IEC 62656-8

Edition 1.0 2020-06

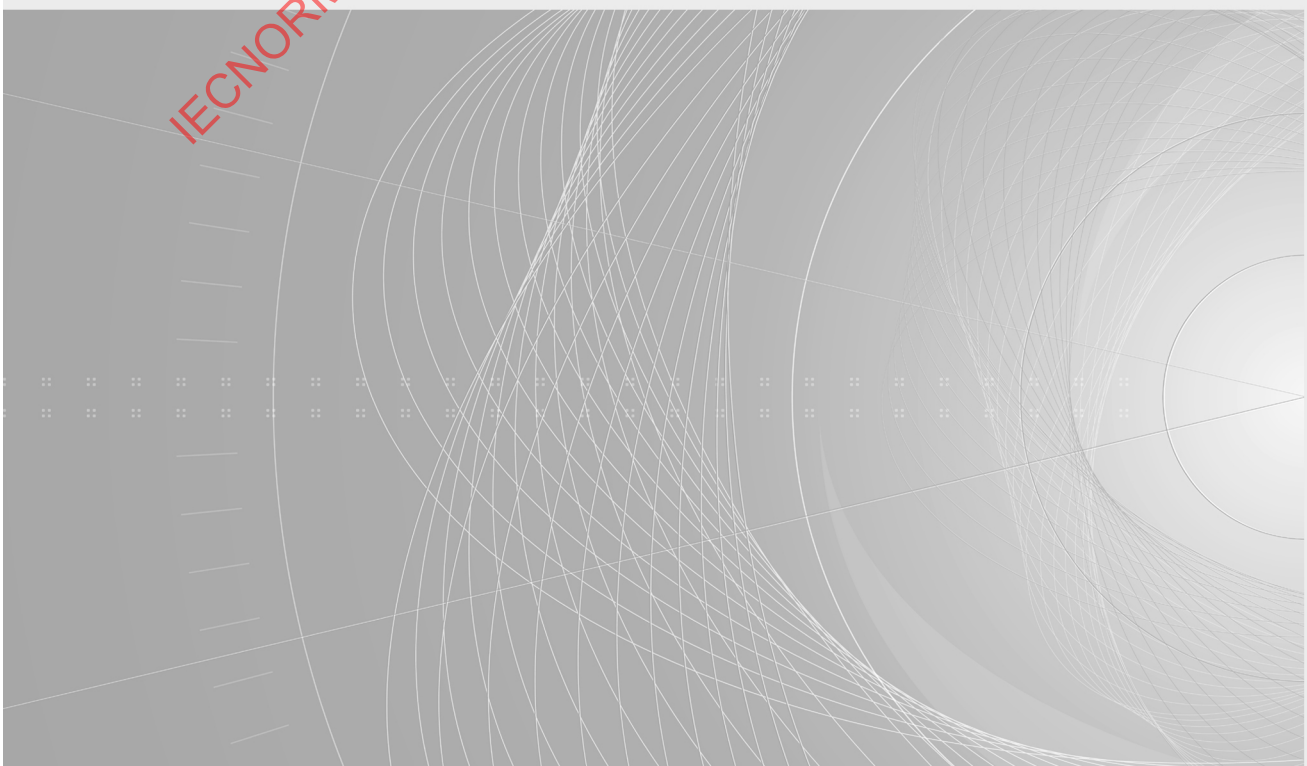
INTERNATIONAL STANDARD

NORME INTERNATIONALE

colour inside

Standardized product ontology register and transfer by data parcels – Part 8: Web service interface for data parcels

Enregistrement d'ontologie de produits normalises et transfert par paquets de données – Partie 8: Interface de service Web pour les paquets de données



IEC 62656-8:2020-06(en-fr)

IEC Central Office
3, rue de Varembé
CH-1211 Geneva 20
Switzerland

Tel.: +41 22 919 02 11
info@iec.ch
www.iec.ch

IEC 62656-8

Edition 1.0 2020-06

INTERNATIONAL STANDARD

NORME INTERNATIONALE

colour inside

Standardized product ontology register and transfer by data parcels – Part 8: Web service interface for data parcels

Enregistrement d'ontologie de produits normalises et transfert par paquets de données – Partie 8: Interface de service Web pour les paquets de données

INTERNATIONAL ELECTROTECHNICAL COMMISSION

COMMISSION ELECTROTECHNIQUE INTERNATIONALE

ICS 01.040.01; 01.110

ISBN 978-2-8322-8469-8

CONTENTS

INTERNATIONAL ELECTROTECHNICAL COMMISSION

_____

STANDARDIZED PRODUCT ONTOLOGY REGISTER AND TRANSFER BY DATA PARCELS –

Part 8: Web service interface for data parcels

FOREWORD

1) The International Electrotechnical Commission (IEC) is a worldwide organization for standardization comprising all national electrotechnical committees (IEC National Committees). The object of IEC is to promote international co-operation on all questions concerning standardization in the electrical and electronic fields. To this end and in addition to other activities, IEC publishes International Standards, Technical Specifications, Technical Reports, Publicly Available Specifications (PAS) and Guides (hereafter referred to as "IEC Publication(s)"). Their preparation is entrusted to technical committees; any IEC National Committee interested in the subject dealt with may participate in this preparatory work. International, governmental and non-governmental organizations liaising with the IEC also participate in this preparation. IEC collaborates closely with the International Organization for Standardization (ISO) in accordance with conditions determined by agreement between the two organizations.
2) The formal decisions or agreements of IEC on technical matters express, as nearly as possible, an international consensus of opinion on the relevant subjects since each technical committee has representation from all interested IEC National Committees.
3) IEC Publications have the form of recommendations for international use and are accepted by IEC National Committees in that sense. While all reasonable efforts are made to ensure that the technical content of IEC Publications is accurate, IEC cannot be held responsible for the way in which they are used or for any misinterpretation by any end user.
4) In order to promote international uniformity, IEC National Committees undertake to apply IEC Publications transparently to the maximum extent possible in their national and regional publications. Any divergence between any IEC Publication and the corresponding national or regional publication shall be clearly indicated in the latter.
5) IEC itself does not provide any attestation of conformity. Independent certification bodies provide conformity assessment services and, in some areas, access to IEC marks of conformity. IEC is not responsible for any services carried out by independent certification bodies.
6) All users should ensure that they have the latest edition of this publication.
7) No liability shall attach to IEC or its directors, employees, servants or agents including individual experts and members of its technical committees and IEC National Committees for any personal injury, property damage or other damage of any nature whatsoever, whether direct or indirect, or for costs (including legal fees) and expenses arising out of the publication, use of, or reliance upon, this IEC Publication or any other IEC Publications.
8) Attention is drawn to the Normative references cited in this publication. Use of the referenced publications is indispensable for the correct application of this publication.
9) Attention is drawn to the possibility that some of the elements of this IEC Publication may be the subject of patent rights. IEC shall not be held responsible for identifying any or all such patent rights.

International Standard IEC 62656-8 has been prepared by subcommittee 3D: Classes, Properties and Identification of products – Common Data Dictionary (CDD), of IEC technical committee 3: Documentation, graphical symbols and representations of technical information.

The text of this International Standard is based on the following documents:

FDIS	Report on voting
3D/342/FDIS	3D/346/RVD

Full information on the voting for the approval of this International Standard can be found in the report on voting indicated in the above table.

This document has been drafted in accordance with the ISO/IEC Directives, Part 2.

A list of all parts in the IEC 62656 series, published under the general title *Standardized product ontology register and transfer by data parcels*, can be found on the IEC website.

Future standards in this series will carry the new general title as cited above. Titles of existing standards in this series will be updated at the time of the next edition.

The committee has decided that the contents of this document will remain unchanged until the stability date indicated on the IEC website under "http://webstore.iec.ch" in the data related to the specific document. At this date, the document will be

- reconfirmed,
- withdrawn,
- replaced by a revised edition, or
- amended.

IMPORTANT – The 'colour inside' logo on the cover page of this publication indicates that it contains colours which are considered to be useful for the correct understanding of its contents. Users should therefore print this document using a colour printer.

INTRODUCTION

For a description of products and services throughout their lifecycle, an enhanced data interoperability with reduced human interventions is an ultimate goal of developing international standards for intelligent production systems. In attaining this goal, an industrial ontology is expected to play a significant role by allowing components of systems to talk to each other, namely machine-machine understanding, about their functions, capabilities, structures and their configurations.

The parcellized ontology model defined in IEC 62656-1, also known by its acronym "POM", is a generic ontology model with quadruple layers to capture different types of ontology models by sorting elements into categories of homogeneous collection of ontological entities, such as classes (concepts), properties, relations, enumerations, terms (constants), data types, etc. At the second layer from the top, named the Meta-Ontology (MO) layer, 11 types of category are defined. Each layer is a collection of categories, while each category is represented by a relational table-like matrix called "data parcel" whose meta data (attributes) are embodied as a selection of instances of the immediate upper layer. The top layer of the POM, named the Axiomatic Ontology (AO) layer, comprises two data parcels only, which conjointly define the "concept of concepts" by classes and properties, which is an information technology (IT) embodiment of the math-logical notion of the class (i.e., "concept") itself.

Other parts of the IEC 62656 series, which are collectively known as "Parcel standards", are intended as a specialization of the POM for a specific purpose.

IEC 62656-2 [1][1] is a guide for domain experts to apply the POM in order to capture a data dictionary from the definitions available in product standards in a form conformant to the IEC 61360-2 [2] and ISO 13584-42 [3] dictionary schema (i.e., common data dictionary model, or CDDM) and using the specification of a part of IEC 62656-1 as an official data interface for the IEC 61360-4 database known as the IEC CDD (Common Data Dictionary), enabling the uploading and downloading of the dictionary to and from the IEC CDD. A referential implementation of IEC 62656-1 is available as a tool, free of charge for standardization purposes.

IEC 62656-3 is intended as a mapping specification between a standard data model of the "Smart-Grid" domain, with acronym CIM (Common Information Model), and an extended, or rather generalized data model of the IEC CDD, namely, the POM. The CIM comprises the IEC 61968/IEC 61970/IEC 62325 series of International Standards. Thus, the IEC CDD can accommodate the CIM provided the IEC CDD sufficiently implements the POM as the data interface or database. Alternatively, this mapping inevitably entails a small but significant extension of the IEC CDD, without which the accommodation of the CIM into the IEC CDD is infeasible. Nevertheless, nothing needs to be added to or subtracted from the tool which is currently used as a data interface for the IEC CDD and which fully embodies IEC 62656-1.

IEC 62656-5 is intended as an interface for the description of activities as an ontology conformant to IEC 62656-1, thus opening a way to store definitions available from activity-centric International Standards, for instance IEC 62224-3, as an ontology. IEC 62656-5 can also be applied to the description of non-manufacturing use scenarios, such as for the description of activities of natural hazard management or electronic tourist guidance or navigation, with a harmonious integration of activities with related products and services.

This means a common ontology repository ("COR") based on the POM can store both IEC CDD and CIM types of data dictionaries or ontologies. Furthermore it can smoothly bridge the differences and fill the gaps covering ontologies of different provenances.

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

Future parts of the IEC 62656 series are expected to shed light on a new spectrum of applications for the COR based on the POM.

Above all, this document specifies a description of basic web services for semantic repositories based on the POM, whilst an advanced type of web interface, including complex enquiry about products as well as query forwarding to another repository, is left to a future part of the series, to be developed.

STANDARDIZED PRODUCT ONTOLOGY REGISTER AND TRANSFER BY DATA PARCELS –

Part 8: Web service interface for data parcels

1 Scope

This part of IEC 62656 specifies a web service interface to exchange data parcel(s) conformant to IEC 62656-1, between a parcel server and a parcel client or between parcel servers. This interface comprises three basic services: a registration service, resolution service and subscription service.

This document includes the following:

– holistic use scenario;
– detailed specification of the three basic services;
– JSON [1] and XML [5] notation schemas for data parcel(s).

The following items are outside the scope of this document:

– user identification and authorization;
– query language for a data parcel;
– transportation protocol;
– data and communication security techniques.

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.

IEC 62656-1:2014, *Standardized product ontology register and transfer by spreadsheets – Part 1: Logical structure for data parcels*

ISO/IEC 21778, *Information technology – The JSON data interchange syntax*

ISO 639-1, *Codes for the representation of names of languages – Part 1: Alpha-2 code*

ISO 3166-1, *Codes for the representation of names of countries and their subdivisions – Part 1: Country codes*

ISO 8601-1, *Date and time – Representations for information interchange – Part 1: Basic rules*

ISO 8601-2, *Date and time – Representations for information interchange – Part 2: Extensions*

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

3 Terms, definitions and abbreviated terms

3.1 Terms and definitions

For the purposes of this document, the following terms and definitions apply.

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

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

3.1.1
application ontology repository
AOR
ontology repository where proprietary ontologies and their instances can be stored

Note 1 to entry: A proprietary ontology can be a modification and/or an extension of a standardized ontology.

Note 2 to entry: Standardized ontologies can be also stored in an AOR.

3.1.2
common ontology repository
COR
shared ontology repository where standardized ontologies of different base models can be stored

3.1.3
conjunctive parcels
parcel sheets that are used together to define a library, reference dictionary, or meta-dictionary

[SOURCE: IEC 62656-1:2014, 3.6]

3.1.4
data parcel
parcel
information structure in a form of a level-pair, comprising a set of properties and a set of tuples of values for the set of properties, with an aim to describe a domain data dictionary, a domain data library or an ontological modelling concept

Note 1 to entry: A data parcel is typically implemented and exchanged as a set of spreadsheets, but the medium of implementation or exchange is not limited to spreadsheets; it may be in any other form.

[SOURCE: IEC 62656-1:2014, 3.8]

3.1.5
dictionary
data dictionary
set of terms with respective identifiers formulated in a canonical syntax and with commonly accepted definitions designed to yield a lexical or taxonomical framework for knowledge representation in a computer interpretable form, which can be shared by different information systems and communities

[SOURCE: IEC 62656-1:2014, 3.10]

3.1.6
JSON data
data represented in compliance with the ISO/IEC 21778 specification

3.1.7
JSON name
series of characters assigned to a value or object for referring to it in JSON data

3.1.8
ontological entity
artefact that is used to represent a category of being of things or relationship among them

[SOURCE: IEC 62656-1:2014, 3.36]

3.1.9
ontology repository
data repository where ontologies or ontological entities are stored

3.1.10
parcel client
client system or application that can read or write parcelling sheets in general, and may have an optional capability to send them to or receive them from a server system

[SOURCE: IEC 62656-1:2014, 3.37]

3.1.11
parcel ontology layer
abstraction layer which is embodied as a set of data parcels on the same level

Note 1 to entry: IEC 62656-1:2014 classifies parcel ontology layers as axiomatic ontology (AO), meta ontology (MO), domain ontology (DO) and domain library (DL).

3.1.12
parcel server
server system or application that can provide parcel spreadsheets in general over Internet

[SOURCE: IEC 62656-1:2014, 3.41]

3.1.13
parcel registration
operation to enter new record of information by data parcels to a parcel server

Note 1 to entry: Registration operations are classified into addition, modification and deletion operations.

3.1.14
parcel resolution
operation to receive information by data parcels by using the specific parameter(s) as search condition(s)

3.1.15
parcel subscription
agreement to receive the status or the change of the information by data parcels

3.2 Abbreviated terms

AOR Application Ontology Repository
CDD Common Data Dictionary
COR Common Ontology Repository
HTTP Hypertext Transfer Protocol
HTTPS Hypertext Transfer Protocol Secure
ICID International Concept Identifier
JSON JavaScript Object Notation
RAI Registration Authority Identifier
SOAP Simple Object Access Protocol
VI Version Identifier
XML eXtensible Markup Language
WADL Web Application Description Language
WSDL Web Services Description Language

4 Use scenarios

4.1 Holistic use scenario

A standardized ontology such as that set out in IEC 61360-4 consists of a standardized set of classes and properties which should be used for common understanding of the meaning of data among users. In most cases, such a standardized ontology contains a minimum agreed set of classes and properties expressed in the English language; therefore it usually needs to be customized for actual use. For example, language translation may be important for the use of an ontology for non-native English users in a regional company. For another example, extensions of classes and properties, such as commercial data or administrative information, are essential for product information management in each company. The four modelling layers approach defined in IEC 62656-1 allows for the representation of such an extension and modification in data parcels.

Such a customized ontology is usually maintained and published in a server (e.g., application ontology repository) such as a national server and an enterprise server, separately from the server (e.g., common ontology repository) where the standardized ontology is stored. If there is a mechanism not only for downloading or uploading data parcels but also for subscription to an ontology by using the web technique, the use of a standardized ontology and its extension will be enhanced and made easy. Web services specified in this document will contribute to enabling the exchange of data parcel(s) of an ontology between a server and a client or between servers.

In many cases, data exchange both between servers and between a server and a client are done over the HTTP(S) protocol, because there are many hardware, software and security techniques for it. Therefore, the use of the HTTP(S) protocol is recommended for a parcel web service, but other protocols are also available for a data exchange by means of a parcel web service for a specific requirement such as communication speed.

