are also other definitions for metamodel in ISO/IEC and IEEE standards. Some of them are listed in Table C.1 in Annex C. Several metamodel definitions in different resources are collected in ISO/IEC/IEEE 24765:2017 [11]. In this document, Definition 7 of metamodel in Table C.1, i.e. "special kind of model that specifies the abstract syntax of a modelling language", is adopted. From this definition, it is clear that an approach of creating information exchange rules with elements available in metamodels is actually based on the syntax and therefore is acceptable for syntactical interoperability. UML, OWL (Ontology Language), OntoML (Ontology Markup Language [12]), XML, etc. are modelling languages adopted and utilized in different systems and domains.

5.3.2 Metamodel-driven approaches supporting interoperability issues

Metamodel-driven information exchange and interoperability approaches are adopted as holistic approaches in industry domains [13], [14] to enable a model-driven engineering approach in the area of information integration and interoperation. By creating declarative mapping specifications, i.e. exchange rules, automatic information exchange can be executed at run-time and off-line among heterogeneous systems and devices. As the metamodel-driven approaches tackle the interoperability problems at a higher abstraction level than models, it can increase the efficiency of achieving interoperability among heterogeneous systems and devices which comply with the same metamodel. In other words, information exchange rules can be reused by IoT systems and IoT devices whose information models are in compliance with the same metamodel.

5.4 The overall structure of the proposed approach

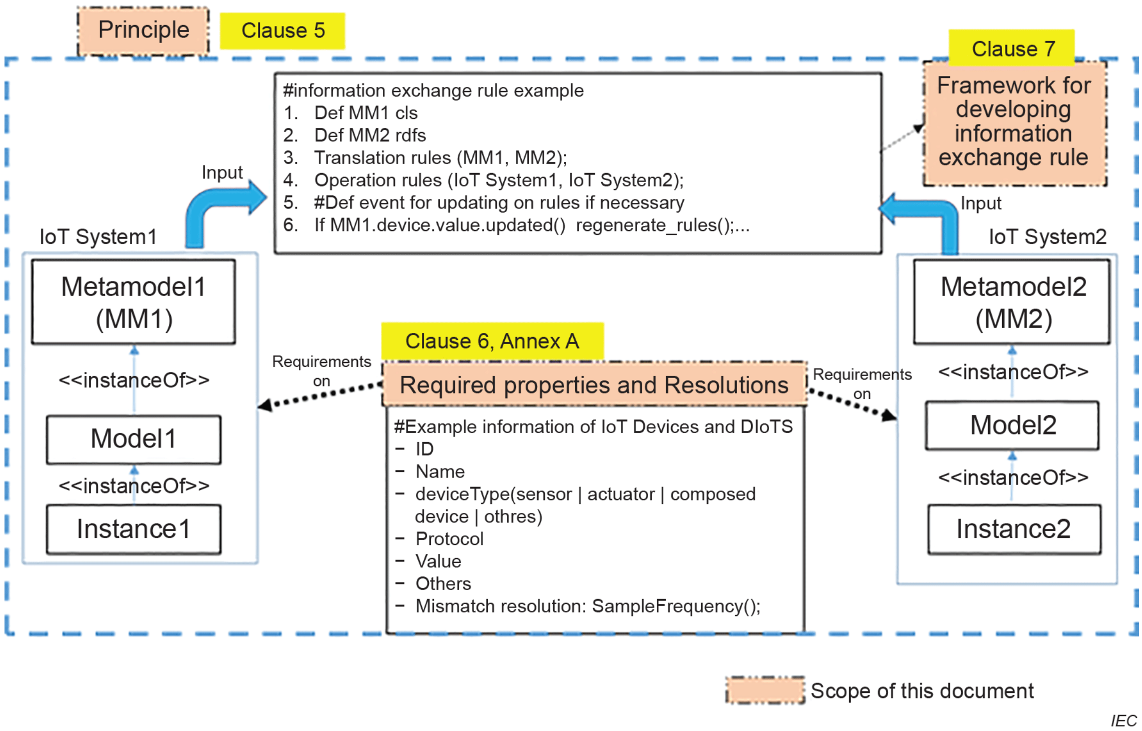


Figure 1 – The overall structure of the proposed approach

Figure 1 illustrates the overall structure of the proposed approach. Figure 1 shows two IoT systems: IoT System1 and IoT System2. In each IoT system, its information consists of a metamodel, model, and instance data. In order to achieve syntactic interoperability between these two systems, the information exchange rules based on the metamodels of both IoT systems need to be created. To create information exchange rules, their required properties and resolutions to support executing information exchange need to be analysed and defined.

In Figure 1:

– lines starting with "#" denote comment lines;

– in the text box of "information exchange rule example", sample information for syntactic interoperability is listed;

– in the text box of "required properties and resolutions", example properties and syntactic resolution for mismatch are listed.

In this document, three major clauses are specified to support achieving IoT syntactic interoperability.

a) In Clause 5, relevant technologies of the metamodel and their applicability in the area of solving syntactical interoperability issues are explained. The methodology of how to create information exchange rules among heterogeneous IoT systems and devices is specified. The information exchange rules are in general categorized into two groups:

   1) translation rules that specify transformations among elements in metamodels. Details are in 5.6;

   2) operation rules that specify mismatches between IoT systems. Details are in 6.3.

b) In Clause 6, requirements on IoT-related information are specified. Requirements include:

   1) firstly, the required properties related to IoT devices for translation rules (specified in 6.2). For example, an identifier of an IoT system or an IoT device is a required property;

   2) secondly, the required properties and resolutions for mismatches between IoT systems for operation rules. Mismatches occur during information exchange between IoT systems. Resolutions are required to resolve these mismatches. For example, if the time interval requesting information exchange is different, i.e. not matched in involved IoT systems for their interoperability, then syntactic resolutions are required to fill up this mismatch. Required properties and resolutions for mismatches are analysed and described in 6.3.

c) In Clause 7, a framework of how to create information exchange rules is specified. The necessary procedures to realize the IoT syntactic interoperability following the proposed approach are defined. Whether it is necessary to create or extend an IoT system's metamodel, what kinds of information exchange rules are defined, and how exchange rules can be executed and evaluated are also described.

5.5 The methodology of metamodel-driven information exchange

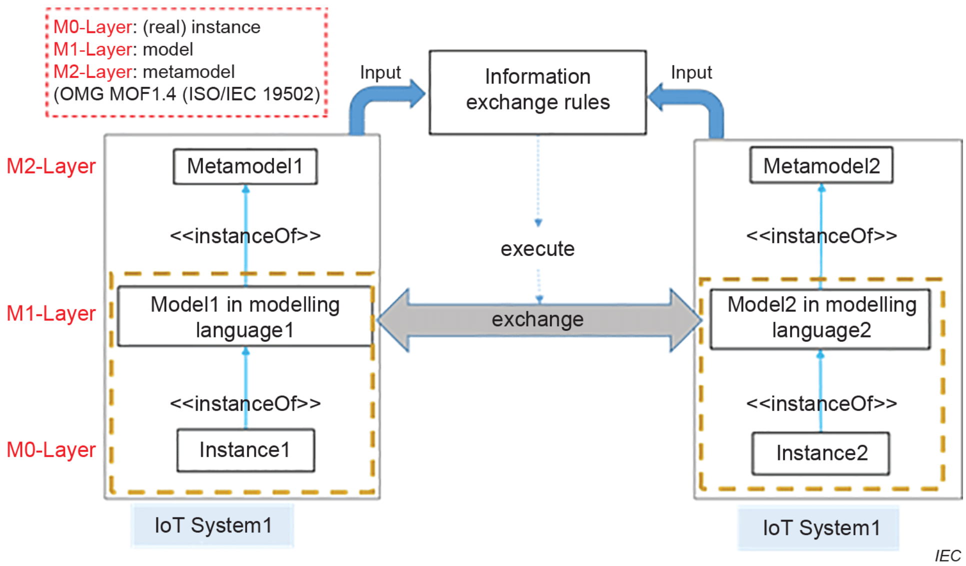


Figure 2 – Model hierarchies and metamodel-driven information exchange rules

During the last decades, in the field of model-based engineering (MBE), models have been constructed to represent information from the physical world. The community of OMG proposes MOF (ISO/IEC 19502 [15]), a four-layer modelling architecture to describe models. Models here in general include the instance in M0-Layer, the model in M1-Layer, the metamodel in M2-Layer, and the meta-metamodel in M3-Layer. M3-Layer is not included in this document thus it is omitted from Figure 2.

As shown in Figure 2, the model in M1-Layer defines structures, available entities, relationships, etc. for instances in M0-Layer, and the metamodel in M2-Layer specifies the syntax for the models. Therefore, models in M1-Layer are the instances of their metamodel in M2-Layer, i.e. M1-Layer has relationships with M2-Layer as "<<instanceOf>>". And the same relationships exist between M0-Layer and M1-Layer. Each metamodel can have many models and each model can have many instances. In Figure 2, Model1 in IoT System1 is the model of Instance1, and Metamodel1 is the metamodel of Model1. Model2 and Metamodel2 in IoT System2 have the same relationships.

From the syntactic point of view, information exchange rules as projections allow converting information in all layers from a specific system to information in another system in a modelling environment. The information exchange rules based on metamodels in M2-Layer are applicable to the transformation of models in M1-Layer [15][16] because the information in M1-Layer is defined with elements available in M2-Layer. The same relationships are applicable to M1-Layer and M0-Layer. Therefore, the metamodel-based information exchange rules are applicable to its models and instances.

5.6 Information exchange rules

5.6.1 Categories of information exchange rules

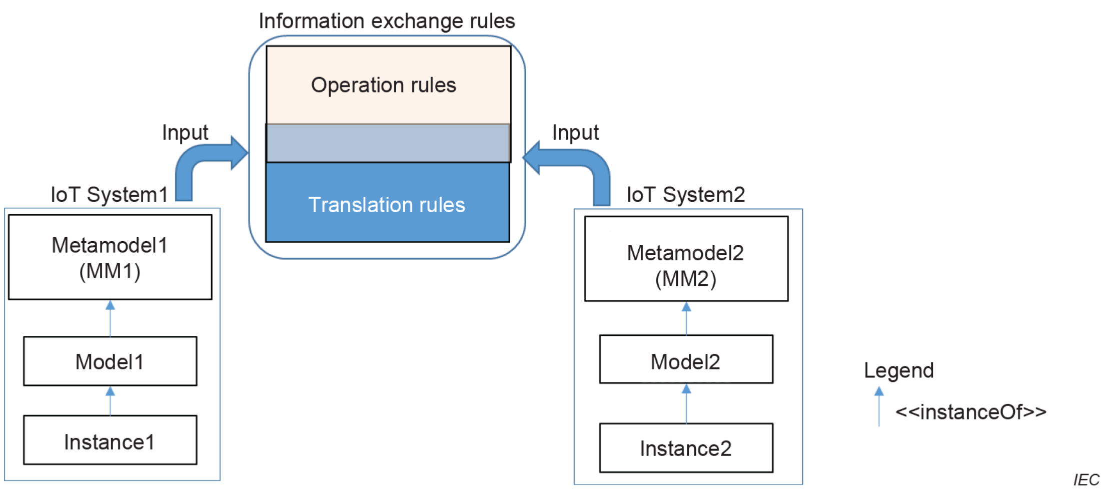
As explained in 5.2 and 5.5, for an IoT system including IoT devices (IoT System1), in order to achieve syntactic interoperability with other IoT systems and devices (IoT System2), information exchange rules are adopted. Figure 3 shows that the information exchange rules can be classified into two categories.