The web services for the exchange of or the subscription to data parcels defined in this document presume, but are not limited to, the following use scenarios depicted in Figure 1.

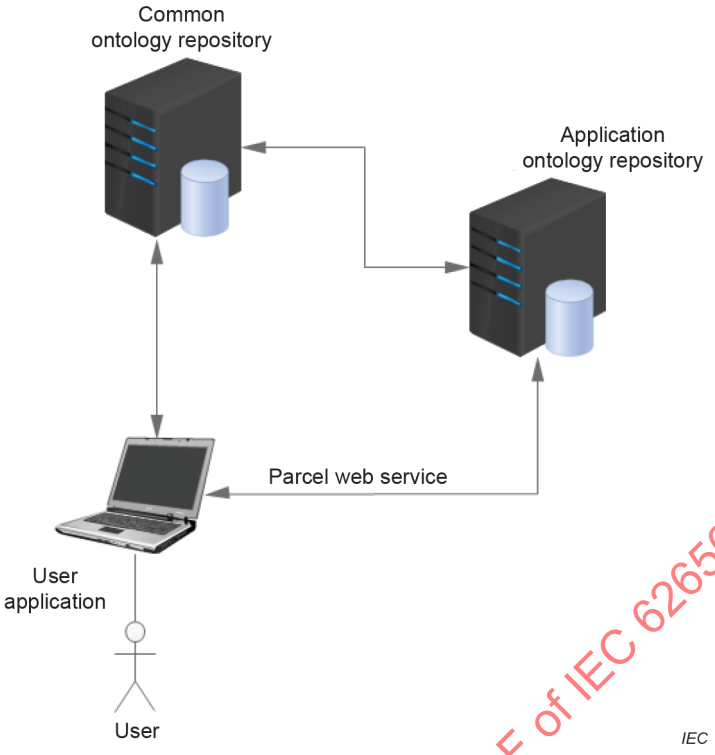


Figure 1 – Holistic use scenario of parcel web services

4.2 Use scenario between server and client

A detailed use scenario between an ontology repository and a parcel client is depicted in Figure 2 with the following activities and associated information flows:

– A1: A client application, such as an ontology editor, registers the content of a set of data parcels into an ontology repository without going through an operation on the web page of the ontology repository.

– A2: A client application, such as a controller, CAD application or purchasing system, requests to resolve the identifier of a dictionary item and obtains a part, or the whole set, of information about a dictionary item designated by the identifier without going through an operation on the web page of an ontology repository.

NOTE Content delivered by a set of data parcels is a data dictionary or an update of a data dictionary.

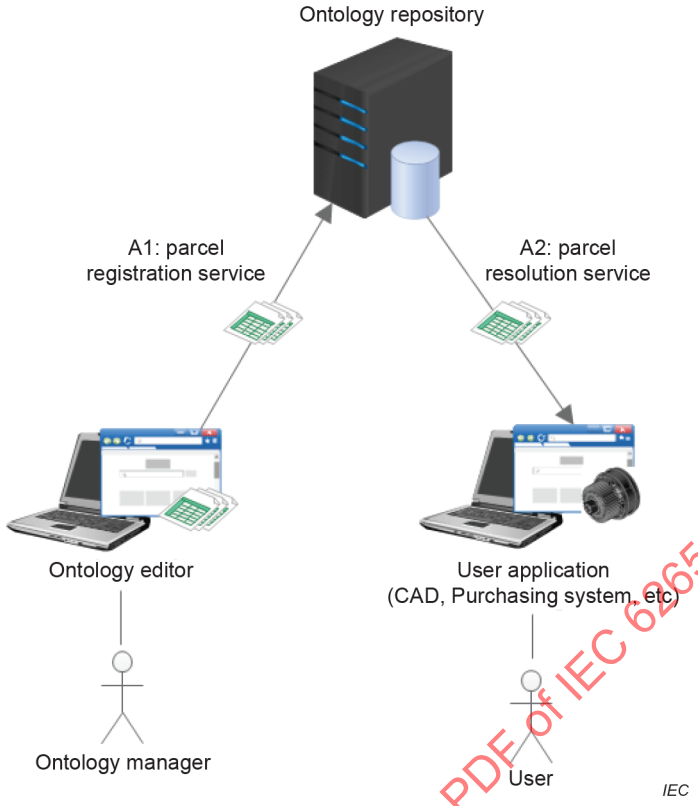


Figure 2 – Parcel resolution and registration services between a server and a client

4.3 Use scenario between servers

A detailed use scenario between servers is depicted in Figure 3 with the following activities and associated information flows:

– A3: An application ontology repository requests to subscribe to a dictionary item stored in a common ontology repository or another application ontology repository to obtain an update of the content stored in the latter repository when a new instance is registered in the latter repository.

– A4: An application ontology repository that subscribes to a dictionary item stored in the common ontology repository obtains an update of the content on a regular, specified time-span basis or on an event-driven basis, i.e., when a new instance is registered in the latter repository. Ontology repository servers can make a cascade connection to deliver the content from an upstream server to a downstream server.

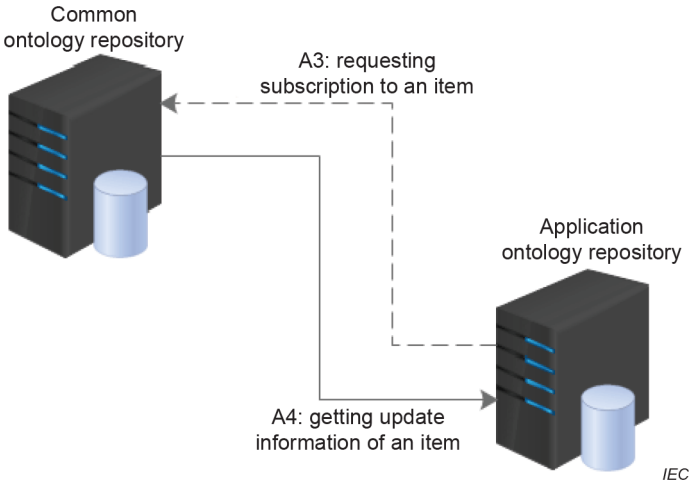


Figure 3 – Parcel subscription service between registries

5 Parcel web service specification

5.1 General

This document specifies the following three basic web services, upon which more sophisticated or advanced user applications can be built:

– parcel registration service (see 5.4);
– parcel resolution service (see 5.5);
– parcel subscription service (see 5.6).

Like other web services in general, each basic web service specified in this document also has means for handling requests, responses and exceptions.

A request is a message which is sent from a parcel client to a parcel server. The message may contain one or more parameters, each of which is used for executing a service on a parcel server.

A response is a message which is sent from a parcel server to a parcel client as a result of executing a service. The message may contain the result of the inquiry from a parcel client.

An exception is an unexpected event which occurs during the executing of a parcel web service. If such an event occurs on a parcel server, and if information about the event is helpful or meaningful for a parcel client, then an exceptional message will be thrown from the parcel server to the parcel client. This allows a parcel client to continue its flow of functions when a service fails. If an exception occurs during the processing of a function which a parcel client requests to execute, the execution will be terminated and the parcel server will throw the exception to the parcel client.

For a machine-readable description of web services, WADL (Web Application Description Language) [6] and WSDL (Web Service Description Language) [7] are widely used. A WADL description for the services specified in Clause 5 is provided in Annex B, Clause B.1, while a WSDL description for them is provided in Clause B.2.

5.2 Exception

5.2.1 General

As mentioned in 5.1, an exception is an unexpected event which occurs during the executing of a parcel web service.

This document specifies the following types of exceptions for an unexpected event:

– Standard-defined exception;
– User-defined exception.

"Standard-defined exception" is the super type of exception which is defined in this document. The specification of those exceptions is provided in 5.2.3.

"User-defined exception" is the super type of exception to be used for defining exceptions which are defined in user applications for a specific purpose. For a parcel web service between parcel servers or between a parcel server and a parcel client, the use of a user-defined exception is allowed, but it is possible that it will not be handled by a receiver of such an exception. In other words, such an exception may be recognized by a receiver simply as text information.

Figure 4 shows a structure of exceptions which consists of the general exceptions and exceptions which are defined in Table 1.

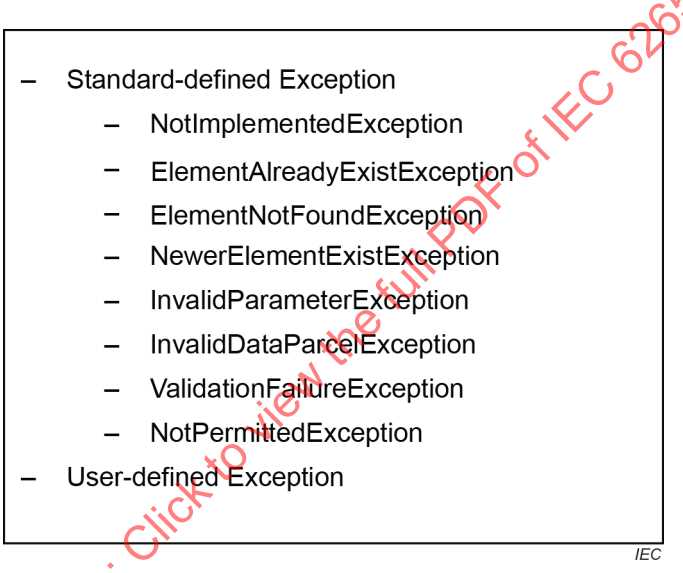


Figure 4 – Tree structure of exceptions

5.2.2 Naming convention for an exception

For the robust implementation of a user-defined exception on both a parcel server and a parcel client, it is necessary to specify what characters can be used for naming the exception. As naming conventions, the following rules shall apply for naming user-defined exceptions:

– Each value shall contain only alphabetic or numeric characters;
– Each value shall start with a capital alphabetic letter;
– Each value shall end with "Exception".

5.2.3 Standard-defined exceptions

Standard-defined exceptions which may be thrown or returned from a parcel server to a parcel client through a parcel web service are depicted in Table 1.

The name and definition of each exception is given for human understanding. The preferred letter symbol of each exception, which follows the naming convention specified in 5.2.2, is given for an implementation which can be used as a label of the exception in a programming language.

Table 1 – Standard-defined exceptions for parcel web services

Preferred name	Definition	Preferred letter symbol
no implementation	standard-defined exception indicating the specified function is not implemented on the ontology repository server	NotImplementedException
element already existing	standard-defined exception indicating the specified function is not executed because the same element already exists in the ontology repository	ElementAlreadyExistException
no element found	standard-defined exception indicating the specified request does not match any element in the ontology repository	ElementNotFoundException
newer element existing	standard-defined exception indicating a newer version of an element has been found in the ontology repository	NewerElementExistException
invalid parameter	standard-defined exception indicating an unexpected parameter has been specified in the request message	InvalidParameterException
unexpected value	standard-defined exception indicating an unexpected value has been inputted in a parameter of the request message	InvalidDataParcelException
validation failure	standard-defined exception indicating one or more errors occurring during a validation process on the specified data parcels	ValidationFailureException
no permission	standard-defined exception indicating the execution of the specified function is not allowed to the user	NotPermittedException

5.3 Search scope

In accordance with the parcel resolution service specified in 5.5, if there is a line which matches an identifier or a name specified for the parameter "keyword" of a request message, the line will be returned. For example, if the identifier of a class is specified in a request, then the definition line of this class will be returned.

Similar to the parcel resolution service, in accordance with the parcel subscription service specified in 5.6, if there is an item which matches an identifier specified for the parameter "identifiers" of a request message, the item will be subscribed or unsubscribed to. For example, if the identifier of a class is specified in a request, then the class will be subscribed or unsubscribed to. If a subclass is newly added to the class, i.e., the structure of the branch is changed, the information of the class itself is not changed because the information of the "is-a" relationship is usually described on the subclass side by using the attribute MDC_P010 (superclass). As a result, it is possible that subscribers to the class will not be informed of the change.

In order to simultaneously subscribe/unsubscribe to, or to obtain both the information of a class and information of its superclasses or subclasses in one request, this document specifies a search scope modifier for the request parameter "keyword" of the parcel resolution service. Table 2 shows the specification of those modifiers.

Table 2 – Specification of search scope modifiers

Search scope modifier	Description	Example
*	This modifier extends the scope to all the subclasses of a given class, including classes importing properties from the given class or its subclasses.	AAA004* 0112/2///61360_4#AAA004##001*
$	This modifier extends the scope to all direct subclasses. The relationship is not taken into account.	AAA004$ 0112/2///61360_4#AAA004##001$
%	This modifier extends the scope to all superclasses, including classes from which those subclasses import properties.	AAA004% 0112/2///61360_4#AAA004##001%
!	This modifier extends the scope to all direct superclasses only.	AAA004! 0112/2///61360_4#AAA004##001!

Figure 5 shows how each search scope modifier works to extend a search scope by using a classification tree. If AAA004 of the parameter "keyword" is specified in a request message of a parcel resolution service, then the line corresponding to AAA004 will be returned. If AAA004% or AAA004! of the parameter "keyword" is specified, then lines corresponding to AAA004 or identifiers of its superclasses, i.e., AAA001, AAA002 and AAA003, will be returned. If AAA004* or AAA004$ of the parameter "keyword" is specified, then lines corresponding to AAA004 or identifiers of its subclasses, i.e., AAA005, AAA006, AAA007, AAA008, AAA009 and AAA010, will be returned.

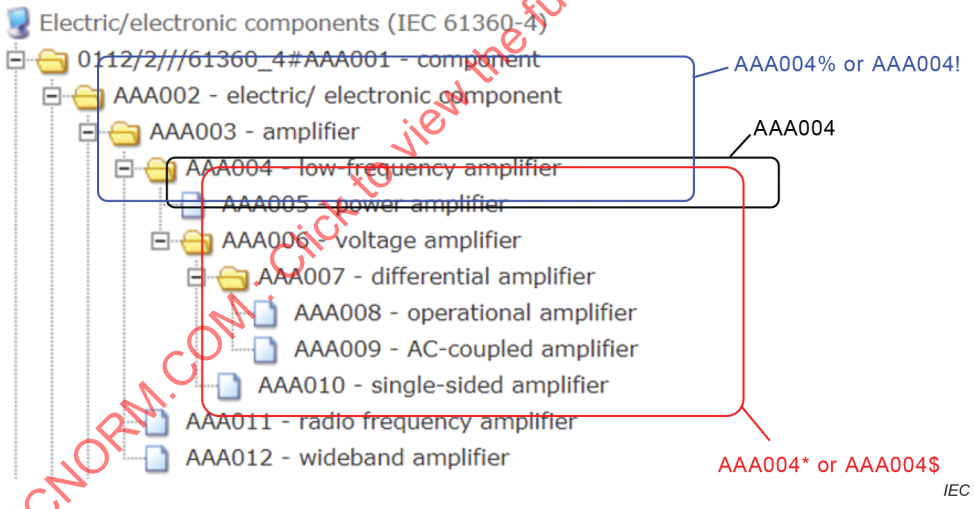


Figure 5 – Example of structural view of the use of search scope modifiers

Figure 6 shows an example of a class sheet of the class tree depicted in Figure 5. Each solid box with a label of an identifier with or without a search scope modifier indicates what lines will be returned by specifying the identifier with or without the search scope modifier at the parameter "keyword" in a request message of a parcel resolution service.

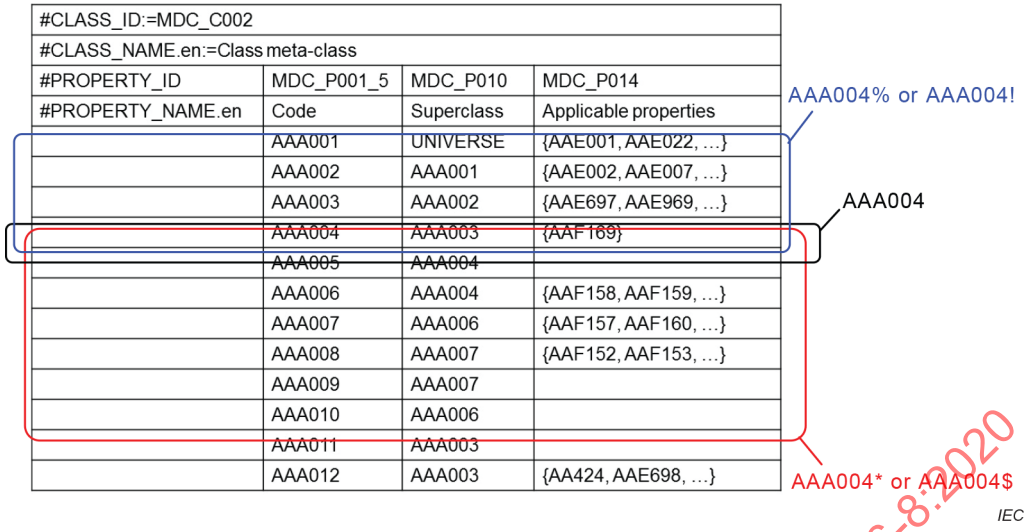


Figure 6 – Example of a parcel sheet view of the use of search scope modifiers

5.4 Parcel registration service

5.4.1 General

The parcel registration service is a service enabling a parcel client to register items in data parcels to a parcel server. When a parcel server receives a request from a parcel client which has an authority for registration, the parcel server checks if all items in data parcels in the request are valid or not and then updates the content in its database. If an error occurs during execution of a request from a client, for example, if the parcel server detects one or more invalid items, the parcel server should terminate the execution of the request and then throw an exception to the client, with some detailed information about the errors. In such a case, no item in the data parcels in the request is added, updated or deleted on the parcel server.

The parcel registration service is used for registering a whole ontology from scratch to a parcel server. It is also used for changing the existing content of an ontology registered in a parcel server. A change of content is any of adding a new item, modifying the existing item or deleting the existing item.

5.4.2 Request message

5.4.2.1 General

The request message of the parcel registration service shall contain "conjunctiveParcels" as its parameter. The parameter has a string value which consists of a conjunctive set of data parcels in JSON or XML, each instance of which is requested to be registered to a parcel server. Table 3 shows the summary of its parameter.

Table 3 – Structure of a request message of the parcel registration service

Name	Data type	Obligation	Example
conjunctiveParcels	String	Mandatory	<?xml version="1.0" encoding="utf-8"?> <conjunctiveParcels ontoLayer="DO"> <parcel> ... </parcel> </conjunctiveParcels>

5.4.2.2 Requirements for input data parcels

5.4.2.2.1 General

The instruction #PARCEL_MODE specified in IEC 62656-1 shall be specified in all data parcels which are sent from a parcel client to a parcel server through a parcel registration service.

With respect to the specification of #PARCEL_MODE, FULL and UPDATE assume that the result of the operation contains a complete set of data parcels to validate as an ontology, while PARTIAL by default assumes the result of the operation is an incomplete set of data parcels. Thus, if validation takes place, the result is usually an error.

The usage of the instructions in the parcel web services is as follows:

– Either FULL, UPDATE or PARTIAL may be assigned as the value for this instruction.
– Within a conjunctive set of data parcels, the value of the instruction #PARCEL_MODE of each parcel shall be consistent.
– If FULL or UPDATE is specified, a validation should be executed on the server.

If UPDATE or PARTIAL is specified for #PARCEL_MODE, one of the following three operations shall be used for each instance in a data parcel:

– #ADD (see 5.4.2.2.2);
– #MOD (see 5.4.2.2.3);
– #DEL (see 5.4.2.2.4).

If FULL is specified for #PARCEL_MODE, the use of the above three operations is not required. If those operations appear in a data parcel of the FULL mode, they shall be ignored.

5.4.2.2.2 Addition operation

Keyword: #ADD
Name: Addition operation
Definition: operation for the UPDATE and PARTIAL modes indicating that an instance is intended to be newly added to the content stored in a parcel server
Description: In the UPDATE mode, a parcel server shall check the duplication of the instance with the value(s) of the (composite) key or the data object identifier identified with MDC_P066, before updating the content. If duplication is found, the parcel server will reject the request and then return the standard-defined exception "ValidationFailureException" to the parcel client. If there is an inconsistency in the configuration of the (composite) key between the data parcels uploaded through the service and the content stored in the parcel server, the parcel server shall return the standard-defined exception "ValidationFailureException", containing a list of identifiers of items for which the validation failed, to the parcel client.

5.4.2.2.3 Modification operation

Keyword: #MOD
Name: Modification operation
Definition: operation for the UPDATE and PARTIAL modes indicating that the instance of the content stored in a parcel server, which corresponds to the value(s) of the (composite) key or the data object identifier identified with MDC_P066, is intended to be modified
Description: In the UPDATE mode, a parcel server shall check the existence of the instance before executing the process for updating it. If it is not found, the parcel server will reject the request and then return the standard-defined exception "ValidationFailureException" to the parcel client. If there is inconsistency in the configuration of the (composite) key between the data parcels uploaded through the service and the content stored in the parcel server, the parcel server shall return the standard-defined exception "ValidationFailureException", containing a list of identifiers of items for which the validation failed, to the parcel client.

5.4.2.2.4 Deletion operation

Keyword: #DEL
Name: Deletion operation
Definition: operation for the UPDATE and PARTIAL modes indicating that the instance of the content stored on a parcel server, which corresponds to the value(s) of the (composite) key or the data object identifier identified with MDC_P066, is intended to be deleted
Description: In the UPDATE mode, a parcel server shall check the existence of the instance before executing deleting it. If it is not found, the parcel server will reject the request and then return the standard-defined exception "ValidationFailureException" to the parcel client. An instance with this operation does not need to have values either of the (composite) key value(s) or of the data object identifier identified with MDC_P066. If an instance to be deleted has neither the (composite) key value(s) nor the data object identifier, it will be ignored by the execution process of the parcel resolution service. If there is inconsistency in the configuration of the (composite) key between the data parcels uploaded through the service and the content stored on the parcel server, the parcel server shall return the standard-defined exception "ValidationFailureException", containing a list of identifiers of items for which the validation failed, to the parcel client.

5.4.3 Response message

The response message of the parcel registration service shall contain "operationResult" as its parameter. The parameter has a Boolean value indicating whether the execution of a registration request has succeeded or not. If an execution succeeds, the ontology repository will return a "true" indication to the parcel client. If an execution fails, one of the exceptions will be returned to the parcel client, in addition to returning a "false" indication. Table 4 shows the summary of the structure of a response message.

Table 4 – Structure of a response message of the parcel registration service

Name	Data type	Obligation	Example
operationResult	Boolean	Mandatory	true

5.4.4 Exception

The following standard-defined exceptions specified in 5.2 will apply to the parcel registration service:

– no implementation,
– element already existing for the #ADD operation,
– no element found for the #MOD and #DEL operations,
– newer version element existing,
– unexpected parameter,
– unexpected value,
– validation failure,
– no permission.

5.5 Parcel resolution service

5.5.1 General

The parcel resolution service is a service enabling a parcel client to resolve ontological entities which are stored on a parcel server. A resolution on a parcel server is done by using identifiers or names specified in a request message from a parcel client. If there are ontological entities which match those identifiers or names on a parcel server, the parcel server will return those ontological entities in an XML or JSON form conformant to this document. If an error occurs during an execution of a request from a client, the parcel server should terminate the execution of the request and then throw an exception to the client, with some detailed information about the errors. In such a case, no content will be returned to the parcel client.

The parcel resolution service provides two kinds of resolutions for specifying an identifier of an ontological entity, as follows:

– definition resolution, for resolving definitions of ontological entities which correspond to identifiers or names specified in a request message.
– instance resolution, for resolving instances of ontological entities which correspond to identifiers or names specified in a request message.

The kind of resolution selected is determined by a parameter value of the request message.

EXAMPLE 1 In a definition resolution, if the identifier of a class is specified in a request message, all information of the class, such as code, name and definition, will be returned. Such information consists of pairs of a class attribute defined in IEC 62656-1 and its value.

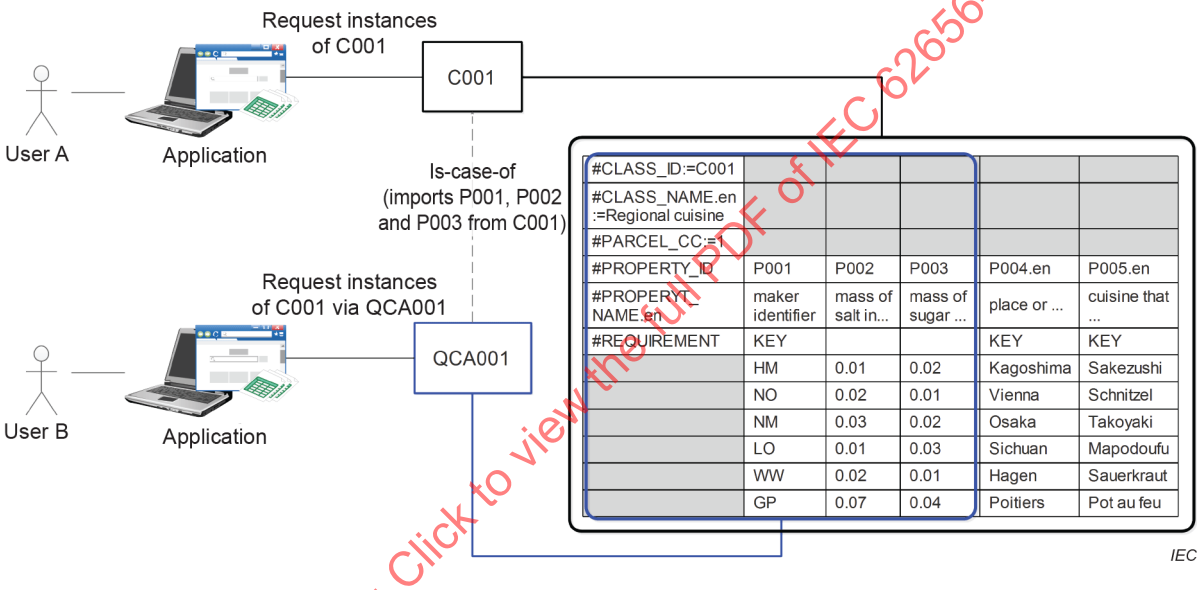
EXAMPLE 2 In an instance resolution, if the identifier of a class is specified in a request message, all instances of the class will be returned.

If multiple identifiers are simultaneously specified in a request message, the result may comprise a conjunctive set of data parcels. For example, if identifiers of the class meta-class and the property meta-class are specified in a request message, then the response message will contain a conjunctive set of data parcels which comprises instances of those meta-classes.

NOTE A complex query mechanism, such as retrieving classes through a class hierarchy, will be specified either in another standard or in another part of IEC 62656.

If an identifier of an item class is specified in a request message, an instance contained in a response message to the request will contain values of all properties of the item class. If a parcel server prepares a class as a view to an item class, then the identifier of the class as a view is available instead of specifying the identifier of the item class. That is, such a class as a view is used for specifying a predefined set of useful or meaningful inquiries about the properties of a product. Such a class as a view can be modelled based on the dictionary extension mechanism specified in ISO 13584-24:2003. It can be understood as a set of typical inquiries about a product class, like FAQs (Frequently Asked Questions).

Figure 7 shows an example of the instance resolution service. The table in Figure 7 contains instances of an item class C001 in an IEC 62656-1 spreadsheet form. A class QCA001 is a class as a view to the item class C001, which imports properties P001, P002 and P003 from the item class C001. When User A sends a request specifying C001, then he/she will obtain all instances of the class. Meanwhile, User B sends a request specifying QCA001 instead of C001 to obtain instances of item class C001, each of which has values of only P001, P002 and P003.



#CLASS_ID:=C001					
#CLASS_NAME.en :=Regional cuisine					
#PARCEL_CC:=1					
#PROPERTY_ID	P001	P002	P003	P004.en	P005.en
#PROPERTY_NAME.en	maker identifier	mass of salt in...	mass of sugar ...	place or ...	cuisine that ...
#REQUIREMENT	KEY			KEY	KEY
	HM	0.01	0.02	Kagoshima	Sakezushi
	NO	0.02	0.01	Vienna	Schnitzel
	NM	0.03	0.02	Osaka	Takoyaki
	LO	0.01	0.03	Sichuan	Mapodoufu
	WW	0.02	0.01	Hagen	Sauerkraut
	GP	0.07	0.04	Poitiers	Pot au feu

Figure 7 – Overview of parcel resolution service

5.5.2 Request message

5.5.2.1 General

The request message of the parcel resolution service shall contain "requestKind", "keywordKind", "keyword", "language", "pwsCodificationMode", "startPoint" and "endPoint" as its parameters. Table 5 shows the summary of those parameters.

Table 5 – Structure of a request message of the parcel resolution service

Name	Data type	Obligation	Example
requestKind	String	Mandatory	DEFINITION
keywordKind	String	Mandatory	ID
keyword	String	Mandatory	0112/2///61360_4#AAA013 0112/2///61360_4#AAA013* rated voltage
dictionaryId	String	Optional	0112/2///61360_4
language	String	Optional	en en, fr
pwsCodificationMode	String	Mandatory	VERTICAL
notationFormat	String	Optional	XML JSON
startPoint	Integer	Optional	10
endPoint	Integer	Optional	101

5.5.2.2 requestKind

The parameter "requestKind" has a string value, either of "DEFINITION" or "INSTANCE".

If "DEFINITION" is specified, it is regarded that information of an ontological entity itself, which corresponds to the identifier or name specified in a request message, will be resolved. This is used for retrieving all information of a class by specifying its identifier, including code, preferred name and definitions.

EXAMPLE 1 In the "DEFINITION" mode, if the identifier of a class is specified, information of the class, including its code, name and definition, will be retrieved.

If "INSTANCE" is specified, it is regarded that instances of ontological entities, which correspond to identifiers or names specified in a request message, will be resolved. This is used, for example, to obtain codes of all classes in an ontology by specifying the identifier of the class meta-class.

EXAMPLE 2 In the "INSTANCE" mode, if the identifier of a class meta-class is specified, information of each class of an ontology, including its code, name and definition, will be retrieved.

5.5.2.3 keywordKind

The parameter "keywordKind" has a string value, either of "ID" or "NAME". If "ID" is specified, a value of the parameter "keyword" shall contain a list of identifiers of ontological entities. If "NAME" is specified, a value of the parameter "keyword" shall contain a list of the names of ontological entities.

5.5.2.4 keyword

The parameter "keyword" has a string value which consists of one of the following:

– a list of identifiers of ontological entities with/without a search scope modifier specified in 5.3;

– a list of names of ontological entities.

If a parcel server receives keywords through a parcel web service, then it will return a result which matches one of the keywords to its client.

Rules of description for a value of the parameter "keyword" are summarized in the following list.

– If more than one item is specified, each item shall be separated by a comma ",".
– If a name itself contains a double quotation mark, the double quotation mark shall be encoded by another double quotation mark.
– If a name itself contains a comma or double quotation mark, then the name shall be enclosed between opening and closing double quotation marks.

EXAMPLE 1 When "0112/2///62656_1#MDC_C002##001,0112/2///62656_1#MDC_C003##001" as a value of the parameter "keyword" and "INSTANCE" as a value of the parameter "requestKind" are specified in a request message of the parcel resolution service, whole classes and properties which are stored in the parcel server are expected to be returned to the parcel client. Here, "0112/2///62656_1#MDC_C002##001" and "0112/2///62656_1#MDC_C003##001" are ICIDs of the class meta-class and the property meta-class, respectively, which are defined in IEC 62656-1:2014.

EXAMPLE 2 When "0112///61360_4#AAA001##001*" as a value of the parameter "keyword" and "DEFINITION" as a value of the parameter "requestKind" are specified in a request message of the parcel resolution service, whole dictionary elements under the specified class are expected to be returned to the parcel client.

5.5.2.5 dictionaryId

The optional parameter "dictionaryId" has a string value which indicates the identifier of a data dictionary for limiting a search scope to items of the data dictionary. In other words, items of other data dictionaries stored in a repository will be ignored in the request.

5.5.2.6 language

The parameter "language" has a string value which consists of a list of languages, each language of which is one of the following in accordance with the ICID specified in IEC 62656-1:

– two-letter language code based on ISO 639-1, such as "en";
– the combination of a two-letter language code based on ISO 639-1 and a two-letter country code based on ISO 3166-1, which are combined with a hyphen, such as "en-US".

If one or more language is specified, a parcel server will return the content of the specified languages to its client.

NOTE 1 If more than one language is specified, each language is separated by a comma ",".

NOTE 2 If no language is specified in a request, the content of a standard language is returned.

5.5.2.7 pwsCodificationMode

The parameter "pwsCodificationMode" has a string value, either "VERTICAL" or "LATERAL". If "VERTICAL" is specified, a conjunctive set of data parcels returned as a result of the parcel resolution service shall be structured in the vertical codification specified in Clause 5. If "LATERAL" is specified, a conjunctive set of data parcels returned as a result of the parcel resolution service shall be structured in the lateral codification specified in Clause 5.

5.5.2.8 notationFormat

The parameter "notationFormat" has a string value, either "XML" or "JSON". If "JSON" is specified, a conjunctive set of data parcels returned as a result of the parcel resolution service shall be in the form of the JSON format specified in Clause 7. If "XML" is specified, a conjunctive set of data parcels returned as a result of the parcel resolution service shall be in the form of the XML format specified in Clause 8.

If a parcel server supports transaction of inquiries, a notation format can be negotiated between the parcel server and a parcel client before starting transaction. In such a case, this parameter can be omitted in each request message.

5.5.2.9 startPoint

The parameter "startPoint" indicates a starting point or continuation point of a (conjunctive set of) data parcel(s) as a result of an execution of a resolution request. "startPoint" has a positive integer value. Its default value is 1. If this parameter is not specified, the default value will be applied.