– Translation rules

  Translation rules are created with elements in the metamodels of IoT System1 and IoT System2. Elements in the metamodels are classes, properties, relationships, etc. Transformation rules among these elements are defined and named "translation rules" in order to achieve structure, data format, and syntactic constraints transformations between IoT systems. Required properties for translation rules are specified in 6.2.

– Operation rules

  Operation rules are specified to resolve mismatches between two IoT systems. Potential operational mismatches that happen during processes of achieving interoperability are detected. To solve these mismatches, necessary properties and available resolutions are specified. Mismatches that cannot be resolved from syntactic viewpoints are out of the scope. Simultaneously, resolutions not based on syntactic approaches for mismatches are also out of the scope. Details of the operation rules are specified in 6.3.



The overlapped area includes properties used both in translation rules and operation rules.

Figure 3 – Categories of information exchange rules

5.6.2 Information exchange rules expression

Information exchange rules shall include translation rules among metamodels, and operation rules for IoT syntactic interoperability. Information exchange rules can be expressed in various languages. Some well-known languages such as QVT (Query/View/Transformation [17]), OCL (Object Constraint Language [18]), ATL (Atlas Transformation Language [19]), TGG (Triple graph grammar [20], [21]), etc. can be applied to describe information exchange rules among different metamodels and models. This document does not provide new languages for information exchange rules, and sample information exchange rules are described in Annex B. An implementation of this document can define information exchange rules with selected language and data format.

5.6.3 Information exchange rules expression example

```
(1)   module Probe2Fiware;
(2)   create OUT: Fiware from IN: ProbeVehicle;
(3)   -- translation rule: from "name" to "identifier"
(4)   rule id {
(5)         from
(6)               p:Probe!ProbeDataElementType,
(7)               pv:Probe!VelocityType
(8)         to
(9)               v:Fiware!Vehicle_schema(
(10)                    id <- p."ASN.1_name" + p."ASN.1_object_identifier"
(11)              )
(12)  }
(13)
(14)  -- operation rule: if unit mismatch detected dm to m
(15)  helper def : transDecimeterToMeter(dm: Integer): Integer =
(16)        dm / 10;
```

IEC

Figure 4 – Excerpted information exchange rules for Annex B

Excerpted information exchange rules in ATL for Annex B are listed in Figure 4. In Figure 4;

– IoT System1 is a connected vehicle; IoT System2 is a traffic management system (TMS) adopting FIWARE to represent its system. Metamodels of these two systems are defined in line (2) as IN: ProbeVehicle and OUT: Fiware, respectively.
– Lines (4) to (12) show the translation rule of a vehicle "name" and "identifier" to the TMS "name". Properties utilized in translation rules are defined separately in each metamodel [22] and [23].
– Lines (15) and (16) show an implementation of the operation rule for a unit mismatch resolution. Here, while the unit mismatch is detected, the exchange between different units is specified manually. As explained in 6.3, the implementations of resolutions are out of the scope. This example is a guide for implementers.

6 Requirements on information related to IoT devices

6.1 General

In Clause 6, requirements on the information which is necessary for IoT syntactic interoperability are described. The information shall be defined in the metamodel or model of an IoT system or an IoT device. The requirements apply to IoT devices for the data exchange among IoT systems, excluding cloud-computing-based back-end services.

In coincidence with the two categories of the information exchange rues described in 5.6.1, the requirements for the information related to IoT devices are also classified into two groups: the requirements on translation rules and those on operation rules as shown in Figure 5.

– Requirements on translation rules are further divided into two groups depending on required properties. One group is "Required intrinsic properties of physical IoT devices" and the other is "Required extrinsic properties of physical IoT devices" [24], [25]. They are specified in 6.2.2 and 6.2.3, respectively. An intrinsic property is defined as a property of a specified subject that exists itself or within the subject, and an extrinsic property is a property not essential or inherent to the subject that is being characterized [25].
– Required properties and resolutions for operation rules are specified in 6.3.

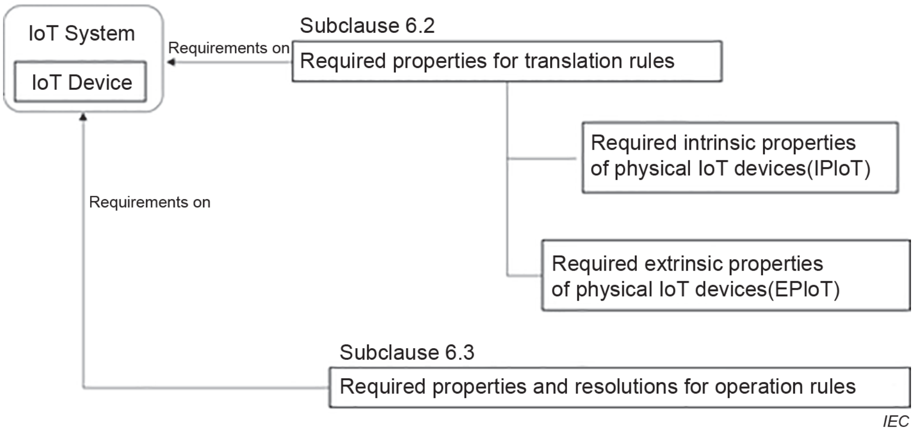


Figure 5 – Classifications of requirements on information related to IoT devices

6.2 General requirements on the translation rules

6.2.1 General

The translation rules are specified with elements in metamodels of IoT systems. An IoT system contains IoT devices, and the information of IoT devices is represented with properties. In 6.2, required properties related to IoT devices for syntactic interoperability are specified.

– No new identification structure nor new data modelling method is specified in this document for IoT syntactic interoperability.
– Existing ID standards and data models adopted in the IoT systems shall be applied if they are used in an IoT system while realizing its syntactic interoperability.
– For each property, no specific property definitions, formats, or classifications are required. But if there are standards for its definition, format, etc., and these standards are used in an IoT system, then the property complying with these standards shall be applied for realizing its IoT syntactic interoperability.

NOTE The above descriptions are to avoid misunderstanding.

6.2.2 Required intrinsic properties of physical IoT devices (IPIoT)

In order to support IoT syntactic interoperability, intrinsic properties of physical IoT devices are required and provided by an IoT system. Available informative intrinsic properties are listed in Clause A.1. Some typical properties utilized in IoT use cases are explained in Table 1.

Table 1 – Required intrinsic properties of physical IoT devices

Property name	Description	Mandatory/Optional
ID	IoT device identifier based on a given standardized object identification system.	Mandatory
Name	IoT device name.	Optional NOTE For example, UID, IRDI, identifiable string name such as "DevicID.temperature" can be used.
DeviceType	Type of an IoT device. It shall be a sensor, actuator, composed IoT device, and any user-defined device type.	Mandatory NOTE Value of this property can be null.
Location	Uniquely identifiable physical point or area Note 1 to entry: The location can be characterized by coordinates. [SOURCE: ISO 29404:2015, 3.11] NOTE For a moveable IoT device, the current coordinates obtained from a positioning system such as GPS can be used as the location. For a fixed IoT device its setting location can be utilized.	Optional
DeviceOwner	Person(s) or organization(s) which has legal title to the product to be used Note 1 to entry: The owner may also be the operator. [SOURCE: ISO/TR 20183:2015, 2.21]	Optional
MaintenanceRecord	Device maintenance history	Optional

6.2.3 Required extrinsic properties of physical IoT devices (EPIoT)

In order to support IoT syntactic interoperability, extrinsic properties of physical IoT devices are required and provided by an IoT system. Extrinsic properties shall be defined in a metamodel/model of an IoT device or an IoT system. Available informative extrinsic properties are listed in Clause A.2. Some typical properties utilized in IoT use cases are explained in Table 2.

Table 2 – Required extrinsic properties of physical IoT devices

Property name	Description	Mandatory /Optional
DataID	Identifier of a piece of data based on a given standardized or user-defined data reference system.	Mandatory
DeviceID	ID of the device from which a datum is collected. NOTE "ID" in Table 1 shall be used.	Mandatory
Value	Data value [SOURCE: ISO/IEC 20944-1:2013, 3.18.2.7]	Mandatory
Timestamp	Attribute or field in data which denotes the time of data generation [SOURCE: ISO/TS 27790:2009, 3.73]	Mandatory
Accuracy	Closeness of agreement between a test result or measurement result and the true value. [SOURCE: ISO 3534-2:2006, 3.3.1]	Optional
AccessAuthority	Access permission such as forbidden, readable, writable, executable to device datum. NOTE Device data means data produced by the device in operation.	Optional

6.3 General requirements on the operation rules

6.3.1 Overview of mismatches between IoT systems

Required properties and resolutions for operation rules are related to the mismatches between what one IoT system is expecting and what the other IoT system can provide. A mismatch is a difference between these two IoT systems regarding a specified property for data. To accomplish syntactic interoperability, the mismatches between IoT systems are detected by comparing the required properties of two IoT systems. The operation rules are prepared to resolve the mismatches. These required properties and resolutions are defined as requirements on the operation rules.

Figure 6 shows the overall procedures for mismatch detection and resolution. Firstly, the mismatch is detected by comparing the required properties. If the property is defined in the metamodel, the translation rule is created to resolve the format and structural differences. After creating the translation rules, the operation rules are created. If the property is not defined in the metamodel, either the metamodel is extended to include the property, or operation rules are directly created.

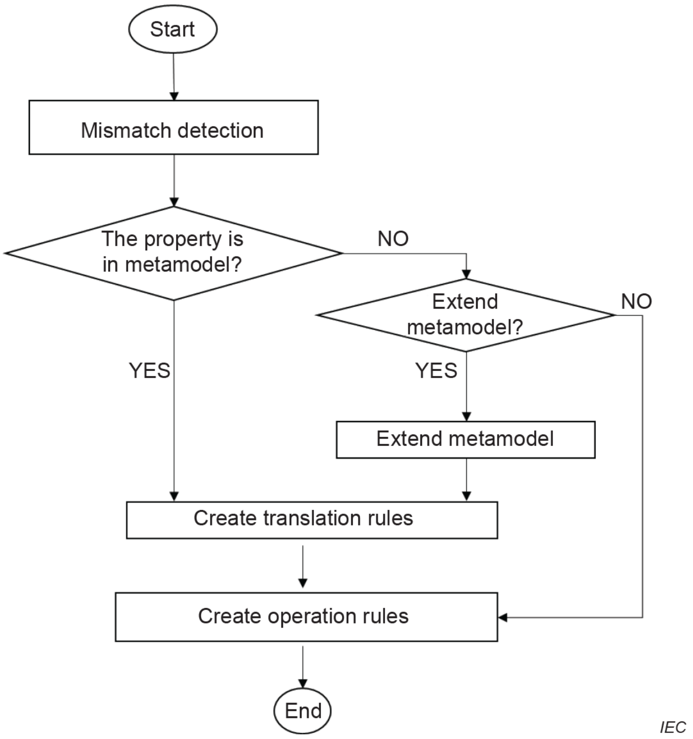


Figure 6 – A procedure for mismatch detection and resolution

Figure 7 shows an example of IoT mismatch and its syntactic resolution. IoT System2 requires data with 3 significant figures for the temperature, while IoT System1 can provide data with 5 significant figures. Required properties for this mismatch are "significantFigure". The "significantFigure" describes the precision or uncertainty of data by the number of digits. The property of IoT System1 has RDF format, while the property of IoT System2 has JSON format, thus they have format differences. They also have structural differences.