For each instance which is contained in a (conjunctive set of) data parcel(s) as a result of an execution of a resolution request, the number shall be virtually assigned.

5.5.2.10 endPoint

The parameter "endPoint" indicates an ending point of a result of a parcel resolution service. "endPoint" has a positive integer value which shall be equal to or greater than the value of the parameter "startPoint". If this parameter is not specified, all of the items after the starting point will be returned as a result of the request.

For each instance which is contained in a (conjunctive set of) data parcel(s) as a result of an execution of a resolution request, the number shall be virtually assigned.

5.5.3 Response message

The response message of the parcel resolution service shall contain "conjuntiveParcels" as its parameter. The parameter has a string value which consists of a conjunctive set of data parcels which matches the conditions specified in a request message. Table 6 shows the summary of the structure of a response message.

Table 6 – Structure of a response message of the parcel resolution service

Name	Data type	Obligation	Example
conjunctiveParcels	String	Mandatory	<?xml version="1.0" encoding="utf-8"?> <conjunctiveParcels ontoLayer="DO"> <parcel> ... </parcel> </conjunctiveParcels>

5.5.4 Exception

The following standard-defined exceptions specified in 5.2 will apply to the parcel resolution service:

– no implementation,
– no element found,
– unexpected parameter,
– unexpected value,
– no permission.

5.6 Parcel subscription service

5.6.1 General

The parcel subscription service is a service enabling a parcel client to start or stop receiving a notification which informs the parcel client that the status, information or a value of the specified item has been changed. Here, the change of an item means an addition, modification or deletion of an item. If an error occurs during an execution of a request from a client, for example if the parcel server detects one or more invalid identifier, the parcel server should terminate the execution of the request and then throw an exception to the client, with some detailed information about the errors. In such a case, the subscription status is not changed on the parcel server.

The parcel subscription service provides push subscriptions. That is, if the status or a value of an item is changed, the parcel server will send a notification to its subscribers.

5.6.2 Request message

5.6.2.1 General

The request message of the parcel subscription service shall contain "requestKind", "identifiers" and "subscriptionOperationKind" as its parameters. Table 7 shows the summary of the structure of a request message.

Table 7 – Structure of a request message of the parcel subscription service

Name	Data type	Obligation	Example
requestKind	String	Mandatory	DEFINITION
identifiers	String	Mandatory	0112/2///61360_4#AAA013 0112/2///61360_4#AAA013* AAA013
subscriptionOperationKind	String	Mandatory	START
expirationDate	String	Optional	20200131 2020-01-31

5.6.2.2 requestKind

The parameter "requestKind" has a string value, either of "DEFINITION" or "INSTANCE".

If "DEFINITION" is specified, it is regarded that a subscription to the information of an ontological entity itself, which corresponds to the identifier specified in a request message, will be started or ended.

If "INSTANCE" is specified, it is regarded that a subscription to instances of ontological entities, which corresponds to identifiers specified in a request message, will be started or ended.

5.6.2.3 identifiers

The parameter "identifiers" has a string value which consists of a list of identifiers of ontological entities with/without search scope modifiers specified in 5.3, which a parcel client or user requests in order to start a subscription to them.

5.6.2.4 subscriptionOperationKind

The parameter "subscriptionOperationKind" has a string value, either "START" or "END". If "START" is specified, a subscription to ontological entities of identifiers specified in the parameter "identifiers" will be started. If "END" is specified, a subscription to ontological entities of identifiers specified in the parameter "identifiers" will be ended.

5.6.2.5 expirationDate

The parameter "expirationDate" has a string value for specifying a carender date in the form of "YYYYMMDD" or "YYYY-MM-DD" conformant to ISO 8601-1 and ISO 8601-2. If a date is specified in a request message, the subscription to ontological entities, which are started with this subscription request, will automatically end on the parcel server on the specified date.

5.6.3 Response message

The response message shall contain "operationResult" as its parameter. The parameter has a Boolean value indicating whether the execution of a subscription request has succeeded or not. If an execution succeeds, the ontology server will return a "true" indication to the parcel client. If an execution fails, one of the exceptions will be thrown or returned to a parcel client, in addition to returning a "false" indication. Table 8 shows the summary of the structure of a response message.

Table 8 – Structure of a response message of the parcel subscription service

Name	Data type	Obligation	Example
operationResult	Boolean	Mandatory	true

5.6.4 Exception

The following standard-defined exceptions specified in 5.2 will apply to the parcel subscription service:

– no implementation,
– no element found,
– unexpected parameter,
– unexpected value,
– no permission.

5.6.5 Specification of change notification

A notification which will be sent to a parcel client consists of a mandatory parameter "identifiers" and two optional parameters "defaultDataSupplier" and "defaultDataVersion". Table 9 shows the summary of the specification of a notification.

The parameter "identifiers" is used for specifying a list of identifiers of items, in which the parcel client is interested. Those identifiers are described using a comma as a separator.

The optional parameters can be used to prevent the repetition of the same RAI or VI, as a result, enabling the reduction of the data size of the notification. If "defaultDataSupplier" is present in a notification, an RAI may be omitted from some identifiers in the notification. In this case, the value of "defaultDataSupplier" should be applied to those identifiers. In the same manner, if "defaultDataVersion" is present in a notification, a VI may be omitted from some identifiers in the notification. In this case, the value of "defaultDataVersion" should be applied to those identifiers.

Table 9 – Specification of a notification

Name	Data type	Obligation	Example
identifiers	String	Mandatory	
defaultDataSupplier	String	Optional	
defaultDataVersion	String	Optional	

6 Specification of parcel data representation in a web service message

6.1 General

The format of data exchanged through a web service should satisfy the following criteria:

– specification is publicly available;

– the format is widely used in enterprise systems and devices including mobile devices and application;

– libraries of computer software for various programming languages are provided.

In accordance with those criteria, XML and JSON are adopted in this document.

In industry, XML has been traditionally used as a data format for data exchange among industrial systems. XML has some advantages, such as a well-defined schema and security technique over the web. ISO 13584-32 specifies the XML schema [5] for representing ISO 13584-42 compliant data in an XML form. On the other hand, there are some disadvantages to XML, such as the big data size because each item of data should be enclosed between a pair of XML tags.

Another data notation, JSON is also used, especially within mobile applications. JSON gives a simple and lightweight description of data, based on the JavaScript notation. Because of ease of use and understanding, JSON is used not only in JavaScript but also in other popular programming languages such as Java, C#, PHP and python, as a data format exchanged among systems, (mobile) devices and applications.

6.2 Basic data representation

Data parcels are exchanged between a parcel server and a parcel client, which are enveloped in a request or response message through a web service. This document specifies data representation of data parcels in a message.

Figure 8 shows the conceptual data structure of a conjunctive set of data parcels specified in IEC 62656-1. A conjunctive set of data parcels named "conjunctive parcels" comprises one or more data parcel, each of which is named "parcel". Each data parcel consists of two sections: a header section and a data section, in that order. The header section further consists of two subsections: a class header section and a schema header section. The class header section describes the information about the data parcel, while the schema header section describes the data schema of the data parcel. The data section describes instances in accordance with the schema described in the schema header section.

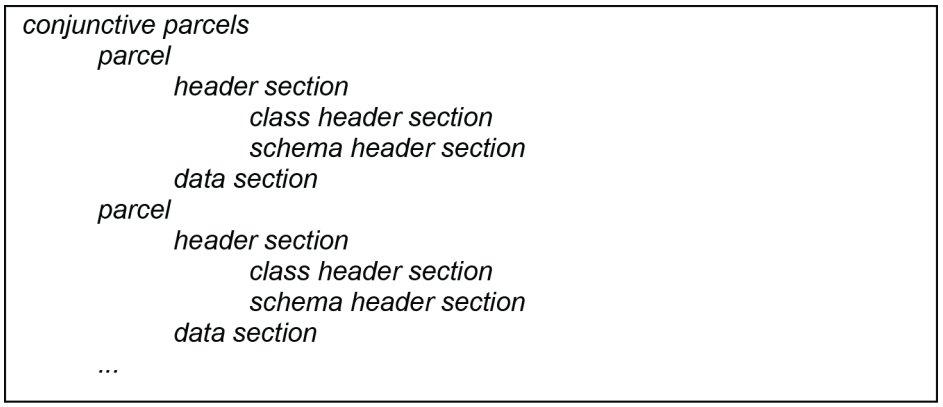


Figure 8 – Basic structure of a data representation for a conjunctive set of data parcels

Subclause 6.3 specifies reserved keywords used for data representation in both JSON and XML. For data representation in JSON, each keyword is used as a JSON name. For data representation in XML, each keyword is used as an XML element name or attribute name.

6.3 Reserved keywords

6.3.1 Keyword indicating conjunctive parcels

Keyword: conjunctiveParcels
Name: conjunctive parcels
Definition: reserved keyword to collect one or more data parcels, each of which is enclosed in the keyword "parcel"
Occurrence: 1 for a request or response message

6.3.2 Keyword indicating parcel ontology layer of a set of data parcels

Keyword: ontoLayer
Name: parcel ontology layer
Definition: reserved keyword to describe a parcel ontology layer of a set of data parcels collected by the keyword "conjunctiveParcels"
Description: The value shall be one of "AO", "MO", "DO" or "DL"
Occurrence: 1 for an occurrence of the keyword "conjunctiveParcels"

6.3.3 Keyword indicating header section

Keyword: header
Name: header section
Definition: reserved keyword to collect the class header section enclosed in the keyword "classHeader" and the schema header section enclosed in the keyword "schemaHeader"
Occurrence: 1 for an occurrence of the keyword "parcel"

6.3.4 Keyword indicating class header section

Keyword: classHeader
Name: class header section
Definition: reserved keyword to collect instructions of the class header section
Occurrence: 1 for an occurrence of the keyword "header"

6.3.5 Keyword indicating schema header section

Keyword: schemaHeader
Name: schema header section
Definition: reserved keyword to collect instructions of the schema header section
Occurrence: 1 for an occurrence of the keyword "header"

6.3.6 Keyword indicating data section

Keyword: data
Name: data section
Definition: reserved keyword to collect instances
Occurrence: 1 for an occurrence of the keyword "conjunctiveParcels"

6.3.7 Keyword indicating default supplier in data section

Keyword: defaultSupplier
Name: default supplier
Definition: reserved keyword to specify the default supplier of property identifiers in a data section, each of which is used as a key for its values
Occurrence: 1 for an occurrence of the keyword "data"

6.3.8 Keyword indicating default version in data section

Keyword: defaultVersion
Name: default version
Definition: reserved keyword to specify default version of property identifiers in a data section, each of which is used as a key for its values
Occurrence: 1 for an occurrence of the keyword "data"

6.4 Additional instructions to data parcels for parcel web services

6.4.1 Codification mode

In both JSON and XML, two codification modes are provided. One is vertical codification and the other is lateral codification.

Vertical codification is a mode for collecting values per property. The vertical codification is suitable for obtaining the values of a property, for example, checking whether a specified value exists in a list of values of a property or not, and for aggregating and calculating values of a property.

Lateral codification is a mode for collecting values per instance. The lateral codification is suitable for obtaining the property-value pairs of an instance.

A benefit of providing those two modes is the continuity of a process. If an error occurs during the communication between a parcel server and a parcel client, the parcel server (or the parcel client) may receive only a subset of data parcels as a result. In the case of vertical codification, if the parcel server (or the parcel client) needs a set of values of a property and it is contained in the subset of data parcels, the parcel server (or the parcel client) can continue to execute the process. In the case of lateral codification, a similar scenario is applicable.

This document defines the following instruction #PWS_CODIFICATION_MODE to specify which mode is used to represent a conjunctive set of data parcels exchanged through a parcel web service.

Keyword: #PWS_CODIFICATION_MODE
Name: codification mode
Definition: instruction for a data parcel, indicating what codification is used to represent the data parcel
Description: If "LATERAL" is specified, the data parcel is represented in the lateral codification. If "VERTICAL" is specified, the data parcel is represented in the vertical codification.
Category: optional – functional
Format: Either LATERAL or VERTICAL may be assigned as a value.

6.4.2 Intended language

A data parcel can contain definitions in multiple languages. Through a parcel registration service, the content of all languages contained in a conjunctive set of data parcels will be processed on the receiving side (a parcel server, usually), but this is sometimes dangerous because it may contain unintended changes in unintended languages. For example, in a translation from English to another language, the English content can be used simply as a reference in a translation into another language, called a "language variant". In this case, even if the English definition is changed accidentally, the change shall be ignored on the parcel server. Thus the language definition(s) specifically intended for change, shall be clearly marked. To prevent an unintended change of the content, this document defines the following instruction "#INTENDED_LANGUAGE" for the parcel registration service.

Keyword: #INTENDED_LANGUAGE
Name: intended language(s)
Definition: designation of language(s) in line with ISO 639-1, which contains (contain) intended change of definitions
Description: If there is a language in a data parcel which is not specified as an intended language, its definition shall be ignored on the receiving side (a parcel server, usually). If this instruction is not specified in a data parcel, all languages in the data parcel, including the source language, shall be regarded as being intended. The source language can be specified as well as other languages as an element of the value of this command. If the source language is specified, the description in the source language will contain some intended change in the definitions.
Category: optional – functional
Format: The keyword "#INTENDED_LANGUAGE" shall be described in the first column and language code(s) shall be described after the keyword separated by the symbol ":=" (colon-equals). The language code according to ISO 639-1, possibly followed by a two-letter country code based on ISO 3166-1, enables the identification of the language. If plural languages are specified, those language codes shall be separated by commas and shall be enclosed within a pair of curly brackets "{" and "}". The cells in the second and subsequent columns shall be ignored.

6.4.3 Default value

For some computer network environments, such as a mobile network and low bandwidth network, a means of reducing the total amount of data to be exchanged between a server and a client should be taken into account to achieve efficient and reliable network communication.

IEC 62656-1 specifies two instructions #DEFAULT_DATA_SUPPLIER and #DEFAULT_DATA_VERSION to avoid repeating the same RAI and VI, respectively, of ICIDs in the data section. In a similar manner, a means of specifying a default value of a property would be helpful.

This document defines the following instruction #DEFAULT_VALUE to specify a default value of a property. If a default value is set to a property column in a parcel sheet, the value will be automatically applied to all blank cells of the data section of the property column.

Keyword: #DEFAULT_VALUE
Name: default value
Definition: preset value of the property specified by the property ID
Description: The default value for the property shall be applied to all the blank fields of the property in the data section, where values are unspecified. To keep a null value in a field, a special value "(null)", which consists of a string "null" enclosed in a pair of parentheses, shall be explicitly described in the field. If a data parcel containing a default value is to be sent to an external system that is not capable of processing the default value, the data parcel should be preprocessed to fill blank fields with the default value except in cases where the field is explicitly marked "(null)" before it is actually sent to the external system.
Category: optional – functional
Format: The keyword "#DEFAULT_VALUE" will be described in the first column. The default values will be described in the cells corresponding to the second and subsequent columns. Each such default value corresponds to the property which is described in the #PROPERTY_ID line.

Figure 9 shows how default values are described and applied to cells in the data section. Blank cells circled in thick lines indicates that the values of those cells will be replaced with their corresponding default values when the data parcels are processed.

#PROPERTY_ID	P001	P002	P003	P004	P005
#DATATYPE	STRING_TYPE	STRING_TYPE	LEVEL(MAX) OF INT_TYPE	BOOLEAN_TYPE	ENUM_REAL_ TYPE(E001 (1.0, 2.0, 3.0)
#DEFAULT_ VALUE	**New York**		**2**	**TRUE**	**1.0**
			1.	FALSE	2.0
	Paris				3.0
			3		2.0

IEC

Figure 9 – Example of the use of default values

6.5 Description of instructions

IEC 62656-1 specifies the instructions to be used in the header section of each data parcel. Those instructions are categorized into four types: "mandatory", "optional – functional", "optional – informative" and "comment". According to IEC 62656-1, "mandatory" is a category indicating that an instruction of this kind shall be present in a data parcel. "optional – functional" is a category indicating that if an instruction of this kind is present in a data parcel, its value shall be processed according to the function implied by the keyword. "optional – informative" is a category indicating that a value of an instruction of this kind is simply provided to users of a data parcel as an informative message. Finally, "comment" is a category indicating that a line where an instruction of this kind is described shall be interpreted as a comment line. The web service specified in this document is intended to be implemented for data exchange between a parcel server and a parcel client; therefore, instructions of both "mandatory" and "optional – functional" shall be treated in a data parcel exchanged between them.

The string of each instruction starts with a number sign "#", such as #CLASS_ID, to describe the class identifier for a data parcel. However, the XML prohibits the use of a number sign "#" as a leading character of an element and attribute. Therefore, this document specifies naming conventions in accordance with the instructions specified in IEC 62656-1, which allows the safe use of those for data representation in XML or in JSON.

The naming conventions are as follows:

– a leading number sign "#" shall be removed.
– an underscore "_" shall be removed.
– each alphabetic letter immediately after where each underscore "_" used to be shall be changed to its capitalized form.
– the other alphabetic letters shall be changed to their lower-case form.

For example, the instruction "#SUPER_ALT_CLASSID" is changed to "superAltClassid".

The descriptions of the instructions specified in this document are listed in Table 10.

Table 10 – Description of the instructions specified in IEC 62656-1

Instruction keyword specified in IEC 62656-1	Description in XML and JSON
#CLASS_ID	classID
#PARCEL_MODE	parcelMode
#PARCEL_ID	parcelID
#DEFAULT_SUPPLIER	defaultSupplier
#DEFAULT_VERSION	defaultVersion
#DEFAULT_DATA_SUPPLIER	defaultDataSupplier
#DEFAULT_DATA_VERSION	defaultDataVersion
#OBJECT_ID_NAME	objectIDNameobjectIDName
#PROPERTY_ID	propertyID
#REQUIREMENT	requirement
#ID_ENCODE	idEncoding

NOTE This document does not limit the use of instruction keywords of "optional – informative" in a web service. Annex D, Table D.1 gives the description for those keywords.

Moreover, the translation of the instructions specified in this document is listed in Table 11.

Table 11 – Description of the instructions specified in this document

Instruction keyword specified in this document	Description in XML and JSON
#PWS_CODIFICATION_MODE	pwsCodificationMode
#INTENDED_LANGUAGE	intendedLanguage
#DEFAULT_VALUE	defaultValue

7 Data representation in JSON

7.1 Basic structure of data representation in JSON

In accordance with ISO/IEC 21778, a JSON value is one of an object, array, number, string, true, false and null. Each JSON object is represented as a pair of curly brackets "{" and "}" surrounding zero or more name and value pairs. Each JSON array is represented as a pair of square brackets "[" and "]" surrounding zero or more objects and/or values.

Both a JSON object and a JSON array are structured data. The difference is that the former is an unordered collection of name/value pairs, while the latter is an ordered collection of objects and/or values.

The basic structure of data representation in JSON is depicted in Figure 10. ISO/IEC 21778 recommends that JSON names within a JSON object should be unique. In the basic structure of a conjunctive set of data parcels depicted in Figure 8, the element "parcel" is duplicated in the element "conjunctiveParcels"; therefore, the element "parcel" cannot be represented using a JSON object. Instead of the element "parcel", this document specifies a special JSON name "parcels", which contains an array of JSON objects, each of which indicates a data parcel. The JSON name is defined in 7.2. The other elements in the basic structure of a conjunctive set of data parcels are represented using JSON objects.

The JSON name "classHeader" in the JSON name "header" is used for containing the information of the class header section of a data parcel. Data representation of the class header section in JSON consists of JSON name/value pairs, each of which indicates an instruction and its value. The specification of JSON names for describing the class header section is provided in 7.3.

The JSON name "schemaHeader" in the JSON name "header" is used to contain a list of the information of properties of a data parcel, which are the schema for the data section. The specification of JSON names for describing the schema header section is provided in 7.4.

The JSON name "data" is used to contain the data of the data section in a vertical JSON notation or a lateral JSON notation. Those JSON notations are specified in 7.5.

```
{
  "conjunctiveParcels": {
    "ontoLayer": <any of AO, MO, DO or DL>,
    "parcels": [
      {
        "header": {
          "classHeader":{
            <Data representation of the class header section in JSON>
          },
          "schemaHeader":[
            <Data representation of the schema header section in JSON>
          ]
        },
        "data":{
          <Data representation of the data section in JSON>
        }
      },
      {
        "header": {
          "classHeader":{
            <Data representation of the class header section in JSON>
          },
          "schemaHeader":[
            <Data representation of the schema header section in JSON>
          ]
        },
        "data":{
          <Data representation of the data section in JSON>
        }
      },
      ...
    ]
  }
}
```

IEC

Figure 10 – Basic structure of data representation in JSON

The JSON schema for describing JSON data in the vertical JSON notation is provided in Annex A, A.1.1, while the one for describing JSON data in the lateral JSON notation is provided in A.1.2. The JSON schema for the JSON notation of the exceptions is specified in A.1.3.

7.2 Reserved JSON name indicating an array of data parcels

JSON name: parcels
Name: data parcel
Definition: reserved keyword to describe an array of data parcels, each of which is described as a JSON object
Occurrence: 1 for an occurrence of the keyword "conjunctiveParcels"

7.3 JSON names for class header section

7.3.1 JSON name indicating the instruction "#CLASS_ID"

JSON name: classID
Definition: reserved JSON name to describe a value of the instruction named "#CLASS_ID" in a data parcel
Data type: String
Obligation: Mandatory
Example: "classID": "MDC_C002"

7.3.2 JSON name indicating the instruction "#PARCEL_MODE"

JSON name: parcelMode
Definition: reserved JSON name to describe a value of the instruction named "#PARCEL_MODE" in a data parcel
Description: If this keyword is present, its value may be any of FULL, UPDATE, or PARTIAL. If a data parcel is used in a parcel registration service, this keyword shall be present.
Data type: String
Obligation: Optional
Example: "parcelMode": "PARTIAL"

7.3.3 JSON name indicating the instruction "#PARCEL_ID"

JSON name: parcelID
Definition: reserved JSON name to describe a value of the instruction named "#PARCEL_ID" in a data parcel
Data type: String
Obligation: Optional
Example: "parcelID": "2006-06-25 08:19:49"

7.3.4 JSON name indicating the instruction "#DEFAULT_SUPPLIER"

JSON name: defaultSupplier
Definition: reserved JSON name to describe a value of the instruction named "#DEFAULT_SUPPLIER" in a data parcel
Data type: String
Obligation: Optional
Example: "defaultSupplier": "0112/2///62656_1"

7.3.5 JSON name indicating the instruction "#DEFAULT_VERSION"

JSON name: defaultVersion
Definition: reserved JSON name to describe a value of the instruction named "#DEFAULT_VERSION" in a data parcel
Data type: String
Obligation: Optional
Example: "defaultVersion": "1"

7.3.6 JSON name indicating the instruction "#OBJECT_ID_NAME"

JSON name: objectIDName
Definition: reserved JSON name to describe a value of the instruction named "#OBJECT_ID_NAME" in a data parcel
Description: If this keyword is present, its value may be either "GUID" or "UUID".
Data type: String
Obligation: Optional
Example: "objectIDName": "GUID"

7.3.7 JSON name indicating the instruction "#ID_ENCODE"

JSON name: idEncode
Definition: reserved JSON name to describe a value of the instruction named "#ID_ENCODE" in a data parcel
Data type: String
Obligation: Optional
Example: "idEncode": "ICID"

7.3.8 JSON name indicating the instruction "#PWS_CODIFICATION_MODE"

JSON name: pwsCodificationMode
Definition: reserved JSON name to describe a value of the instruction named #PWS_CODIFICATION_MODE" in a data parcel
Description: If this keyword is present, its value shall be either "VERTICAL" or "LATERAL".
Data type: String
Obligation: Mandatory
Example: "pwsCodificationMode": "LATERAL"

7.3.9 JSON name indicating the instruction "#INTENDED_LANGUAGE"

JSON name: intendedLanguage
Definition: reserved JSON name to describe a value of the instruction named #INTENDED_LANGUAGE" in a data parcel
Description If more than one language is specified, each language is separated by a comma ",".
Data type: String
Obligation: Optional
Example: "intendedLanguage": "en,fr"

7.4 JSON names for schema header section

7.4.1 Basic structure of data representation for schema header section in JSON

The JSON name "schemaHeader" has JSON objects, each of which indicates information of a property which is used as the schema of a cell column. The structure of data representation for the schema header section in JSON is depicted in Figure 11. The information of a property contained in each JSON object is specified in 7.4.2.

```
"schemaHeader": [
  {
     <information of a property as schema of a cell column>
  },
  {
     <information of a property as schema of a cell column>
  },
   …
]
```

IEC

Figure 11 – Basic structure of data representation for schema header section in JSON

An example of data representation for a schema header section in JSON is provided in Annex A, Clause C.2.

7.4.2 JSON names for the schema header section

7.4.2.1 JSON name indicating a value of the instruction "#PROPERTY_ID"

JSON name: propertyID
Definition: reserved JSON name to describe a value of the instruction named "#PROPERTY_ID" in a data parcel
Data type: String
Obligation: Mandatory
Example: "propertyID": "MDC_P001_5"

7.4.2.2 JSON name indicating a value of the instruction "#DEFAULT_DATA_SUPPLIER"

JSON name: defaultDataSupplier
Definition: reserved JSON name to describe a value of the instruction named "#DEFAULT_DATA_SUPPLIER" in a data parcel
Data type: String
Obligation: Optional
Example: "defaultDataSupplier": "0112/2///61360_4"

7.4.2.3 JSON name indicating a value of the instruction "#DEFAULT_DATA_VERSION"

JSON name: defaultDataVersion
Definition: reserved JSON name to describe a value of the instruction named "#DEFAULT_DATA_VERSION" in a data parcel
Data type: String
Obligation: Optional
Example: "defaultDataVersion": "1"

7.4.2.4 JSON name indicating the instruction "#DEFAULT_VALUE"

JSON name: defaultValue
Definition: reserved JSON name to describe a value of the instruction named #DEFAULT_VALUE" in a data parcel
Data type: String
Obligation: Optional
Example: "defaultValue": "New York"

7.4.2.5 JSON name indicating a value of the instruction "#REQUIREMENT"

JSON name: Requirement
Definition: reserved JSON name to describe a value of the instruction named "#REQUIREMENT" in a data parcel
Description: If this keyword is present, its value shall be either blank or one of the following: "CONST", "KEY", "NOT_NULL", "MANDATORY", "OPTIONAL" or "OBSOLETE". When it is blank, it is equivalent to the designation as "OPTIONAL", while the values "MANDATORY", "OPTIONAL", "OBSOLETE" may be abbreviated as "MAND", "OPT", "OBS", respectively. The details of each value are specified in 5.11.24 of IEC 62656-1:2014.
Data type: String
Obligation: Optional
Example: "requirement": "KEY"

7.5 Data representation for data section in JSON

7.5.1 Vertical JSON notation for data section

In the vertical JSON notation, the JSON name "data" contains a pair of the JSON name "values" and a JSON object as its value, which contains an unordered collection of JSON name/value pairs, each of which indicates a property identifier and its values. The JSON name "data" also has optional JSON names "defaultSupplier" and "defaultVersion" and their values. If the JSON name "defaultSupplier" and its value is present in the data section, the RAI of each property identifier which is used as a JSON name in the value of the JSON name "values" can be omitted and shall be complemented by a data receiver. In a similar manner, if the JSON name "defaultVersion" and its value is present in the data section, the VI of each property identifier which is used as a JSON name in the value of the JSON name "values" can be omitted and shall be complemented by a data receiver.

If a value is blank, "null" as a JSON value shall be described in a JSON array.

An example of the vertical JSON notation of the data section is provided in Clause C.1 and in C.2.1.

7.5.2 Lateral JSON notation for data section

In the lateral JSON notation, the JSON name "data" contains a pair of the JSON name "instances" and a JSON array as its value, each element of which is a JSON array of property-values of an instance. The order and number of elements of each JSON array of property-values shall be the same as the order and number of properties in the schema header section.

An example of the lateral JSON notation of the data section is provided in Clause C.1 and in C.2.2.

7.6 Character encode

ISO/IEC 21778 recommends the use of UTF-8, UTF-16 or UTF-32 as a character encode. In accordance with those specifications, all text data in JSON exchanged in the parcel web service should be encoded in any of UTF-8, UTF-16 or UTF-32.

8 Data representation in XML

8.1 Basic structure of data representation in XML

In accordance with the XML specification provided by World Wide Web Consortium (W3C), there are two ways to describe a value, one is an XML element and the other is an XML attribute. There is no rule or recommendation in the specification of which one should be used; however, an XML element is suitable for representing the structured data. Therefore, as a basis of the use of those methods, this document specifies the use of an XML element to describe the elements in the basic structure of a conjunctive set of data parcels, depicted in Figure 2. An XML attribute is used to represent a supplemental value for an element.

The basic structure of data representation in XML is depicted in Figure 12. In the basic structure of a conjunctive set of data parcels depicted in Figure 8, all of the elements are represented using XML elements. A parcel ontology layer described as a value of "ontoLayer" is a supplemental piece of information for a conjunctive set of parcels; therefore, it is represented as an XML attribute.

The XML element "classHeader" in the XML element "header" is used for containing the information of the class header section of a data parcel. The XML element of the class header section consists of XML elements, each of which indicates an instruction and its value. The specification of data representation for the class header section in XML is specified in 8.3.

The XML element "schemaHeader" in the XML element "header" is used for containing a list of the information of properties of a data parcel, which is the schema for the data section. The specification of data representation for the schema header section in XML is specified in 8.3.9.

The XML element "data" is used for containing data of the data section in a vertical XML notation or a lateral XML notation. Those XML notations are specified in 8.5.

```
<conjunctiveParcels ontLayer="<any of AO, MO, DO or DL>">
    <parcel>
        <header>
            <classHeader>
                <Data representation of class header section in XML>
            </classHeader>
            <schemaHeader>
                <Data representation of schema header section in XML>
            </schemaHeader>
        </header>
        <data>
            <Data representation of data section in XML>
        </data>
    </parcel>
</conjunctiveParcels>
```

IEC

Figure 12 – Basic structure of data representation in XML

The XML schema for representing data in the vertical XML notation is provided in A.2.1, while the one for representing data in the lateral XML notation is provided in A.2.2. The XML schema for the XML notation of the exceptions is specified in A.2.3.

8.2 Reserved keyword indicating data parcel

Keyword: parcel
Name: data parcel
Definition: reserved XML element to describe a data parcel
Occurrence: 1..n for an occurrence of the keyword "conjunctiveParcels"

8.3 XML elements for class header section

8.3.1 XML element indicating the instruction "#CLASS_ID"

Keyword: classID
Definition: reserved XML element to describe a value of the instruction named "#CLASS_ID" in a data parcel
Data type: String
Obligation: Mandatory
Example: <classID>MDC_C002</classID>

8.3.2 XML element indicating the instruction "#PARCEL_MODE"

Keyword: parcelMode
Definition: reserved XML element to describe a value of the instruction named "#PARCEL_MODE" in a data parcel
Description: If this XML element is present, its value may be any of "FULL", "UPDATE" or "PARTIAL". If a data parcel is used in a parcel registration service, this element shall be present.
Data type: String
Obligation: Optional
Example: <parcelMode>PARTIAL</parcelMode>

8.3.3 XML element indicating the instruction "#PARCEL_ID"

Keyword: parcelID
Definition: reserved XML element to describe a value of the instruction named "#PARCEL_ID" in a data parcel
Data type: String
Obligation: Optional
Example: <parcelID>2006-06-25 08:19:49</parcelID>

8.3.4 XML element indicating the instruction "#DEFAULT_SUPPLIER"

Keyword: defaultSupplier
Definition: reserved XML element to describe a value of the instruction named "#DEFAULT_SUPPLIER" in a data parcel
Data type: String
Obligation: Optional
Example: <defaultSupplier>0112/2///62656_1</defaultSupplier>

8.3.5 XML element indicating the instruction "#DEFAULT_VERSION"

Keyword: defaultVersion
Definition: reserved XML element to describe a value of the instruction named "#DEFAULT_VERSION" in a data parcel
Data type: String
Obligation: Optional
Example: <defaultVersion>1</defaultVersion>

8.3.6 XML element indicating the instruction "#OBJECT_ID_NAME"

Keyword: objectIDName
Definition reserved XML element to describe a value of the instruction named "#OBJECT_ID_NAME" in a data parcel
Description: If this element is present, its value may be either "GUID" or "UUID".
Data type: String
Obligation: Optional
Example: <objectIDName>GUID</objectIDName>

8.3.7 XML element indicating the instruction "#ID_ENCODE"

Keyword: idEncode
Definition: reserved XML element for describing a value of the instruction named "#ID_ENCODE" in a data parcel
Data type: String
Obligation: Optional
Example: <idEncode>ICID</idEncode>

8.3.8 XML element indicating the instruction "#PWS_CODIFICATION_MODE"

Keyword: pwsCodificationMode
Definition reserved XML element to describe a value of the instruction named "#PWS_CODIFICATION_MODE" in a data parcel
Description: If this element is present, its value shall be either "VERTICAL" or "LATERAL".
Data type: String
Obligation: Mandatory
Example: <pwsCodificationMode>LATERAL</pwsCodificationMode>

8.3.9 XML element indicating the instruction "#INTENDED_LANGUAGE"

Keyword: intendedLanguage
Definition reserved XML element to describe a value of the instruction named "#INTENDED_LANGUAGE" in a data parcel
Description: If more than one language is specified, each language is separated by a comma ",".
Data type: String
Obligation: Optional
Example: <intendedLanguage>en,fr</intendedLanguage>