In the example in Figure 7, firstly, the mismatch is detected by comparing the "significantFigure" properties. Then, the differences in format and structure are resolved by translation rules. For example, IoT System1's significantFigure property is translated to JSON format: "significantFigure":{"type":"int", "value":5}. Then, the operation rule detects the mismatch by comparing the value of the "significantFigure" property between IoT System1 (i.e. 5) and IoT System2 (i.e. 3). In case that the prepared "syntactic resolution" for this "significantFigure mismatch" is "truncation", a user can implement the "truncation" function to establish interoperability between IoT System1 and IoT System2. Finally, the value or temperature "24.475" is truncated to "24.4" in this case.

These kinds of mismatches are intended to be resolved by the operation rules. However, for each mismatch, it can have resolution methods from various perspectives such as syntactic, semantic, policy-based, etc. Only syntactic resolutions are specified, and resolutions from other aspects of IoT interoperability are out of the scope.

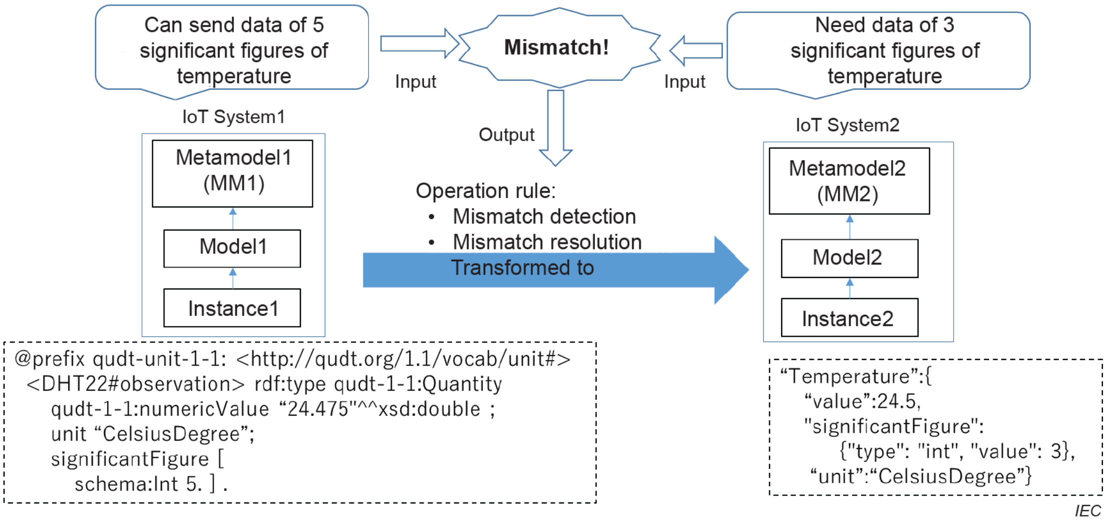


Figure 7 – An example of mismatch detection and resolution

6.3.2 Required properties and syntactic resolutions for potential IoT mismatches

In 6.3.2, required properties and syntactic resolutions for potential IoT mismatches are described. Table 3 lists the minimal required properties and resolutions for the potential mismatches between IoT systems.

– The required properties and syntactic resolutions for each mismatch are specified.

– The classification of syntactic resolutions (F: Format, S: Structure, C: Syntactic Constraint) is also specified in the "Type" column of Table 3.

– For reference, sample non-syntactic resolutions are explained in the "non-syntactic resolution" column of Table 3 to clarify that there can exist other resolutions except syntactic ones.

For reference, the data quality indicators defined in ISO/IEC 25012 [26], which might be affected by the mismatches, are described. To avoid degradations of data quality while exchanging information between IoT systems, the resolutions which supplement mismatches shall be considered.

Table 3 – Required properties and resolutions for potential IoT mismatches

Name	Title	Type[a]	Required property	Syntactic resolution	Non-syntactic resolution	Affected data quality (ISO/IEC 25012)	Example
mismatch1	Synchro-nization mismatch	F/S/C	latestSynchronizedTime	"synchronize" or "mismatch notification"		consistency	different time stamp
mismatch2	Sampling frequency mismatch	F/S	samplingFrequency	"resampling" o "interpolation"		accuracy	every hour vs. every minute
mismatch3	Location mismatch	F/S	location	"compensate location differences" or "mismatch notification"		accuracy	room temperatu re at centre vs. wall
mismatch4	Data recording pattern mismatch	F/S	dataRecordingPattern	"compensate recording pattern" or "mismatch notification"		accessibility	periodic vs. change-driven
mismatch5	Precision mismatch	F/S/C	precision	"truncation" or "mismatch notification"		precision	±0,2° C vs. ±1,0° C
mismatch6	Significant figure mismatch	F/S/C	significantFigure	"rounding" or "truncation" or "mismatch notification"	See tables in 6.3.3	precision	2 digits vs. 5 digits
mismatch7	Range mismatch	F/S	operatingRange	"range check" or "mismatch notification"		credibility	±50,0 °C vs. ±30,0 °C
mismatch8	Calibration mismatch	F/S/C	calibrationTime	"recalibrate" or "compensate calibration differences"		credibility	different timestam p
mismatch9	Response time mismatch	F/S	responseTime	"response time compensatio n" or "mismatch notification"		adjustment	20 ms vs. 40 ms
mismatch10	Acquisition status mismatch	F/S	acquisitionStatus	"status notification"		credibility	failure vs. success
mismatch11	Unit mismatch	F/S	unit	"unit conversion"		precision	Celsius vs. Fahren-heit

[a] F: format, S: structure, C: syntactic constraint.

6.3.3 Details of required properties and syntactic resolutions for potential IoT mismatches

In 6.3.3, each mismatch listed in Table 3 is expressed in detail.

For all tables in 6.3.3, the "Required property" line defines the property by a pseudo schema in XML format. The "<name>" tag specifies the required name of the property. The "<datatype>" tag adopts IEC 61360-1:2017 [27] for its description. Other tags in the XML are for additional information of the property.

The "Function Signature" in "Mismatch detection" and "Syntactic resolution" lines specifies the method name, arguments, and requirements.

– Mismatch1: Synchronization mismatch

  See Table 4.

Table 4 – Mismatch1: Synchronization mismatch

Mismatch1: Synchronization mismatch	
Name	mismatch1
Title	Synchronization mismatch
Description	The "Synchronization mismatch" is the mismatch of clocks between IoT systems. For example, if the "latestSynchronizedTime" defined in IoT System1 is "2004-04-01T12:00Z" while in IoT System2 is "2004-01-01T11:00Z", then there is a mismatch.
Required property	<property> <id>MS1</id> <name>latestSynchronizedTime</name> <datatype> DATE_TIME_TYPE </datatype> <description>The "latestSynchronizedTime" records the latest synchronized time with the format of ISO 8601 (RFC 3339), i.e., the time scale of "UTC" shall be used.</description> <resource> ISO/IEC 22417</resource> </property>
Mismatch detection	Function Signature: mismatchDetection(IoT System1.latestSynchronizedTime, IoT System2.latestSynchronizedTime) Requirement: The synchronization mismatch shall be detected when the "latestSynchronizedTime" property defined in these two IoT systems does not match.
Syntactic resolution	Function Signature: syntacticResolution(IoT System1.latestSynchronizedTime, IoT System2.latestSynchronizedTime) Requirement: The syntactic resolution shall implement either "synchronize" or "mismatch notification".
Non-syntactic resolution	Non-syntactic resolutions are out of the scope of this document. For example, a resolution can be realized through a hardware-based time synchronization or a software-based time synchronization [28].

– mismatch2: Sampling Frequency mismatch

See Table 5.

Table 5 – Mismatch2: Sampling frequency mismatch

mismatch2: Sampling frequency mismatch	
Name	mismatch2
Title	Sampling frequency mismatch
Description	The "Sampling frequency mismatch" is the mismatch of data sampling frequencies between IoT systems. For example, if IoT System2 requires data every minute, while IoT System1 can only provide data hourly, then there is a mismatch.
Required property	<property> <id>MS2</id> <name> samplingFrequency </name> <datatype> REAL_MEASURE_TYPE</datatype> <description> The "samplingFrequency" is the frequency of data sampling cycles.</description> <resource> [https://www.w3.org/TR/2017/REC-vocab-ssn-20171019/,https://w3id.org/iot/qoi, http://www.ontology-of-units-of-measure.org/resource/om-2/Unit]</resource> </property>
Mismatch detection	Function Signature: mismatchDetection(IoT System1.samplingFrequency, IoT System2.samplingFrequency) Requirement: The sampling frequency mismatch shall be detected when the "samplingFrequency" property defined in these two IoT systems does not match.
Syntactic resolution	Function Signature: syntacticResolution(IoT System1.samplingFrequency, IoT System2.samplingFrequency) Requirement: The syntactic resolution shall implement either "resampling" or "interpolation".
Non-syntactic resolution	Non-syntactic resolutions are out of the scope of this document. For example, a resolution can be realized through a policy-based harmonization of data acquisition timings.

– mismatch3: Location mismatch

  See Table 6.

Table 6 – Mismatch3: Location mismatch

mismatch3: Location mismatch	
Name	mismatch3
Title	Location mismatch
Description	The "Location mismatch" is the mismatch of locations between IoT systems. For example, if IoT System1 requires the "room temperature" that represents the temperature of the room, while IoT System2 can provide the temperature measured at the wall, then there is a mismatch.
Required property	<property> <id>MS3</id> <name> location </name> <datatype> STRING_TYPE(geo URI) </datatype > <description> The "location" is the geospatial information where the data is acquired or expected to be acquired at a given coordinate reference system (CRS). The CRS shall be specified in the datatype of geo URI. </description> <resource>"geo URI"=https://tools.ietf.org/rfc/rfc5870 </resource> </property>
Mismatch detection	Function Signature: mismatchDetection(IoT System1.location, IoT System2.location) Requirement: The location mismatch shall be detected when the "location" property defined in these two IoT systems does not match.
Syntactic resolution	Function Signature: syntacticResolution(IoT System1.location, IoT System2.location) Requirement: The syntactic resolution shall implement either "compensate location differences" or "mismatch notification".
Non-syntactic resolution	Non-syntactic resolutions are out of the scope of this document. For example, a resolution can be realized by interpolation or correction factors.

– mismatch4: Data recording pattern mismatch

See Table 7.