8.4 XML elements for schema header section

8.4.1 Basic structure of data representation for schema header section in XML

The XML element "schemaHeader" has XML elements "property", each of which indicates information of a property which is used as schema of a cell column. The structure of data representation for schema header section in XML is depicted in Figure 13. The information of a property contained in each XML element "property" is specified in 8.4.2.

```
<schemaHeader>
   <property>
      <information of a property as schema of a cell column>
   <property>
   <property>
      <information of a property as schema of a cell column>
   <property>
   …
</schemaHeader>
```

IEC

Figure 13 – Basic structure of data representation for schema header section in XML

8.4.2 XML elements of schema header section

8.4.2.1 XML element indicating the instruction "#PROPERTY_ID"

Keyword: propertyID
Definition: reserved XML element to describe a value of the instruction named "#PROPERTY_ID" in a data parcel
Data type: String
Obligation: Mandatory
Example: <propertyID>MDC_P001_5</propertyID>

8.4.2.2 XML element indicating the instruction "#DEFAULT_DATA_SUPPLIER"

Keyword: defaultDataSupplier
Definition: reserved XML element to describe a value of the instruction named "#DEFAULT_DATA_SUPPLIER" in a data parcel
Data type: String
Obligation: Optional
Example: <defaultDataSupplier>MDC_P001_5</defaultDataSupplier>

8.4.2.3 XML element indicating the instruction "#DEFAULT_DATA_VERSION"

Keyword: defaultDataVersion
Definition: reserved XML element to describe a value of the instruction named "#DEFAULT_DATA_VERSION" in a data parcel
Data type: String
Obligation: Optional
Example: <defaultDataVersion>1</defaultDataVersion>

8.4.2.4 XML element indicating the instruction "#DEFAULT_VALUE"

Keyword: defaultValue
Definition: reserved XML element to describe a value of the instruction named "#DEFAULT_VALUE" in a data parcel
Data type: String
Obligation: Optional
Example: <defaultValue>New York</defaultValue>

8.4.2.5 XML element indicating the instruction "#REQUIREMENT"

Keyword: requirement
Definition: reserved XML element to describe a value of the instruction named "#REQUIREMENT" in a data parcel
Description: If this keyword is present, its value shall be either blank or one of the following: "CONST", "KEY", "NOT_NULL", "MANDATORY", "OPTIONAL" or "OBSOLETE. When it is blank, it is equivalent to the designation as "OPTIONAL", while the values "MANDATORY", "OPTIONAL", "OBSOLETE" may be abbreviated as "MAND", "OPT", "OBS", respectively. The details of each value are specified in 5.11.24 of IEC 62656-1:2014.
Data type: String
Obligation: Optional
Example: <requirement>KEY</requirement>

8.5 XML elements and attributes for data section

8.5.1 Vertical XML notation of data section

8.5.1.1 Basic structure of data representation for data section in vertical XML notation

In the vertical XML notation, the XML element "data" contains XML elements "values", each of which is identified with its attribute "propertyID", where the property identifier is specified. Each XML element "values" also contains an ordered collection of XML elements "value". Each XML element "value" contains a property-value.

The XML element "data" has XML attributes "defaultSupplier" and "defaultVersion". If the XML attribute "defaultSupplier" is present in the data section, the RAI of each property identifier specified in the XML attribute "propertyID" of the XML element "values" can be omitted and shall be complemented by a data receiver. In a similar manner, if the XML attribute "defaultVersion" is present in the data section, the VI of each property identifier specified in the XML attribute "propertyID" of the XML element "values" can be omitted and shall be complemented by a data receiver.

Consequently, data representation for data section in the vertical XML notation is structured as depicted in Figure 14.

```
<data defaultSupplier="<RAI>" defaultVersion="<VI>">
  <values propertyID="<property ID>">
    <a value of the specified property in the vertical XML notation>
    <a value of the specified property in the vertical XML notation>
    …
  </values>
  <values propertyID="<property ID>">
    <a value of the specified property in the vertical XML notation>
    <a value of the specified property in the vertical XML notation>
    …
  </values>
  …
</data>
```

IEC

Figure 14 – Structure of data representation for data section in the vertical XML notation

An example of data representation in the vertical XML notation is provided in Clause C.1 and in C.3.1.

8.5.1.2 XML element indicating values of a property in vertical codification

Keyword: values
Definition: reserved XML element in the vertical codification mode to describe a list of values of a property
Description: The identifier of a property is specified in the attribute named "propertyID" of this element.
Obligation: Mandatory
Example:
```
<values propertyID="MDC_P001_5">
  <value>AAA001</value>
  <value>AAA002</value>
  <value>AAA003</value>
  <value>AAA013</value>
</values>
```

8.5.1.3 XML attribute indicating a property identifier for a value in vertical codification

Keyword: propertyID
Definition: reserved XML attribute of the XML element named "values" in the vertical codification mode, to describe the identifier of a property whose values are listed in the element
Description: When the RAI of the identifier is omitted, it shall be complemented with the value of the attribute named "defaultSupplier" of the element named "data". When the VI of the identifier is omitted, it shall be complemented with the value of the attribute named "defaultVersion" of the element named "data".
Data type: String
Obligation: Mandatory
Example:
```
<values propertyID="MDC_P001_5">
  <value>AAA001</value>
  <value>AAA002</value>
  <value>AAA003</value>
  <value>AAA013</value>
</values>
```

8.5.1.4 XML element indicating a property value in vertical codification

Keyword: value
Definition: reserved XML element in the vertical codification mode to describe a value of the property which is specified in the attribute named "propertyID" of the parent element named "values" of this element
Description: If a value is blank or null, the element should be described as "<value></value>" or "<value/>".
Data type: Any
Obligation: Mandatory
Example: <value>component</value>

8.5.2 Lateral XML notation of data section

8.5.2.1 Basic structure of data representation for data section in lateral XML notation

In the lateral XML notation, the XML element "data" contains XML elements "instance", each of which contains XML elements "value". Each XML element "value" contains a property-value and has an XML attribute "propertyID" which specifies the property identifier for the value.

In the same manner as the vertical XML notation for the data section, the XML element "data" has XML attributes "defaultSupplier" and "defaultVersion". If the XML attribute "defaultSupplier" is present in the data section, the RAI of the property identifier which is specified in the XML attribute "propertyID" of each XML element "value" can be omitted and shall be complemented by a data receiver. In a similar manner, if the XML attribute "defaultVersion" is present in the data section, the VI of the property identifier which is specified in the XML attribute "propertyID" of each XML element "value" can be omitted and shall be complemented by a data receiver.

Consequently, data representation for data section in the lateral XML notation is structured as depicted in Figure 15.

```
<data defaultSupplier="<RAI>" defaultVersion="<VI>">
  </instance>
      <a value of an instance in the lateral XML notation>
      <a value of an instance in the lateral XML notation>
      …
  </instance>
  <instance>
      <a value of an instance in the lateral XML notation>
      <a value of an instance in the lateral XML notation>
      …
  </instance>
    …
</data>
```

IEC

Figure 15 – Structure of data representation for data section in lateral XML notation

An example of data representation in the lateral XML notation is provided in Clause C.1 and in C.3.2.

8.5.2.2 XML element indicating an instance in lateral codification

Keyword: instance
Definition: reserved XML element in the lateral codification mode to describe property-value pairs of an instance
Description: Each property-value pair is described in a child element named "value" of this element.
Obligation: Mandatory
Example:

```
<instance>
  <value propertyID="MDC_P001_5">AAA001</value>
  <value propertyID="MDC_P004_1.en">component</value>
  <value propertyID="MDC_P010">UNIVERSE</value>
  <value propertyID="MDC_P014">{AAE001,AAE006}</value>
</instance>
```

8.5.2.3 XML element indicating a property value in lateral codification

Keyword: value
Definition: reserved XML element in the lateral codification mode to describe a property value pair
Description: A property identifier is described in the attribute named "propertyID" of this element, while its value is described as a value of this element. If a value is blank or null, the element should be described as "<value></value>" or "<value/>".
Data type: Any
Obligation: Mandatory
Example: <value propertyID="MDC_P004_1.en">component</value>

8.5.2.4 XML attribute indicating a property identifier for a value in lateral codification

Keyword: propertyID
Definition: reserved XML attribute of the XML element named "value" in the lateral codification mode, to describe a property identifier
Description: When the RAI of the identifier is omitted, it shall be complemented with the value of the attribute named "defaultSupplier" of the element named "data". When the VI of the identifier is omitted, it shall be complemented with the value of the attribute named "defaultVersion" of the element named "data".
Data type: String
Obligation: Mandatory
Example: <value propertyID="MDC_P004_1.en">component</value>

8.6 Character encode

The specification of XML recommends the use of UTF-8 or UTF-16. In accordance with that specification, all text data in XML exchanged in the parcel web service should be encoded either in UTF-8 or UTF-16.

Annex A
(normative)

Schema

A.1 JSON schema

A.1.1 Vertical JSON schema

The JSON schema for the vertical JSON notation of data parcels specified in Clause 7 is depicted in Figure A.1.


```
{
 "$schema": "http://json-schema.org/schema#",
 "type": "object",
 "properties": {
  "conjunctiveParcels": {
   "type": "object",
   "properties": {
    "ontLayer": {
     "enum": ["AO","MO","DO","DL"]
    },
    "parcels": {
     "type": "array",
     "items": {
      "type": "object",
      "properties": {
       "header": {
        "type": "object",
        "properties": {
         "classHeader": {
          "type": "object",
          "properties": {
           "classID": {
            "type": "string"
           },
           "parcelMode": {
            "type": ["string", "null"]
           },
           "parcelID": {
            "type": ["string", "null"]
           },
           "defaultSupplier": {
            "type": ["string", "null"]
           },
           "defaultVersion": {
            "type": ["string", "null"]
           },
           "objectIdName": {
            "type": ["string", "null"]
           },
           "idEncoding": {
            "type": ["string", "null"]
           },
```

IEC

Figure A.1 – Vertical JSON schema *(1 of 2)*

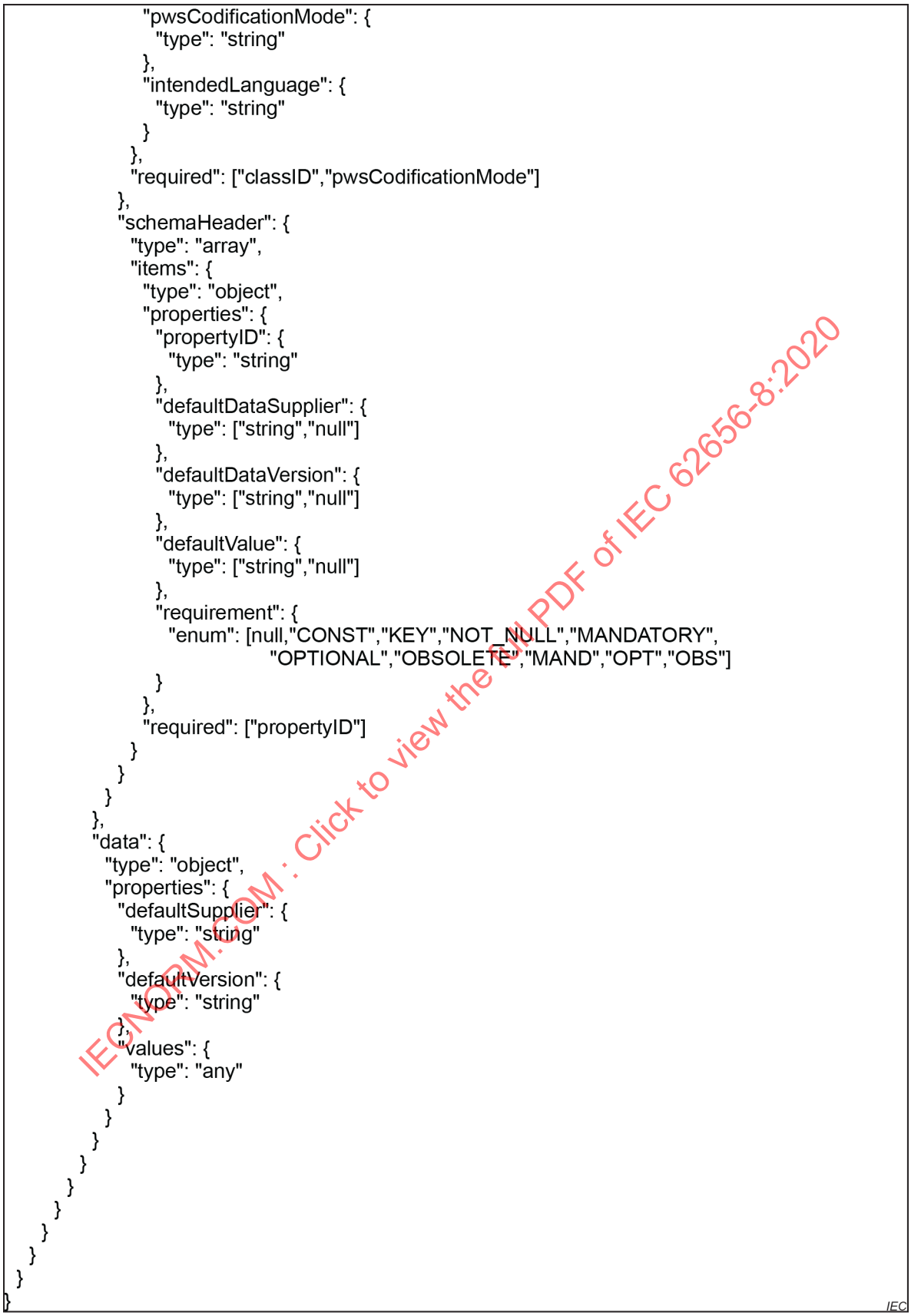


Figure A.1 *(2 of 2)*

A.1.2 Lateral JSON schema

The JSON schema for the lateral JSON notation of data parcels specified in Clause 7 is depicted in Figure A.2.

```
"$schema": "http://json-schema.org/schema#",
"type": "object",
"properties": {
  "conjunctiveParcels": {
    "type": "object",
    "properties": {
      "ontLayer": {
        "enum": ["AO","MO","DO","DL"]
      },
      "parcels": {
        "type": "array",
        "items": {
          "type": "object",
          "properties": {
            "header": {
              "type": "object",
              "properties": {
                "classHeader": {
                  "type": "object",
                  "properties": {
                    "classID": {
                      "type": "string"
                    },
                    "parcelMode": {
                      "type": ["string", "null"]
                    },
                    "parcelID": {
                      "type": ["string", "null"]
                    },
                    "defaultSupplier": {
                      "type": ["string", "null"]
                    },
                    "defaultVersion": {
                      "type": ["string", "null"]
                    },
                    "objectIdName": {
                      "type": ["string", "null"]
                    },
                    "idEncoding": {
                      "type": ["string", "null"]
                    },
                    "pwsCodificationMode": {
                      "type": "string"
                    },
                    "intendedLanguage": {
                      "type": "string"
                    }
                  },
                  "required": ["classID","pwsCodificationMode"]
                },
```

IEC

Figure A.2 – Lateral JSON schema *(1 of 2)*

```
            "schemaHeader": {
              "type": "array",
              "items": {
                "type": "object",
                "properties": {
                  "propertyID": {
                    "type": "string"
                  },
                  "defaultDataSupplier": {
                    "type": ["string","null"]
                  },
                  "defaultDataVersion": {
                    "type": ["string","null"]
                  },
                  "defaultValue": {
                    "type": ["string","null"]
                  },
                  "requirement": {
                    "enum": [null,"CONST","KEY","NOT_NULL","MANDATORY",
                             "OPTIONAL","OBSOLETE","MAND","OPT","OBS"]
                  }
                },
                "required": ["propertyID"]
              }
            }
          }
        },
        "data": {
          "type": "object",
          "properties": {
            "defaultSupplier": {
              "type": "string"
            },
            "defaultVersion": {
              "type": "string"
            },
            "instances": {
              "type": "array",
              "items": {
                "type": "array",
                "items": {
                  "type": "any"
                }
              }
            }
          }
        }
      }
    }
  }
}
```

IEC

Figure A.2 *(2 of 2)*

A.1.3 Exception JSON schema

The JSON schema for the JSON notation of the exceptions specified in 5.2 is depicted in Figure A.3.

```
{
  "$schema": "http://json-schema.org/schema#",
  "type": "object",
  "properties": {
    "code": {
      "type": "string"
    },
    "name": {
      "type": "string"
    },
    "description": {
      "type": "string"
    }
  }
}
```

IEC

Figure A.3 – Exception JSON schema

A.2 XML schema

A.2.1 Vertical XML schema

The XML schema [8] for the vertical XML notation of data parcels specified in Clause 8 is depicted in Figure A.4.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="urn:iec:std:iec:62656:-8:ed-1:xml-schema:vertical"
    elementFormDefault="qualified"
    xmlns="urn:iec:std:iec:62656:-8:ed-1:xml-schema:vertical"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:pws_v="urn:iec:std:iec:62656:-8:ed-1:xml-schema:vertical">
 <xsd:element name="conjunctiveParcels" type="conjunctiveParcelsType"/>
 <xsd:element name="parcel" type="parcelType"/>
 <xsd:element name="header" type="headerType"/>
 <xsd:element name="data" type="dataType"/>
 <xsd:element name="classHeader" type=" classHeaderType"/>
 <xsd:element name="schemaHeader" type="schemaHeaderType"/>
 <xsd:element name="property" type="propertyType"/>
 <xsd:element name="operations" type="operationsType"/>
 <xsd:element name="values" type="valuesType"/>
 <xsd:element name="classID" type="xsd:string"/>
 <xsd:element name="parcelMode" type="xsd:string"/>
 <xsd:element name="parcelID" type="xsd:string"/>
 <xsd:element name="defaultSupplier" type="xsd:string"/>
 <xsd:element name="defaultVersion" type="xsd:string"/>
 <xsd:element name="objectIdName" type="xsd:string"/>
 <xsd:element name="idEncoding" type="xsd:string"/>
 <xsd:element name="pwsCodificationMode">
  <xsd:simpleType>
   <xsd:restriction base="xsd:string">
    <xsd:enumeration value="VERTICAL"/>
    <xsd:enumeration value="LATERAL"/>
   </xsd:restriction>
  </xsd:simpleType>
 </xsd:element>
 <xsd:element name=”intendedLanguage” type=”xsd:string”/>
 <xsd:element name="propertyID" type="xsd:string"/>
 <xsd:element name="defaultDataSupplier" type="xsd:string"/>
 <xsd:element name="defaultDataVersion" type="xsd:string"/>
 <xsd:element name=”defaultValue” type=”xsd:string”/>
 <xsd:element name="requirement">
  <xsd:simpleType>
   <xsd:restriction base="xsd:string">
    <xsd:enumeration value=""/>
    <xsd:enumeration value="CONST"/>
    <xsd:enumeration value="KEY"/>
    <xsd:enumeration value="NOT_NULL"/>
    <xsd:enumeration value="MANDATORY"/>
    <xsd:enumeration value="OPTIONAL"/>
    <xsd:enumeration value="OBSOLETE"/>
    <xsd:enumeration value="MAND"/>
    <xsd:enumeration value="OPT"/>
    <xsd:enumeration value="OBS"/>
   </xsd:restriction>
  </xsd:simpleType>
 </xsd:element>
```

IEC

Figure A.4 – Vertical XML schema *(1 of 3)*

```
<xsd:complexType name="conjunctiveParcelsType">
  <xsd:sequence minOccurs="0" maxOccurs="unbounded">
    <xsd:element name="parcel" type="pws_v:parcelType"/>
  </xsd:sequence>
  <xsd:attribute name="ontLayer" type="pws_v:ontLayerType"/>
</xsd:complexType>
<xsd:simpleType name="ontLayerType">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="AO"/>
    <xsd:enumeration value="MO"/>
    <xsd:enumeration value="DO"/>
    <xsd:enumeration value="DL"/>
  </xsd:restriction>
</xsd:simpleType>
<xsd:complexType name="parcelType">
  <xsd:sequence>
    <xsd:element ref="pws_v:header"/>
    <xsd:element ref="pws_v:data"/>
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="headerType">
  <xsd:sequence>
    <xsd:element ref="pws_v:classHeader"/>
    <xsd:element ref="pws_v:schemaHeader"/>
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="classHeaderType">
  <xsd:sequence>
    <xsd:element ref="pws_v:classID"/>
    <xsd:element ref="pws_v:parcelMode" minOccurs="0"/>
    <xsd:element ref="pws_v:parcelID" minOccurs="0"/>
    <xsd:element ref="pws_v:defaultSupplier" minOccurs="0"/>
    <xsd:element ref="pws_v:defaultVersion" minOccurs="0"/>
    <xsd:element ref="pws_v:objectIdName" minOccurs="0"/>
    <xsd:element ref="pws_v:idEncoding" minOccurs="0"/>
    <xsd:element ref="pws_v:pwsCodificationMode"/>
    <xsd:element ref=”pws_v:intendedLanguage”/>
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="schemaHeaderType">
  <xsd:sequence>
    <xsd:element ref="pws_v:property" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="propertyType">
  <xsd:sequence>
    <xsd:element ref="pws_v:propertyID"/>
    <xsd:element ref="pws_v:defaultDataSupplier" minOccurs="0"/>
    <xsd:element ref="pws_v:defaultDataVersion" minOccurs="0"/>
    <xsd:element ref=”pws_v:defaultValue” minOccurs=”0”/>
    <xsd:element ref="pws_v:requirement" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>
```

IEC

Figure A.4 *(2 of 3)*

```
 <xsd:complexType name="dataType">
  <xsd:sequence>
   <xsd:element ref="pws_v:operations"/>
   <xsd:element ref="pws_v:values" minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute name="defaultSupplier" type="xsd:string"/>
  <xsd:attribute name="defaultVersion" type="xsd:string"/>
 </xsd:complexType>
 <xsd:complexType name="operationsType">
  <xsd:sequence>
   <xsd:element name="operation" type="xsd:string" minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
 </xsd:complexType>
 <xsd:complexType name="valuesType">
  <xsd:sequence>
   <xsd:element name="value" type="xsd:string" minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute name="propertyID" type="xsd:string"/>
 </xsd:complexType>
</xsd:schema>
```

IEC

Figure A.4 *(3 of 3)*

A.2.2 Lateral XML schema

The XML schema for the lateral XML notation of data parcels specified in Clause 8 is depicted in Figure A.5.

```
<?xml version="1.0" encoding="UTF-8"?>
<xsd:schema targetNamespace="urn:iec:std:iec:62656:-8:ed-1:xml-schema:lateral"
    elementFormDefault="qualified"
    xmlns="urn:iec:std:iec:62656:-8:ed-1:xml-schema:lateral"
    xmlns:xsd="http://www.w3.org/2001/XMLSchema"
    xmlns:pws_l="urn:iec:std:iec:62656:-8:ed-1:xml-schema:lateral">

 <xsd:element name="conjunctiveParcels" type="conjunctiveParcelsType"/>
 <xsd:element name="parcel" type="parcelType"/>
 <xsd:element name="header" type="headerType"/>
 <xsd:element name="data" type="dataType"/>
 <xsd:element name="classHeader" type="classHeaderType"/>
 <xsd:element name="schemaHeader" type="schemaHeaderType"/>
 <xsd:element name="property" type="propertyType"/>
 <xsd:element name="instance" type="instanceType"/>
 <xsd:element name="classID" type="xsd:string"/>
 <xsd:element name="parcelMode" type="xsd:string"/>
 <xsd:element name="parcelID" type="xsd:string"/>
 <xsd:element name="defaultSupplier" type="xsd:string"/>
 <xsd:element name="defaultVersion" type="xsd:string"/>
 <xsd:element name="objectIdName" type="xsd:string"/>
 <xsd:element name="idEncoding" type="xsd:string"/>
 <xsd:element name="pwsCodificationMode">
  <xsd:simpleType>
   <xsd:restriction base="xsd:string">
    <xsd:enumeration value="VERTICAL"/>
    <xsd:enumeration value="LATERAL"/>
   </xsd:restriction>
  </xsd:simpleType>
 </xsd:element>
 <xsd:element name=”intendedLanguage” type=”xsd:string”/>
 <xsd:element name="propertyID" type="xsd:string"/>
 <xsd:element name="defaultDataSupplier" type="xsd:string"/>
 <xsd:element name="defaultDataVersion" type="xsd:string"/>
 <xsd:element name=”defaultValue” type=”xsd:string”/>
 <xsd:element name="requirement">
  <xsd:simpleType>
   <xsd:restriction base="xsd:string">
    <xsd:enumeration value=""/>
    <xsd:enumeration value="CONST"/>
    <xsd:enumeration value="KEY"/>
    <xsd:enumeration value="NOT_NULL"/>
    <xsd:enumeration value="MANDATORY"/>
    <xsd:enumeration value="OPTIONAL"/>
    <xsd:enumeration value="OBSOLETE"/>
    <xsd:enumeration value="MAND"/>
    <xsd:enumeration value="OPT"/>
    <xsd:enumeration value="OBS"/>
   </xsd:restriction>
  </xsd:simpleType>
 </xsd:element>
 <xsd:element name="value" type="valueType"/>
```

IEC

Figure A.5 – Lateral XML schema *(1 of 3)*

```
<xsd:complexType name="conjunctiveParcelsType">
  <xsd:sequence minOccurs="0" maxOccurs="unbounded">
    <xsd:element name="parcel" type="pws_l:parcelType"/>
  </xsd:sequence>
  <xsd:attribute name="ontLayer" type="pws_l:ontLayerType"/>
</xsd:complexType>
<xsd:simpleType name="ontLayerType">
  <xsd:restriction base="xsd:string">
    <xsd:enumeration value="AO"/>
    <xsd:enumeration value="MO"/>
    <xsd:enumeration value="DO"/>
    <xsd:enumeration value="DL"/>
  </xsd:restriction>
</xsd:simpleType>
<xsd:complexType name="parcelType">
  <xsd:sequence>
    <xsd:element ref="pws_l:header"/>
    <xsd:element ref="pws_l:data"/>
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="headerType">
  <xsd:sequence>
    <xsd:element ref="pws_l:classHeader"/>
    <xsd:element ref="pws_l:schemaHeader"/>
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="classHeaderType">
  <xsd:sequence>
    <xsd:element ref="pws_l:classID"/>
    <xsd:element ref="pws_l:parcelMode" minOccurs="0"/>
    <xsd:element ref="pws_l:parcelID" minOccurs="0"/>
    <xsd:element ref="pws_l:defaultSupplier" minOccurs="0"/>
    <xsd:element ref="pws_l:defaultVersion" minOccurs="0"/>
    <xsd:element ref="pws_l:objectIdName" minOccurs="0"/>
    <xsd:element ref="pws_l:idEncoding" minOccurs="0"/>
    <xsd:element ref="pws_l:pwsCodificationMode"/>
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="schemaHeaderType">
  <xsd:sequence>
    <xsd:element ref="pws_l:property" maxOccurs="unbounded"/>
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="propertyType">
  <xsd:sequence>
    <xsd:element ref="pws_l:propertyID"/>
    <xsd:element ref="pws_l:defaultDataSupplier" minOccurs="0"/>
    <xsd:element ref="pws_l:defaultDataVersion" minOccurs="0"/>
    <xsd:element ref="pws_l:requirement" minOccurs="0"/>
  </xsd:sequence>
</xsd:complexType>
<xsd:complexType name="dataType">
  <xsd:sequence>
    <xsd:element ref="pws_l:instance" minOccurs="0" maxOccurs="unbounded"/>
  </xsd:sequence>
  <xsd:attribute name="defaultSupplier" type="xsd:string"/>
  <xsd:attribute name="defaultVersion" type="xsd:string"/>
</xsd:complexType>
```

IEC

Figure A.5 *(2 of 3)*

```
  <xsd:complexType name="instanceType">
    <xsd:sequence>
      <xsd:element name="operation" type="xsd:string"/>
      <xsd:element ref="pws_l:value" minOccurs="1" maxOccurs="unbounded"/>
    </xsd:sequence>
  </xsd:complexType>
  <xsd:complexType name="valueType">
    <xsd:simpleContent>
      <xsd:extension base="xsd:string">
        <xsd:attribute name="propertyID" type="xsd:string"/>
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
</xsd:schema>
```

IEC

Figure A.5 *(3 of 3)*

A.2.3 Exception XML schema

The XML schema for the XML notation of the exceptions specified in 5.2 is depicted in Figure A.6.

```
  <xsd:complexType name="valueType">
    <xsd:simpleContent>
      <xsd:extension base="xsd:string">
        <xsd:attribute name="propertyID" type="xsd:string"/>
      </xsd:extension>
    </xsd:simpleContent>
  </xsd:complexType>
</xsd:schema>
```

IEC

Figure A.6 – Exception XML schema

Annex B
(normative)

Web service representation

B.1 Web service representation in WADL

The web service representation in WADL is depicted in Figure B.1. A status code of each response follows the specification in [9].

```
<?xml version="1.0"?>
<application xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://wadl.dev.java.net/2009/02 wadl.xsd"
 xmlns:tns="urn:iec:std:iec:62656:-8:ed-1:wadl"
 xmlns:xsd="http://www.w3.org/2001/XMLSchema"
 xmlns="http://wadl.dev.java.net/2009/02"
 xmlns:pws_v="urn:iec:std:iec:62656:-8:ed-1:xml-schema:vertical"
 xmlns:pws_l="urn:iec:std:iec:62656:-8:ed-1:xml-schema:lateral">
 <grammars>
  <include href="ParcelWebService/vertical.xsd"/>
  <include href="ParcelWebService/lateral.xsd"/>
  <include href="ParcelWebService/exception.xsd"/>
 </grammars>

 <resources base="http://std.iec.ch/iec61360/pws/v1/">
  <resource path="register/{format}">
   <param xmlns:xsd=http://www.w3.org/2001/XMLSchema
          name="format" style="template" type="xsd:string">
    <option value="xml"/>
    <option value="json"/>
   </param>
   <method name="POST" id="register">
    <request>
     <param name="data" type="xsd:string" style="plain" required="true"/>
     <representation mediaType="application/xml" element="pws_v:conjunctiveParcels"/>
     <representation mediaType="application/xml" element="pws_l:conjunctiveParcels"/>
    </request>
    <response status="200">
     <representation mediaType="application/xml"/>
     <representation mediaType="application/json"/>
    </response>
    <response status="404">
     <representation mediaType="application/xml"
                     element="pws_e:notImplementedExceptionType"/>
     <representation mediaType="application/json"/>
    </response>
    <response status="409">
     <representation mediaType="application/xml"
                     element="pws_e:elementAlreadyExistExceptionType"/>
     <representation mediaType="application/json"/>
    </response>
    <response status="403">
     <representation mediaType="application/xml"
                     element="pws_e:elementNotFoundExceptionType"/>
     <representation mediaType="application/json"/>
    </response>
```

IEC

Figure B.1 – Web service representation in WADL *(1 of 4)*

```
    <response status="409">
      <representation mediaType="application/xml"
                      element="pws_e:newerElementExistExceptionType"/>
      <representation mediaType="application/json"/>
    </response>
    <response status="400">
      <representation mediaType="application/xml"
                      element="pws_e:invalidParameterExceptionType"/>
      <representation mediaType="application/json"/>
    </response>
    <response status="400">
      <representation mediaType="application/xml"
                      element="pws_e:invalidDataParcelExceptionType"/>
      <representation mediaType="application/json"/>
    </response>
    <response status="400">
      <representation mediaType="application/xml"
                      element="pws_e:validationFailureExceptionType"/>
      <representation mediaType="application/json"/>
    </response>
    <response status="403">
      <representation mediaType="application/xml"
                      element="pws_e:notPermittedExceptionType"/>
      <representation mediaType="application/json"/>
    </response>
  </method>
</resource>
<resource path="resolve/{format}">
  <param xmlns:xsd=http://www.w3.org/2001/XMLSchema
         name="format" style="template" type="xsd:string">
    <option value="xml"/>
    <option value="json"/>
  </param>
  <method name="GET" id="resolve">
    <request>
      <param name="requestKind" style="header" default="DEFINITION">
        <option value="DEFINITION"/>
        <option value="INSTANCE"/>
      </param>
      <param name="keywordKind" style="header" default="ID">
        <option value="ID"/>
        <option value="NAME"/>
      </param>
      <param name="keyword" type="xsd:string" style="header" required="true"/>
      <param name="language" type="xsd:string" style="header"/>
      <param name=”dictionary_identifier” type=”xsd:string” style=”header”/>
      <param name="pwsCodificationMode" style="header" default="VERTICAL">
        <option value="VERTICAL"/>
        <option value="LATERAL"/>
      </param>
      <param name=”notation_format” type=”xsd:string” style=”header”>
        <option value=”JSON”/>
        <option value=”XML”/>
      </param>
      <param name="maxResult" type="xsd:unsignedInt" style="header"/>
    </request>
```

IEC

Figure B.1 *(2 of 4)*

```
    <response status="200">
      <param name="conjunctiveParcels" style="plain"/>
     <representation mediaType="application/xml" element="pws_v:conjunctiveParcels"/>
     <representation mediaType="application/xml" element="pws_l:conjunctiveParcels"/>
     <representation mediaType="application/json"/>
    </response>

    <response status="404">
     <representation mediaType="application/xml"
                     element="pws_e:notImplementedExceptionType"/>
     <representation mediaType="application/json"/>
    </response>
    <response status="403">
     <representation mediaType="application/xml"
                     element="pws_e:elementNotFoundExceptionType"/>
     <representation mediaType="application/json"/>
    </response>
    <response status="400">
     <representation mediaType="application/xml"
                     element="pws_e:invalidParameterExceptionType"/>
     <representation mediaType="application/json"/>
    </response>
    <response status="400">
     <representation mediaType="application/xml"
                     element="pws_e:invalidDataParcelExceptionType"/>
     <representation mediaType="application/json"/>
    </response>
    <response status="403">
     <representation mediaType="application/xml"
                     element="pws_e:notPermittedExceptionType"/>
     <representation mediaType="application/json"/>
    </response>
  </method>
</resource>
<resource path="subscribe/{format}">
  <param xmlns:xsd=http://www.w3.org/2001/XMLSchema
         name="format" style="template" type="xsd:string">
    <option value="xml"/>
    <option value="json"/>
  </param>
  <method name="POST" id="subscribe">
    <request>
      <param name="requestKind" style="header" default="DEFINITION">
        <option value="DEFINITION"/>
        <option value="INSTANCE"/>
      </param>
      <param name="identifiers" type="xsd:string" style="header" required="true"/>
      <param name="subscriptionKind" style="header" default="START">
        <option value="START"/>
        <option value="END"/>
      </param>
      <param name=”expiration_date” type=”xsd:string” style=”header”/>
    </request>
    <response status="200">
      <representation mediaType="application/xml"/>
      <representation mediaType="application/json"/>
    </response>
```

IEC

Figure B.1 *(3 of 4)*

```
      <response status="404">
        <representation mediaType="application/xml"
                        element="pws_e:notImplementedExceptionType"/>
        <representation mediaType="application/json"/>
      </response>
      <response status="403">
        <representation mediaType="application/xml"
                        element="pws_e:elementNotFoundExceptionType"/>
        <representation mediaType="application/json"/>
      </response>
      <response status="400">
        <representation mediaType="application/xml"
                        element="pws_e:invalidParameterExceptionType"/>
        <representation mediaType="application/json"/>
      </response>
      <response status="400">
        <representation mediaType="application/xml"
                        element="pws_e:invalidDataParcelExceptionType"/>
        <representation mediaType="application/json"/>
      </response>
      <response status="403">
        <representation mediaType="application/xml"
                        element="pws_e:notPermittedExceptionType"/>
        <representation mediaType="application/json"/>
      </response>
    </method>
  </resource>
 </resources>
</application>
```

IEC

Figure B.1 *(4 of 4)*

B.2 Web service representation in WSDL

The web service representation in WSDL is depicted in Figure B.2.