Table 7 – Mismatch4: Data recording pattern mismatch

mismatch4: Data recording pattern mismatch	
Name	mismatch4
Title	Data recording pattern mismatch
Description	The "Data recording pattern mismatch" is the mismatch of data acquisition timings between IoT systems. For example, if IoT System1 expects periodically recorded data, but IoT System2 can only record data when data change is acknowledged, then there is a mismatch.
Required property	<property> <id>MS4</id> <name> dataRecordingPattern </name> <datatype> ENUM_CODE_TYPE ("periodic","change-driven") </datatype> <description>The "dataRecordingPattern" specifies the pattern indicating whether the information is acquired with a periodical specified frequency or through a change-driven method.</description> <resource> https://www.w3.org/TR/websub/ </resource> </property>
Mismatch detection	Function Signature: mismatchDetection(IoT System1.dataRecordingPattern, IoT System2.dataRecordingPattern) Requirement: The data recording pattern mismatch shall be detected when the "dataRecordingPattern" property defined in these two IoT systems does not match.
Syntactic resolution	Function Signature: syntacticResolution(IoT System1.dataRecordingPattern, IoT System2.dataRecordingPattern) Requirement: The syntactic resolution shall implement either "compensate recording pattern" or "mismatch notification".
Non-syntactic resolution	Non-syntactic resolutions are out of the scope of this document. For example, a resolution can be realized by interpolation or a change point detection.

– mismatch5: Precision mismatch

See Table 8.

Table 8 – Mismatch5: Precision mismatch

mismatch5: Precision mismatch	
Name	mismatch5
Title	Precision mismatch
Description	The "Precision mismatch" is the mismatch of data precisions between IoT systems. For example, if IoT System1 requires the precision ±0,2 °C of the temperature sensor, while IoT System2 can only provide ±1,0 °C precision, then there is a mismatch.
Required property	<property> <id>MS5</id> <name> precision </name> <datatype> REAL_MEASURE_TYPE</datatype> <description> The "precision" is a quantitative symmetrical or asymmetric offset of a physical quantity by specified values with its associated unit at specified conditions.</description> <resource> https://www.w3.org/TR/vocab-ssn/ </resource> </property>
Mismatch detection	Function Signature: mismatchDetection(IoT System1.precision, IoT System2.precision) Requirement: The precision mismatch shall be detected when the "precision" property defined in these two IoT systems does not match.
Syntactic resolution	Function Signature: syntacticResolution(IoT System1.precision, IoT System2.precision) Requirement: The syntactic resolution shall implement either "truncation" or "mismatch notification".
Non-syntactic resolution	Non-syntactic resolutions are out of the scope of this document. For example, a resolution can be realized through a policy-based harmonization of data precisions.

– mismatch6: Significant figure mismatch

See Table 9.

Table 9 – Mismatch6: Significant figure mismatch

mismatch6: Significant figure mismatch	
Name	mismatch6
Title	Significant figure mismatch
Description	The "Significant figure mismatch" is the mismatch of significant figures of data between IoT systems. For example, if IoT System1 requires two significant figures for temperature data, while IoT System2 can provide five significant figures, then there is a mismatch.
Required property	<property> <id>MS6</id> <name> significantFigure </name> <datatype> REAL_TYPE</datatype> <description>The "significantFigure" is used to express, in an approximate way, the precision or uncertainty associated with a reported numerical result. </description> <resource> https://www.w3.org/TR/vocab-ssn/ </resource> </property>
Mismatch detection	Function Signature: mismatchDetection(IoT System1.significantFigure, IoT System2.significantFigure) Requirement: The significant figure mismatch shall be detected when the "significantFigure" property defined in these two IoT systems does not match.
Syntactic resolution	Function Signature: syntacticResolution(IoT System1.significantFigure, IoT System2.significantFigure) Requirement: The syntactic resolution shall implement "rounding" or "truncation" or "mismatch notification".
Non-syntactic resolution	Non-syntactic resolutions are out of the scope of this document. For example, a resolution can be realized through a policy-based harmonization of significant figures.

– mismatch7: Range mismatch

See Table 10.

Table 10 – Mismatch7:Range mismatch

mismatch7: Range mismatch	
Name	mismatch7
Title	Range mismatch
Description	The "Range mismatch" is the mismatch of operating ranges between IoT systems. For example, if IoT System1 expects ±50,0 °C operating range while IoT System2 can only provide ±30,0 °C operating range, then there is a mismatch.
Required property	<property> <id>MS7</id> <name> operatingRange </name> <datatype> LEVEL (MIN, MAX) OF REAL_MEASURE_TYPE </datatype> <description> The "operatingRange" describes a required range inside which a system is operated.</description> <resource> https://www.w3.org/TR/vocab-ssn/ </resource> </property>
Mismatch detection	Function Signature: mismatchDetection(IoT System1.operatingRange, IoT System2.operatingRange) Requirement: The range mismatch shall be detected when the "operatingRange" property defined in these two IoT systems does not match.
Syntactic resolution	Function Signature: syntacticResolution(IoT System1.operatingRange, IoT System2.operatingRange) Requirement: The syntactic resolution shall implement either "range check" or "mismatch notification".
Non-syntactic resolution	Non-syntactic resolutions are out of the scope of this document. For example, a resolution can be realized through a system-wide harmonization of operating ranges.

– mismatch8: Calibration mismatch

See Table 11.

Table 11 – Mismatch8: Calibration mismatch

mismatch8: Calibration mismatch	
Name	mismatch8
Title	Calibration mismatch
Description	The "Calibration mismatch" is the mismatch of calibration times at a given time scale. For example, if IoT System1 requires a specific calibration time, while IoT System2 can only provide a different calibration time, then there is a mismatch.
Required property	<property> <id>MS8</id> <name> calibrationTime </name> <datatype> REAL_MEASURE_TYPE </datatype> <description> The "calibrationTime" describes the timestamp when the system is calibrated. </description> <resource> https://www.w3.org/TR/vocab-ssn/ </resource> </property>
Mismatch detection	Function Signature: mismatchDetection(IoT System1.calibrationTime, IoT System2.calibrationTime) Requirement: The calibration mismatch shall be detected when the "calibrationTime" property defined in these two IoT systems does not match.
Syntactic resolution	Function Signature: syntacticResolution(IoT System1.calibrationTime, IoT System2.calibrationTime) Requirement: The syntactic resolution shall implement either "recalibrate" or "compensate calibration differences".
Non-syntactic resolution	Non-syntactic resolutions are out of the scope of this document. For example, a resolution can be realized through adjustments based on a system-wide policy.

– mismatch9: Response time mismatch
  See Table 12.

Table 12 – Mismatch9: Response time mismatch

mismatch9: Response time mismatch	
Name	mismatch9
Title	Response time mismatch
Description	The "Response time mismatch" is the mismatch of response times between IoT systems. For example, if IoT System1 expects "20 ms" for the response time, but IoT System2 provides a "40 ms" response time, then there is a mismatch.
Required property	<property> <id>MS9</id> <name> responseTime </name> <datatype> REAL_MEASURE_TYPE </datatype> <description> The "responseTime" is the difference between the time when an IoT device sets a value and when the IoT device observes this value. </description> <resource> https://www.w3.org/TR/vocab-ssn/ </resource> </property>
Mismatch detection	Function Signature: mismatchDetection(IoT System1.responseTime, IoT System2.responseTime) Requirement: The response time mismatch shall be detected when the "responseTime" property defined in these two IoT systems does not match.
Syntactic resolution	Function Signature: syntacticResolution(IoT System1.responseTime, IoT System2.responseTime) Requirement: The syntactic resolution shall implement either "response time compensation" or "mismatch notification".
Non-syntactic resolution	Non-syntactic resolutions are out of the scope of this document. For example, a resolution can be realized through a policy-based compensation of response times.

– Mismatch10: Acquisition status mismatch

See Table 13.

Table 13 – Mismatch10: Acquisition status mismatch

Mismatch10: Acquisition status mismatch	
Name	mismatch10
Title	Acquisition status mismatch
Description	The "Acquisition status mismatch" is the mismatch between data acquisition statuses. For example, if IoT System1 expects that the data is acquired successfully, while IoT System2 fails to acquire data or can only provide erroneous data, then there is a mismatch.
Required property	<property> <id>MS10</id> <name> acquisitionStatus </name> <datatype> ENUM_CODE_TYPE ("success","failure","error") </datatype> <description>The "acquisitionStatus" is an enumeration of states occurring in the data acquisition among different IoT systems. It can be "success" or "failure" or "error". </description> <resource> https://www.w3.org/TR/vocab-ssn/ </resource> </property>
Mismatch detection	Function Signature: mismatchDetection(IoT System1.acquisitionStatus, IoT System2.acquisitionStatus) Requirement: The "Acquisition status mismatch" shall be detected when the " acquisitionStatus" property defined in these two IoT systems does not match.
Syntactic resolution	Function Signature: syntacticResolution(IoT System1.acquisitionStatus, IoT System2.acquisitionStatus) Requirement: The syntactic resolution shall implement "status notification".
Non-syntactic resolution	Non-syntactic resolutions are out of the scope of this document. For example, a resolution can be realized through a translation or notification of failure codes and error codes.

– Mismatch11: Unit mismatch

  See Table 14.

Table 14 – Mismatch11: Unit mismatch

Mismatch11: Unit mismatch	
Name	mismatch11
Title	Unit mismatch
Description	The "Unit mismatch" is the mismatch of the units of measurements between IoT systems. For example, if IoT system 1 expects room temperature in "Celsius" while IoT system 2 provides temperature in "Fahrenheit", then there is a mismatch.
Required property	<property> <id>MS11</id> <name> unit </name> <datatype> CLASS_REFERENCE_TYPE(0112/2///62720) </datatype> <description> The "unit" is the unit of the measurement. </description> <resource> IEC 62720 </resource> </property>
Mismatch detection	Function Signature: mismatchDetection(IoT System1.unit, IoT System2.unit) Requirement: The "Unit mismatch" shall be detected when the "unit" property defined in these two IoT systems does not match.
Syntactic resolution	Function Signature: syntacticResolution(IoT System1.unit, IoT System2.unit) Requirement: The syntactic resolution shall implement "unit conversion".
Non-syntactic resolution	Non-syntactic resolutions are out of the scope of this document. For example, a resolution can be realized through a system-wide harmonization of units [29].

7 A framework for IoT syntactic interoperability

7.1 General

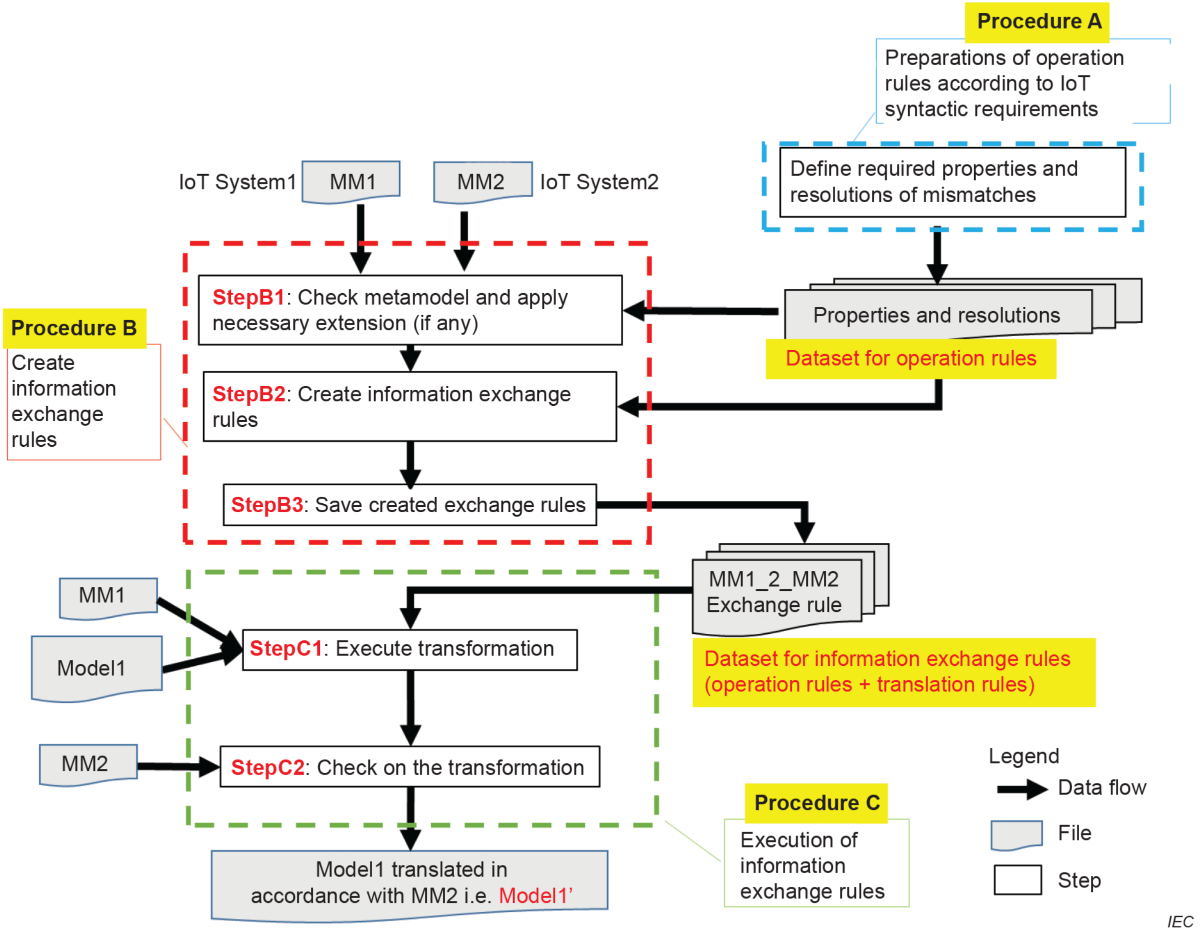


Figure 8 – A framework for processes on developing information exchange rules related to IoT devices from the syntactic viewpoint

In order to realize the syntactic interoperability between IoT systems, a framework shown in Figure 8 shall be provided according to descriptions in Clause 5 and Clause 6. The implementation of the framework is out of the scope.

Procedures in the framework are classified into three groups.

– Procedure A enclosed by a blue dashed line is to prepare necessary properties and resolutions based on the requirements of 6.3. This procedure's output is "dataset for operation rules"(DOR).

– Procedure B enclosed by a red dashed line is to create information exchange rules. This procedure's output is "dataset for information exchange rules" (DIER). DIER is composed of DOR and "dataset for translation rules" (DTR).

– Procedure C enclosed by a green dashed line is to execute the information exchange rules and check the result.

In Figure 8, the created DOR shall be reused by IoT systems for syntactic interoperability. And the DIER shall be reused if an IoT system has the same metamodel as Metamodel1 (MM1) or Metamodel2 (MM2).

Once the DOR and DIER are created, Procedure A and Procedure B can be omitted for two IoT systems that are satisfied to reuse DOR and DIER.

7.2 A conceptual model for dataset of operation rules (DOR)

Figure 9 shows the conceptual model for DOR. The following requirements apply for the classes shown in Figure 9.

- DOR shall be created through Procedure A, and it shall contain required properties and required resolutions for mismatches defined in Table 2, required intrinsic properties for IoT devices defined in 6.2.2 and required extrinsic properties for IoT devices specified in 6.2.3.
- The class "RequiredPropertyOfMismatch" shall define required properties for mismatches.
- The class "RequiredSetOfIPIoT" shall define required intrinsic properties for IoT devices.
- The class "RequiredSetOfEPIoT" shall define required extrinsic properties for IoT devices.
- The class "RequiredResolutionOfMismatch" shall inherit all properties from the above three classes.
- The class "RequiredResolutionOfMismatch" shall contain possible resolutions for mismatches. Each mismatch in 6.3.3 is defined as a class, which shall contain the required properties and syntactic resolutions for the mismatch. The class representing each mismatch has a name the same as the title of each mismatch. For example, the "Class:Synchronization mismatch" is defined in accordance with mismatch1 in 6.3.3.This class has a property as "latestSynchronizedTime" and a syntactic resolution as "syntacticResolution()". Other classes for mismatches are defined in the same approach.

All entities in this conceptual model shall be flexibly modified, updated, deleted and added according to evolving requirements from IoT systems and IoT devices.

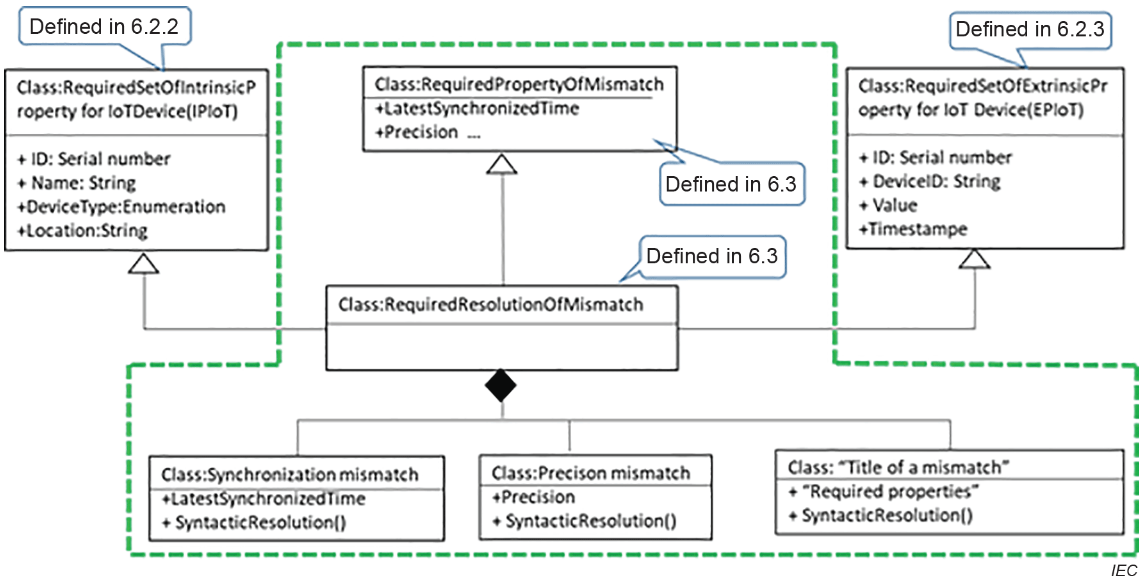


Figure 9 – An excerpted conceptual model of DOR (dataset of operation rules)

7.3 Detailed procedures for a syntactic interoperability framework

7.3.1 Procedure A to prepare the required properties and resolutions

In order to prepare DOR according to descriptions in Clause 6, an overall flowchart of Procedure A is illustrated in Figure 10. In this procedure, the following steps shall be equipped. The order of step A1, step A2, and step A3 can be changed.

- In Step A1, properties for IPIoT shall be defined to achieve syntactic interoperability.
- In Step A2, properties for EPIoT shall be defined.
- In Step A3, properties and syntactic resolutions for IoT mismatches shall be defined.

– In Step A4, all defined data shall be saved to DOR. The DOR shall be in coincidence with the conceptual model shown in Figure 9.
– When there are updates on DOR, updated data shall be defined in Step A5 and be saved to an updated DOR.

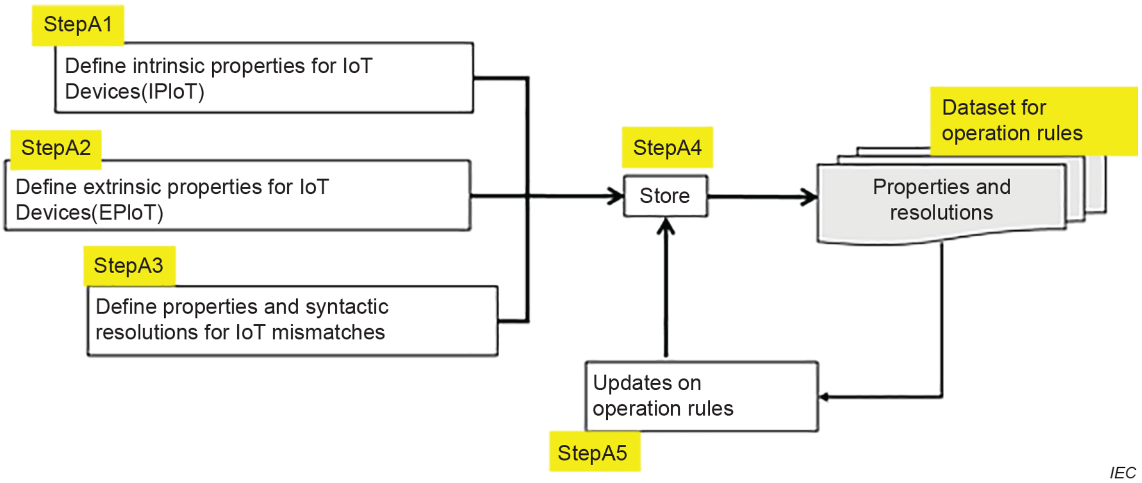


Figure 10 – Steps of Procedure A

7.3.2 Procedure B to create information exchange rules (DIER)

A general flowchart of Procedure B is shown in Figure 8 from step B1 to step B3. Through Procedure B, information exchange rules between two IoT systems can be generated. The following steps are required.

– In step B1, if Metamodel1 (MM1) of IoT System1 and Metamodel2 (MM2) of IoT System2 do not contain mandatory properties required for the syntactic interoperability, then necessary properties shall be considered to be appropriately added to the metamodels.
– In step B2, information exchange rules between MM1 and MM2 shall be created.
– In step B3, created information exchange rules shall be saved as DIER, which shall be utilized to execute information transformation between MM1 and MM2.

Information exchange rules can be one-directional or bi-directional.

The information exchange rules are dependent on MM1 and MM2. IoT systems complying with MM1 and MM2 can reuse these information exchange rules for their syntactic interoperability.

7.3.3 Procedure C to execute the information exchange rules and check the result

A flowchart of Procedure C is shown in Figure 8, enclosed by a green dashed line. Through Procedure C, Model1 and data in IoT System1 can be transformed into an IoT System2 or vice versa. In Procedure C, the following steps shall be installed.

– In step C1, data in IoT System1, i.e. MM1 and Model1 in IoT System1, and the created information exchange rules shall be input to execute the information transformation.
– Through step C1, Model1 is transformed to Model1' which shall be in accordance with MM2, the metamodel of IoT System2.
– In step C2, it shall be checked whether Model1' complies with MM2. If Model1' complies with MM2, then Model1' can be understood and utilized by IoT System2.

Through procedures in Clause 7, the syntactic interoperability between IoT System1 and IoT System2 can be achieved with the created information exchange rules based on their metamodels.

Annex A
(informative)

Properties for physical IoT devices and data

A.1 Intrinsic properties of physical IoT devices

Possible intrinsic properties of physical IoT devices are listed in Table A.1. This list can be evolved in accordance with requirements from IoT devices and IoT systems.

Table A.1 – Intrinsic properties of physical IoT devices

ID	Name	M/O[a]	Description	Available IoT use case
IPIoT_P1	ID	M	See "ID" in Table 1.	ISO/IEC TR 22417: IoT usecases
IPIoT_P2	Name	O	See "Name" in Table 1.	ISO/IEC TR 22417: IoT usecases
IPIoT_P3	DeviceType	M/O	See "DeviceType" in Table 1.	ISO/IEC TR 22417: IoT usecases
IPIoT_P4	Location	O	See "Location" in Table 1.	ISO/IEC TR 22417: IoT usecases
IPIoT_P5	Digital communication protocol type	O	value list of protocol types used in digital communication NOTE 1 A product can be manufactured allowing communication by one or many digital communication protocols. NOTE 2 In operation, a product occurrence is using one defined communication protocol only.	ISO/IEC TR 22417: IoT usecases
IPIoT_P6	HeartBeatCurrentStatus	O	qualifier that specifies operating stages of an item [SOURCE: IEC 61360-4 CDD, 61360_4#ADA356 – operational state qualifier]	ISO/IEC TR 22417: IoT usecases
IPIoT_P7	Event	O	Events emitted from the IoT device	ISO/IEC TR 22417: IoT usecases
IPIoT_P8	SecurityLevelOfDevice	O	Combination of a hierarchical security classification and a security category that represents the sensitivity of an object or the security clearance of an individual [SOURCE: ISO/IEC 20944-1: 2013, 3.11.1.14]	ISO/IEC TR 22417: IoT usecases
IPIoT_P9	CommunicationAddress	O	Address that is permanently assigned to a device or storage location and that identifies the device or location without the need for translation or calculation [SOURCE: ISO/IEC/IEEE 24765:2017, 3.19]	ISO/IEC TR 22417: IoT usecases
IPIoT_P10	Receiver	O	Party intended to receive the message [SOURCE: ISO 16609:2012, 3.20] NOTE Party stands for a device.	ISO/IEC TR 22417: IoT usecases

ID	Name	M/O[a]	Description	Available IoT use case
IPIoT_P11	Power	O	Source of energy (battery, mains) [SOURCE: ISO 14708-5: 2020, 3.18]	ISO/IEC TR 22417: IoT usecases
IPIoT_P12	Port	O	interface for communicating with a computer program over a network [SOURCE: ISO 17532:2007, 3.29]	ISO/IEC TR 22417: IoT usecases
IPIoT_P13	Sender	O	Party responsible for, and authorized to, send a message [SOURCE: ISO 16609:2012, 3.21] NOTE Party stands for a device.	ISO/IEC TR 22417: IoT usecases
IPIoT_P14	Description	O	Description of an IoT device	ISO/IEC TR 22417: IoT usecases
IPIoT_P15	DeviceOwner	O	See "DeviceOwner" in Table 1.	ISO/IEC TR 22417: IoT usecases
IPIoT_P17	AC|DC	O	Power supply type: AC or DC	ISO/IEC TR 22417: IoT usecases
IPIoT_P18	Cardinality	O	Cardinality with related equipment	ISO/IEC TR 22417: IoT usecases
IPIoT_P19	StateOfDevice	O	IoT device current status(running|stopped|error|, etc.)	ISO/IEC TR 22417: IoT usecases
IPIoT_P20	PrivacyOfDevice	O	IoT device privacy (privacy information, level, etc.)	ISO/IEC TR 22417: IoT usecases
IPIoT_P21	SamplingFrequency	O	Frequency of data collection from IoT device	ISO/IEC TR 22417: IoT usecases
IPIoT_P22	ContinuousWorkingPeriod	O	Period (months/days/hours, etc.) that an object has been continuously running	ISO/IEC TR 22417: IoT usecases
IPIoT_P23	Size	O	Relevant dimensional characteristics of the equipment as defined by the manufacturer [SOURCE: ISO 10432:2004, 3.25]	ISO/IEC TR 22417: IoT usecases
IPIoT_P24	RatedPower	O	A conventional value of apparent power, establishing a basis for the design of a transformer, a shunt reactor or an arc-suppression coil, the manufacturer's guarantees and the tests, determining a value of the rated current that may be carried with rated voltage applied, under specified conditions [SOURCE: IEC 60050-421: 1990, 421-04-04]	ISO/IEC TR 22417: IoT usecases
IPIoT_P25	MaintenanceRecord	O	See "MaintenanceRecord" in Table 1.	ISO/IEC TR 22417: IoT usecases
IPIoT_P26	RoHS	O	RoHS compliance status	ISO/IEC TR 22417: IoT usecases
IPIoT_P27	StartTimestamp| OnTimestamp	O	Startup timestamp	ISO/IEC TR 22417: IoT usecases

ID	Name	M/O[a]	Description	Available IoT use case
IPIoT_P28	StopTimestamp| OffTimestamp	O	Shutdown timestamp	ISO/IEC TR 22417: IoT usecases
IPIoT_P29	OnOffCount	O	Number of on/off	ISO/IEC TR 22417: IoT usecases
IPIoT_P30	Relation	O	Relationship with related equipment	ISO/IEC TR 22417: IoT usecases
IPIoT_P31	SecurityLevelOfEnvironment	O	Security level of the location where an IoT device is installed	ISO/IEC TR 22417: IoT usecases
IPIoT_P32	BatteryLife	O	Life of a timekeeping instrument without a new energy supply [SOURCE: ISO 6426-2:2002, 3.20]	W3C Semantic Sensor Network Ontology
IPIoT_P33	VersionOfSoftware	O	Software version identification	-
IPIoT_P34	VersionOfDevice	O	Device version information (serial number, year, etc.)	-
IPIoT_P35	CO2	O	CO_2 emissions	-
IPIoT_P36	Readable| Writable	O	Restrictions on whether or not an IoT device can be writable or readable	-
IPIoT_P37	MeasurementRange	O	Set of values of measurands for which the error of a measuring instrument is intended to lie within specified limits [SOURCE: ISO 26782:2009, 3.11]	W3C Semantic Sensor Network Ontology
IPIoT_P38	ActuationRange	O	Range of actuator data	W3C Semantic Sensor Network Ontology
[a] M: mandatory; O: optional.				

A.2 Extrinsic properties of physical IoT devices

Possible extrinsic properties of physical IoT devices are listed in Table A.2. This list can be evolved in accordance with requirements from IoT devices and IoT systems.

Table A.2 – Extrinsic properties of physical IoT devices

ID	Name	M/O[a]	Description	Available IoT use case
EPIoT_P1	DataID	M	See "DataID" in Table 2.	-
EPIoT_P2	DeviceID	M	See "DeviceID" in Table 2.	ISO/IEC TR 22417: IoT usecases
EPIoT_P3	Timestamp	M	See "Timestamp" in Table 2.	ISO/IEC TR 22417: IoT usecases
EPIoT_P4	DataFormat	O	arrangement of data in a file or stream [SOURCE: ISO/IEEE 11073-10201:2004, 3.14]	ISO/IEC TR 22417: IoT usecases
EPIoT_P5	DataType	O	Domain of values [SOURCE: ISO 10303-11:2004, 3.3.5]	ISO/IEC TR 22417: IoT usecases
EPIoT_P6	AccessAuthority	O	See "AccessAuthority" in Table 2.	ISO/IEC TR 22417: IoT usecases
EPIoT_P7	Value	M	See "Value" in Table 2.	ISO/IEC TR 22417: IoT usecases
EPIoT_P8	Accuracy	O	See "Accuracy" in Table 2.	ISO/IEC TR 22417: IoT usecases
EPIoT_P9	DataDomain	O	Industry domain of information.	ISO/IEC TR 22417: IoT usecases
EPIoT_P10	DataPrivacy	O	Privacy of device data. NOTE Device data means data produced by the device in operation.	ISO/IEC TR 22417: IoT usecases
EPIoT_P11	ProtocolOfData	O	Information communication protocol. If not specified, the same as when connecting.	ISO/IEC TR 22417: IoT usecases
EPIoT_P12	Analog|Digital	O	Kind of data collected from the device(Analogue/Digital).	ISO/IEC TR 22417: IoT usecases
EPIoT_P13	Security LevelOfData	O	Data security level.	ISO/IEC TR 22417: IoT usecases
EPIoT_P14	MessageFormat	O	Format of a message. If not specified, the same as data-format	ISO/IEC TR 22417: IoT usecases
EPIoT_P15	Version	O	Information version.	ISO/IEC TR 22417: IoT usecases
EPIoT_P16	DataSize	O	Size of data such as byte or megabyte.	ISO/IEC TR 22417: IoT usecases
EPIoT_P17	DataSurvivalPeriod	O	Valid date of information.	ISO/IEC TR 22417: IoT usecases
EPIoT_P18	OwnerOfData	O	Data owner. NOTE Owner has the same description as IPIoT_P15.	ISO/IEC TR 22417: IoT usecases
EPIoT_P19	ValueRange	O	Range of values. Difference between the maximum and minimum values of a set of values.	ISO/IEC TR 22417: IoT usecases
EPIoT_P20	ValidPeriod	O	Expiration date of information. If not specified, there is no expiration date.	ISO/IEC TR 22417: IoT usecases
EPIoT_P21	DiscreteData|ContinuousData	O	Whether the data collected from an IoT device is continuous or distributed.	-
EPIoT_P22	Sensitivity	O	Group of relevant data for a specific purpose.	W3C Semantic Sensor Network Ontology
EPIoT_P23	Precision		The closeness of agreement between independent test results obtained under stipulated conditions. [SOURCE: ISO 3534-2:2006, 3.3.4]	

[a] M: mandatory; O: optional.

Annex B
(informative)

A use case

B.1 General

A use case achieving IoT syntactic interoperability is introduced in Annex B.

B.2 The use case overview: Connected car and vehicle in smart city

This use case introduces how to realize syntactic interoperability between connected cars and smart city vehicle data models with information exchange rules of their metamodels.

In this use case, the information model and data requirements for a connected car in ISO 22837 [30] and ISO 14817-1:2015 [31] are utilized. The information model defined in Annex A of ISO 22837:2009 [30] is accepted as the metamodel MM1 for the probe data of a connected car. This information model uses the subset of UML identified in ISO 14817-1:2015 [31].

The vehicle data model [22] in the FIWARE data model [23] for smart cities is accepted as the metamodel MM2 for vehicles.

An overview of this use case is illustrated in Figure B.1. How the information exchange rules based on the metamodels of connected cars (MM1) and a vehicle in FIWARE (MM2) can be applied to support their syntactic interoperability is described.

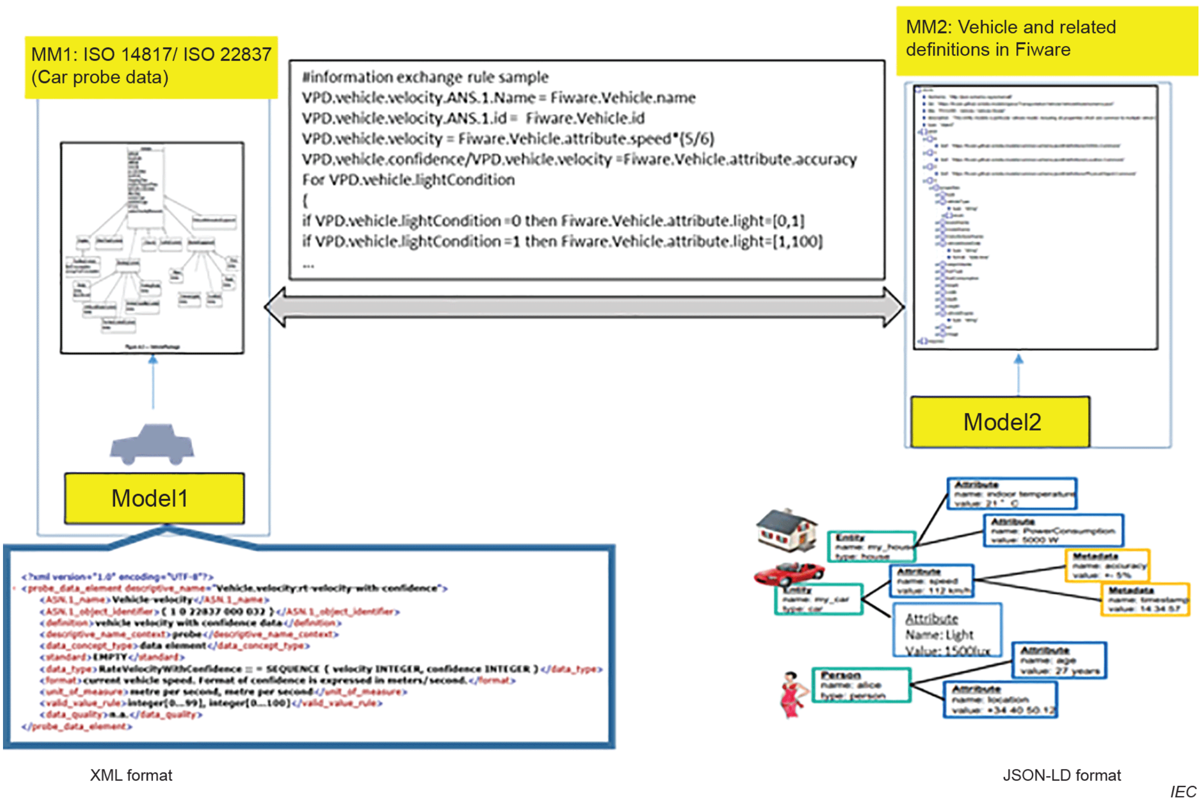
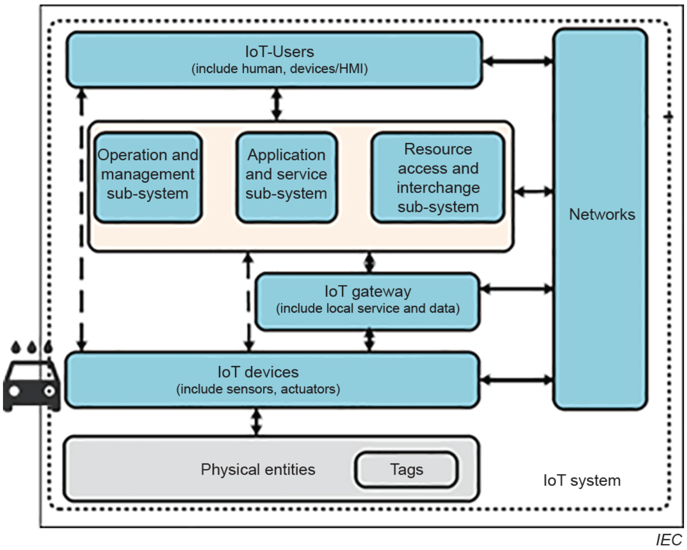


Figure B.1 – Overall view of use case 1

B.3 A scenario of this use case

B.3.1 The architecture of this use case

As shown in Figure B.2, based on Figure 11 of ISO/IEC 30141:2018 [32], a connected car can be recognized as one IoT device. It can share its probe data including environment data, etc. defined in ISO 22837 [30] with sub-systems that include vehicle and related definitions in the FIWARE data model. In this use case, one sub-system – traffic management system – can be used to communicate with a connected car.



NOTE Based on Figure 11 of ISO/IEC 30141:2018.

Figure B.2 – Architecture of connected car and vehicle in smart city use case

B.3.2 Scenario: Data exchange between a connected car and a traffic management system (TMS)

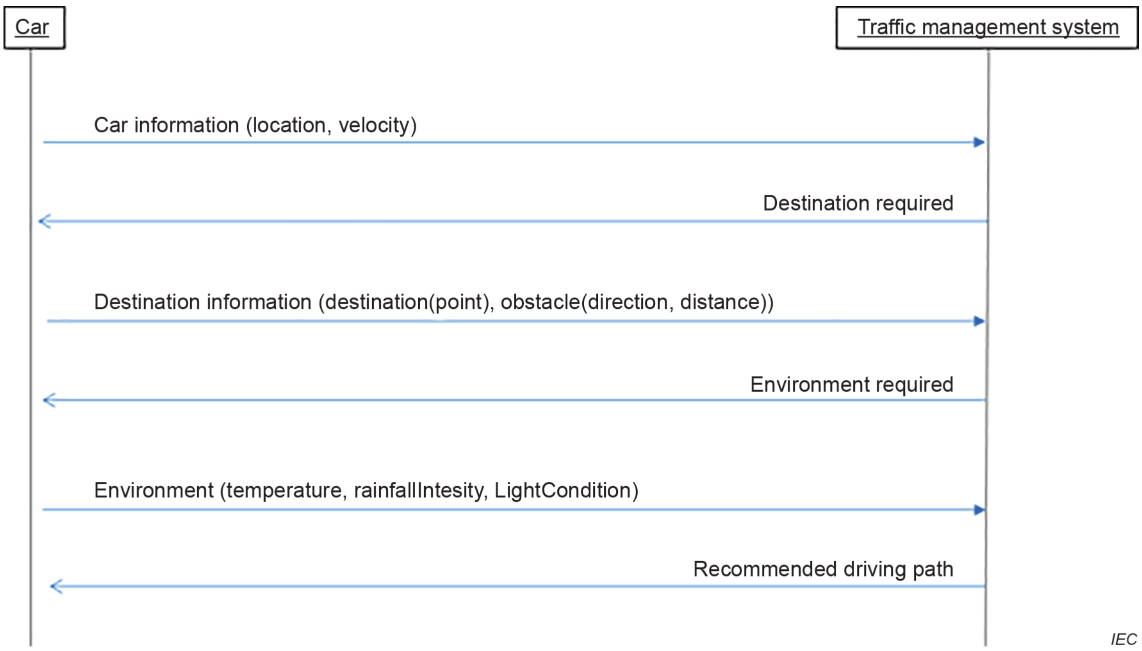


Figure B.3 – Information exchange between a car and a TMS

A scenario shown in Figure B.3 is described to explain the information exchange between a connected car and a TMS based on the architecture of Figure B.2.

– The car includes the probe data, and the traffic management system includes the vehicle and related definitions in the FIWARE data model. And the car can communicate with the TMS.
– This scenario is a simple use case for a car to avoid traffic congestion with a recommended driving path from the TMS.
– The purpose of this scenario is to help readers to grasp how the information exchange based on metamodels can be realized.

Details of this scenario are as follows.

a) The car sends to the TMS information such as its location and its velocity.
b) The TMS then requires the car to share its destination information.
c) The car sends to the TMS its destination information such as destination(point), its around obstacle information such as the direction, distance of the obstacle.
d) The TMS further requires the car to share its environmental information.
e) The car sends to the TMS its environment information such as its around temperature, rainfallIntensity, and lightCondition.
f) The TMS finally sends to the car a recommended driving path.

Through the above steps, the probe data of a car can be exchanged with the vehicle-related data of the TMS. The metamodels, models, and data of this scenario are included in the example of Clause B.2

B.4 Examples used in this use case

B.4.1 General

In this use case, MM1 and its model are described in the XML format. MM2 and its model are described with JSON. Files for this use case will be publicly available on https://github.com/21823-4/usecases/.

B.4.2 Illustrated example files and their relationships

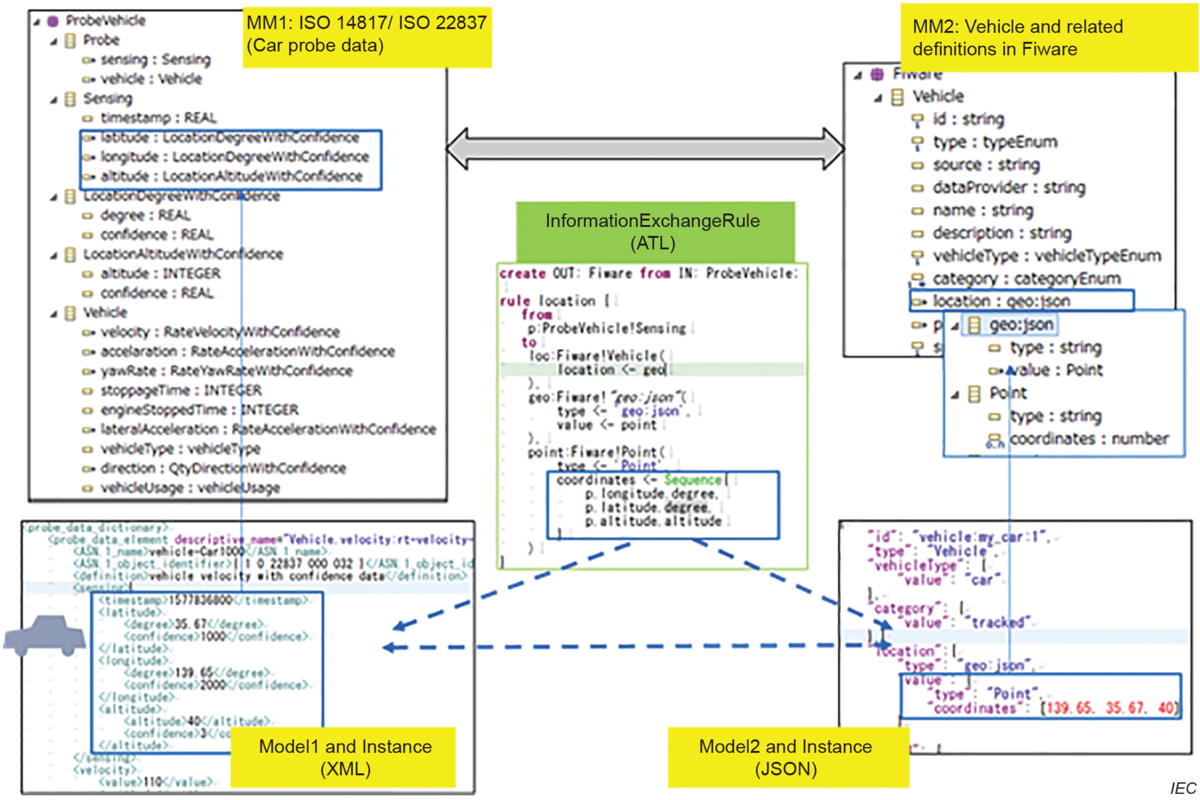


Figure B.4 – Relationships of example files for this use case

Figure B.4 shows relationships among MM1, Model1 with its sample instance, MM2, and Model2 with its sample instance.

- MM1 is the metamodel for probe data of connected cars.
- Model1 is an instance of MM1. It can define its sensing data for a location with "latitude", "longitude" and "altitude", as specified in MM1. Excerpted XML file for Model1 and its instance is listed on the left side of Figure B.4. The XML tags such as <latitude>, <degree> and <confidence> are from its metamodel.
- MM2 is the metamodel for vehicle and related definitions in FIWARE data models.
- Model2 is an instance of MM2. It can define its location with a geo:json format which needs to be described as (longitude, latitude, elevation) according to MM2 specifications. Excerpted JSON file for Model2 and its instance is listed on the right side of Figure B.4. The "key" of JSON files such as "location" and its value format must correspond to its metamodel.
- Excerpted information exchange rules between MM1 and MM2 are defined with ATL (Atlas Transformation Language [19]). The information exchange rules can be defined as bi-directional.
- With a rule interpreter, Model1 and its data can be transformed to Model2 and corresponding data, and vice versa.

In conclusion, the probe data of a car can be exchanged with the vehicle-related data of the TMS.

Annex C
(informative)

Other metamodel definitions

Several definitions of metamodel listed in ISO/IEC/IEEE 24765:2017 [11] are represented in Table C.1.

Table C.1 – Definitions of metamodel in various resources

	Definition	Resource
1	model that specifies one or more other models	ISO/IEC 11179-3:2013, *Information technology – Metadata registries (MDR) – Part 3: Registry metamodel and basic attributes*, 3.2.80
2	logical information model that specifies the modelling elements used within another (or the same) modelling notation	IEEE 1175.1-2002 (R2007), IEEE Guide for CASE Tool Interconnections-Classification and Description, 3.8
3	metamodel Vm for a subset of IDEFobject is a view of the constructs in the subset that is expressed using those constructs such that there exists a valid instance of Vm that is a description of Vm itself	IEEE 1320.2-1998 (R2004), IEEE Standard for Conceptual Modelling Language Syntax and Semantics for IDEF1X97 (IDEFobject), 3.1.111
4	model containing detailed definitions of the meta-entities, meta-relationships and meta-attributes whose instances appear in the model section of a CDIF transfer	ISO/IEC 15474-1:2002, *Information technology – CDIF framework – Part 1: Overview*, 4.2
5	specification of the concepts, relationships and rules that are used to define a methodology	ISO/IEC 24744:2014, *Software engineering – Metamodel for development methodologies*, 3.3
6	model defining the concepts and their relations for some modelling notation	ISO/IEC 15909-2:2011, *Systems and software engineering – High-level Petri nets – Part 2: Transfer format*, 4.1.6
7	special kind of model that specifies the abstract syntax of a modelling language	ISO/IEC 19506:2012, *Information technology – Object Management Group Architecture-Driven Modernization (ADM) – Knowledge Discovery Meta-Model (KDM)*

Bibliography

[1] The Internet Of Things: Mapping the value beyond the hype, McKinsey Global Institute, June 2015. Available at: https://www.mckinsey.com

[2] ISO/IEC 21823-1, *Internet of Things (IoT) – Interoperability for IoT systems – Part 1: Framework*

[3] ISO/IEC 21823-2, *Internet of Things (IoT) – Interoperability for IoT systems – Part 2: Transport interoperability*

[4] ISO/IEC 21823-3, *Internet of Things (IoT) – Interoperability for IoT systems – Part 3: Semantic interoperability*

[5] ISO 19103:2015, *Geographic information – Conceptual schema language*

[6] ISO/IEC 19506:2012, *Information technology – Object Management Group Architecture-Driven Modernization (ADM) – Knowledge Discovery Meta-Model (KDM)*

[7] ISO 19109:2015, *Geographic information – Rules for application schema*

[8] ISO 16484-5:2017, *Building automation and control systems (BACS) – Part 5: Data communication protocol*

[9] ISO/IEC 19941:2017, *Information technology – Cloud computing – Interoperability and portability*

[10] OMG® Unified Modeling Language® (OMG UML®), Version 2.5.1

[11] ISO/IEC/IEEE 24765:2017, *Systems and software engineering – Vocabulary*

[12] ISO 13584-32, *Industrial automation systems and integration – Parts library – Part 32: Implementation resources: OntoML: Product ontology markup language*

[13] Berger S., Grossmann G., Stumptner M. and Schrefl M. (2010) Metamodel-Based Information Integration at Industrial Scale. In: Petriu D.C., Rouquette N., Haugen Ø. (eds) Model Driven Engineering Languages and Systems. MODELS 2010., *Lecture Notes in Computer Science*, vol. 6395. Springer, Berlin, Heidelberg

[14] Haas L.M., Hentschel M., Kossmann D. and Miller R.J. (2009) Schema AND Data: A Holistic Approach to Mapping, Resolution and Fusion in Information Integration. In: Laender A.H.F., Castano S., Dayal U., Casati F., de Oliveira J.P.M. (eds), *Lecture Notes in Computer Science*, vol. 5829. Springer, Berlin, Heidelberg

[15] ISO/IEC 19502:2005, *Information technology – Meta Object Facility (MOF)*

[16] Ahmed A., Kleiner M. and Roucoules L. Model-based Interoperability IoT Hub for the Supervision of Smart Gas Distribution Networks. *IEEE Systems Journal*, IEEE, 2019, 13 (2), pp.1526-1533. hal-02267637

[17] Meta Object Facility (MOF) 2.0 Query/View/Transformation (QVT) Specification Version 1.3, https://www.omg.org/spec/QVT/1.3/PDF,(accessed 2021-10-04)

[18] Object Constraint Language Version 2.4, https://www.omg.org/spec/OCL/2.4/PDF (accessed 2021-10-04)

[19] ATL: Atlas Transformation Language – Specification of the ATL Virtual Machine, http://www.eclipse.org/atl/documentation/old/ATL_VMSpecification[v00.01].pdf (accessed 2020-07-09)

[20] Anjorin A., Leblebici E. and Schurr A. (2016). 20 Years of Triple Graph Grammars: A Roadmap for Future Research. 10.14279/tuj.eceasst.73.1031

[21] Anjorin A., Buchmann T., Westfechtel B., Diskin Z., Ko H-S., Eramo R., Hinkel G., Samimi L. and Zundorf A. (2019)., Benchmarking bidirectional transformations: theory, implementation, application, and assessment. *Software and Systems Modeling* 19, 647–691. 10.1007/s10270-019-00752-x

[22] FIWARE vehicle model: https://FIWARE-datamodels.readthedocs.io/en/latest/Transportation/Vehicle/VehicleModel/doc/spec/index.html (accessed 2020-07-09)

[23] FIWARE data model: https://github.com/FIWARE/data-models/blob/master/specs/guidelines.md (accessed 2020-07-09)

[24] Lewis D. (1983). "Extrinsic Properties". *Philosophical Studies* 44, 197–200. doi:10.1007/bf00354100. ISSN 0031-8116

[25] Marshall D. and Weatherson B., "Intrinsic vs. Extrinsic Properties", *The Stanford Encyclopedia of Philosophy* (Spring 2018 Edition), Edward N. Zalta (ed.), URL = <https://plato.stanford.edu/archives/spr2018/entries/intrinsic-extrinsic/>

[26] ISO/IEC 25012:2008, *Software engineering – Software product Quality Requirements and Evaluation (SQuaRE) – Data quality model*

[27] IEC 61360-1:2017, *Standard data element types with associated classification scheme – Part 1: Definitions – Principles and methods*

[28] Tirado-Andres F., Rozas, A. and Araujo, A. (2019). A Methodology for Choosing Time Synchronization Strategies for Wireless IoT Networks. *Sensors* 19, 3476. 10.3390/s19163476

[29] Hodgson R., Mekonnen D., Price D., Hodges J., Masters J.E., Cox S.J.D. and Ray S. Quantities, Units, Dimensions and Types (QUDT) Schema – Version 2.0. Technical report, qudt.org, Jan. 2017

[30] ISO 22837:2009, *Vehicle probe data for wide area communications*

[31] ISO 14817-1:2015, *Intelligent transport systems – ITS central data dictionaries – Part 1: Requirements for ITS data definitions*

[32] ISO/IEC 30141:2018, *Internet of Things (IoT) – Reference Architecture*

This Indian Standard has been developed from Doc No.: LITD 27 (18512).

Amendments Issued Since Publication

Amend No.	Date of Issue	Text Affected

BUREAU OF INDIAN STANDARDS

Headquarters:

Manak Bhavan, 9 Bahadur Shah Zafar Marg, New Delhi 110002
Telephones: 2323 0131, 2323 3375, 2323 9402 *Website*: www.bis.gov.in

Regional Offices:		*Telephones*
Central	: 601/A, Konnectus Tower -1, 6th Floor, DMRC Building, Bhavbhuti Marg, New Delhi 110002	2323 7617
Eastern	: 8th Floor, Plot No 7/7 & 7/8, CP Block, Sector V, Salt Lake, Kolkata, West Bengal 700091	2367 0012 2320 9474
Northern	: Plot No. 4-A, Sector 27-B, Madhya Marg, Chandigarh 160019	265 9930
Southern	: C.I.T. Campus, IV Cross Road, Taramani, Chennai 600113	2254 1442 2254 1216
Western	: Manakalya, 4th Floor, NTH Complex (W Sector), F-10, MIDC, Andheri (East), Mumbai 400093	283 25838

Branches : AHMEDABAD, BENGALURU, BHOPAL, BHUBANESHWAR, CHANDIGARH, CHENNAI, COIMBATORE, DEHRADUN, DELHI, FARIDABAD, GHAZIABAD, GUWAHATI, HARYNA, HUBLI, HYDERABAD, JAIPUR, JAMMU & KASHMIR, JAMSHEDPUR, KOCHI, KOLKATA, LUCKNOW, MADURAI, MUMBAI, NAGPUR, NOIDA, PARWANOO, PATNA, PUNE, RAIPUR, RAJKOT, SURAT, VIJAYAWADA.

Published by BIS, New Delhi