```
      <response status="404">
        <representation mediaType="application/xml"
                        element="pws_e:notImplementedExceptionType"/>
        <representation mediaType="application/json"/>
      </response>
      <response status="403">
        <representation mediaType="application/xml"
                        element="pws_e:elementNotFoundExceptionType"/>
        <representation mediaType="application/json"/>
      </response>
      <response status="400">
        <representation mediaType="application/xml"
                        element="pws_e:invalidParameterExceptionType"/>
        <representation mediaType="application/json"/>
      </response>
      <response status="400">
        <representation mediaType="application/xml"
                        element="pws_e:invalidDataParcelExceptionType"/>
        <representation mediaType="application/json"/>
      </response>
      <response status="403">
        <representation mediaType="application/xml"
                        element="pws_e:notPermittedExceptionType"/>
        <representation mediaType="application/json"/>
      </response>
    </method>
  </resource>
 </resources>
</application>
```

IEC

Figure B.2 – Web service representation in WSDL *(1 of 4)*

```
    <xsd:complexType name="subscriptRequest">
      <xsd:sequence>
        <xsd:element name="requestKind"
                     type="tns:requestKindType" default="DEFINITION"/>
        <xsd:element name="identifiers" type="xsd:string"/>
        <xsd:element name="subscriptionOperationKind"
                     type="tns:subscriptionOperationKindType" default="START"/>
        <xsd:element name=”expiration_date” type=”xsd:string”/>
      </xsd:sequence>

    </xsd:complexType>
    <xsd:complexType name="subscribeResponse">
      <xsd:sequence>
        <xsd:element name="operationResult" type="xsd:string"/>
      </xsd:sequence>
    </xsd:complexType>

    <xsd:simpleType name="requestTypeType">
      <xsd:restriction base="xsd:string">
        <xsd:enumeration value="DEFINITION"/>
        <xsd:enumeration value="INSTANCE"/>
      </xsd:restriction>
    </xsd:simpleType>

    <xsd:simpleType name="pwsCodificationModeType">
      <xsd:restriction base="xsd:string">
        <xsd:enumeration value="VERTICAL"/>
        <xsd:enumeration value="LATERAL"/>
      </xsd:restriction>
    </xsd:simpleType>

    <xsd:simpleType name="notationFormatType">
      <xsd:restriction base="xsd:string">
        <xsd:enumeration value="JSON"/>
        <xsd:enumeration value="XML"/>
      </xsd:restriction>
    </xsd:simpleType>

    <xsd:complexType name="conjunctiveParcelsType">
      <xsd:choice>
        <xsd:element ref="pws_v:conjunctiveParcels"/>
        <xsd:element ref="pws_l:conjunctiveParcels"/>
      </xsd:choice>
    </xsd:complexType>

    <xsd:simpleType name="keywordKindType">
      <xsd:restriction base="xsd:string">
        <xsd:enumeration value="ID"/>
        <xsd:enumeration value="NAME"/>
      </xsd:restriction>

    </xsd:simpleType>
    <xsd:simpleType name="subscriptionOperationKindType">
      <xsd:restriction base="xsd:string">
        <xsd:enumeration value="START"/>
        <xsd:enumeration value="END"/>
      </xsd:restriction>
    </xsd:simpleType>
  </xsd:schema>
</wsdl:types>
```

IEC

Figure B.2 *(2 of 4)*

```
<!--  Messages -->
 <wsdl:message name="registerRequest">
  <wsdl:part name="parameters" type="tns:registerRequest"/>
 </wsdl:message>
 <wsdl:message name="registerResponse">
  <wsdl:part name="parameters" type="tns:registerResponse"/>
 </wsdl:message>
 <wsdl:message name="resolveRequest">
  <wsdl:part name="parameters" type="tns:resolveRequest"/>
 </wsdl:message>
 <wsdl:message name="resolveResponse">
  <wsdl:part name="parameters" type="tns:resolveResponse"/>
 </wsdl:message>
 <wsdl:message name="subscribeRequest">
  <wsdl:part name="parameters" type="tns:subscribeRequest"/>
 </wsdl:message>
 <wsdl:message name="subscribeResponse">
  <wsdl:part name="parameters" type="tns:subscribeResponse"/>
 </wsdl:message>

 <!-- Faults -->
 <wsdl:message name="notImplementedFault">
  <wsdl:part name="notImplementedException"
             element="pws_e:notImplementedException"/>
 </wsdl:message>
 <wsdl:message name="elementAlreadyExistFault">
  <wsdl:part name="elementAlreadyExistException"
             element="pws_e:elementAlreadyExistException"/>
 </wsdl:message>
 <wsdl:message name="elementNotFoundFault">
  <wsdl:part name="elementNotFoundException"
             element="pws_e:elementNotFoundException"/>
 </wsdl:message>
 <wsdl:message name="newerElementExistFault">
  <wsdl:part name="newerElementExistException"
             element="pws_e:newerElementExistException"/>
 </wsdl:message>
 <wsdl:message name="invalidParameterFault">
  <wsdl:part name="invalidParameterException"
             element="pws_e:invalidParameterException"/>
 </wsdl:message>
 <wsdl:message name="invalidDataParcelFault">
  <wsdl:part name="invalidDataParcelException"
             element="pws_e:invalidDataParcelException"/>
 </wsdl:message>
 <wsdl:message name="validationFailureFault">
  <wsdl:part name="validationFailureException"
             element="pws_e:validationFailureException"/>
 </wsdl:message>
 <wsdl:message name="notPermittedFault">
  <wsdl:part name="notPermittedException" element="pws_e:notPermittedException"/>
 </wsdl:message>
```

IEC

Figure B.2 *(3 of 4)*

```
<!-- Port types -->
 <wsdl:portType name="parcelWebServicePortType">
  <wsdl:operation name="register">
   <wsdl:input message="tns:registerRequest" name="registerRequest"/>
   <wsdl:output message="tns:registerResponse" name="registerResponse"/>
   <wsdl:fault name="notImplementedFault" message="tns:notImplementedFault"/>
   <wsdl:fault name="elementAlreadyExistFault"
               message="tns:elementAlreadyExistFault"/>
   <wsdl:fault name="elementNotFoundFault" message="tns:elementNotFoundFault"/>
   <wsdl:fault name="newerElementExistFault" message="tns:newerElementExistFault"/>
   <wsdl:fault name="invalidParameterFault" message="tns:invalidParameterFault"/>
   <wsdl:fault name="invalidDataParcelFault" message="tns:invalidDataParcelFault"/>
   <wsdl:fault name="validationFailureFault" message="tns:validationFailureFault"/>
   <wsdl:fault name="notPermittedFault" message="tns:notPermittedFault"/>
  </wsdl:operation>

  <wsdl:operation name="resolve">
   <wsdl:input message="tns:resolveRequest" name="resolveRequest"/>
   <wsdl:output message="tns:resolveResponse" name="resolveResponse"/>
   <wsdl:fault name="notImplementedFault" message="tns:notImplementedFault"/>
   <wsdl:fault name="elementNotFoundFault" message="tns:elementNotFoundFault"/>
   <wsdl:fault name="invalidParameterFault" message="tns:invalidParameterFault"/>
   <wsdl:fault name="invalidDataParcelFault" message="tns:invalidDataParcelFault"/>
   <wsdl:fault name="notPermittedFault" message="tns:notPermittedFault"/>
  </wsdl:operation>

  <wsdl:operation name="subscribe">
   <wsdl:input message="tns:subscribeRequest" name="subscribeRequest"/>
   <wsdl:output message="tns:subscribeResponse" name="subscribeResponse"/>
   <wsdl:fault name="notImplementedFault" message="tns:notImplementedFault"/>
   <wsdl:fault name="elementNotFoundFault" message="tns:elementNotFoundFault"/>
   <wsdl:fault name="invalidParameterFault" message="tns:invalidParameterFault"/>
   <wsdl:fault name="invalidDataParcelFault" message="tns:invalidDataParcelFault"/>
   <wsdl:fault name="notPermittedFault" message="tns:notPermittedFault"/>
  </wsdl:operation>
 </wsdl:portType>
```

IEC

Figure B.2 *(4 of 4)*

Annex C
(informative)

Examples of data representation

C.1 Example data parcel

A data parcel for describing examples of data representation in JSON or XML notations is depicted in Figure C.1. This data parcel is a class sheet, which comprises four cell columns and contains 4 classes as instances of the class meta-class. Those 4 classes are taken from an upper part of the class structure given in IEC 61360-4 as at 2017-10-01, in order to simplify the explanation and help readers to understand examples given in Annex C.

Table C.1 – Example data parcel

#CLASS_ID:= MDC_C002				
#DEFAULT_ SUPPLIER:= 0112/2///62656_1				
#DEFAULT_ VERSION:=1				
#INTENDED_LANGUAGE:=en				
#PROPERTY_ ID	MDC_ P001_5	MDC_ P004_1.en	MDC_ P010	MDC_P014
#DEFAULT_ DATA_SUPPLIER	0112/2///61 360_4		0112/2///61 360_4	0112/2///61360_4
#DEFAULT_ DATA_VERSION	1		1	1
#DEFAULT_VALUE			UNIVERSE	
	AAA001	component	UNIVERSE	{AAE001,AAE006,…}
	AAA002	electric/ electronic component	AAA001	{AAE002,AAE007,…}
	AAA003	amplifier	AAA002	{AAE697,AAE969,…}
	AAA013	antenna	AAA002	{AAE340,AAE341,…}

C.2 Example of data representation in JSON notation

C.2.1 Example of data representation in vertical JSON notation

An example of data representation in the vertical JSON notation with respect to the data parcel in Table C.1 is depicted in Figure C.1.

```
{
 "conjunctiveParcels": {
  "ontLayer": "DO",
  "parcels": [
   {
    "header": {
     "classHeader": {
      "classID": "MDC_C002",
      "defaultSupplier": "0112/2///62656_1",
      "defaultVersion": "1",
      “intendedLanguage”: “en”,
      "pwsCodificationMode": "VERTICAL"
     },
     "schemaHeader": [
      {
       "propertyID": "MDC_P001_5",
       "defaultDataSupplier": "0112/2///61360_4",
       "defaultDataVersion": "1",
       “defaultValue”: null
      },
      {
       "propertyID": "MDC_P004_1.en",
       "defaultDataSupplier": null,
       "defaultDataVersion": null,
       “defaultValue”: null
      },
      {
       "propertyID": "MDC_P010",
       "defaultDataSupplier": "0112/2///61360_4",
       "defaultDataVersion": "1",
       “defaultValue”: “UNIVERSE”
      },
      {
       "propertyID": "MDC_P014",
       "defaultDataSupplier": "0112/2///61360_4",
       "defaultDataVersion": "1",
       “defaultValue”: null
      }
     ]
    },
    "data": {
     "defaultSupplier": "0112/2///62656_1",
     "defaultVersion": "1",
     "values": {
      "operations": [null, null, null, null],
      "MDC_P001_5": ["AAA001", "AAA002", "AAA003", "AAA013"],
      "MDC_P004_1.en": ["component", "electric/electronic component",
            "amplifier", "antenna"],
      "MDC_P010": ["UNIVERSE", "AAA001", "AAA002", "AAA002"],
      "MDC_P014": ["{AAE001, AAE006,...}", "{AAE002, AAE007,...}",
           "{AAE697, AAE969,...}", "{AAE340, AAE341,...}"]
     }
    }
   }
  ]
 }
}
```

IEC

Figure C.1 – Example of data representation in vertical JSON notation

C.2.2 Example of data representation in lateral JSON notation

An example of data representation in the lateral JSON notation to the data parcel in Table C.1 is depicted in Figure C.2.

```
{
 "conjunctiveParcels": {
  "ontLayer": "DO",
  "parcels": [
   {
    "header": {
     "classHeader": {
      "classID": "MDC_C002",
      "defaultSupplier": "0112/2///62656_1",
      "defaultVersion": "1",
      “intendedLanguage”: “en”,
      "pwsCodificationMode": "LATERAL"
     },
     "schemaHeader": [
      {
       "propertyID": "MDC_P001_5",
       "defaultDataSupplier": "0112/2///61360_4",
       "defaultDataVersion": "1",
       “defaultValue”: null
      },
      {
       "propertyID": "MDC_P004_1.en",
       "defaultDataSupplier": null,
       "defaultDataVersion": null,
       “defaultValue”: null
      },
      {
       "propertyID": "MDC_P010",
       "defaultDataSupplier": "0112/2///61360_4",
       "defaultDataVersion": "1",
       “defaultValue”: “UNIVERSE”
      },
      {
       "propertyID": "MDC_P014",
       "defaultDataSupplier": "0112/2///61360_4",
       "defaultDataVersion": "1",
       “defaultValue”: null
      }
     ]
    },
    "data": {
     "instances": [
      [null,"AAA001","component","UNIVERSE", "{AAE001, AAE006,...}"],
      [null,"AAA002","electric/ electronic component","AAA001","{AAE002, AAE007,...}"],
      [null,"AAA003","amplifier","AAA002","{AAE697, AAE969,...}"],
      [null,"AAA013","antenna","AAA002", "{AAE340, AAE341,...}"]
     ]
    }
   }
  ]
 }
}
```

IEC

Figure C.2 – Example of data representation in lateral JSON notation

C.3 Example of data representation in XML notation

C.3.1 Example of data representation in vertical XML notation

An example of data representation in the vertical XML notation with respect to the data parcel in Table C.1 is depicted in Figure C.3.

```
<?xml version="1.0" encoding="utf-8"?>
<conjunctiveParcels ontLayer="DO"
 xmlns="urn:iec:std:iec:62656:-8:ed-1:xml-schema:vertical"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:iec:std:iec:62656:-8:ed-1:xml-schema:vertical vertical.xsd">
 <parcel>
  <header>
   <classHeader>
    <classID>MDC_C002</classID>
    <defaultSupplier>0112/2///62656_1</defaultSupplier>
    <defaultVersion>1</defaultVersion>
    <intendedLanguage>en</intendedLanguage>
    <pwsCodificationMode>VERTICAL</pwsCodificationMode>
   </classHeader>
   <schemaHeader>
    <property>
     <propertyID>MDC_P001_5</propertyID>
     <defaultDataSupplier>0112/2///61360_4</defaultDataSupplier>
     <defaultDataVersion>1</defaultDataVersion>
    </property>
    <property>
     <propertyID>MDC_P004_1.en</propertyID>
    </property>
    <property>
     <propertyID>MDC_P010</propertyID>
     <defaultDataSupplier>0112/2///61360_4</defaultDataSupplier>
     <defaultDataVersion>1</defaultDataVersion>
     <defaultValue>UNIVERSE</defaultValue>
    </property>
    <property>
     <propertyID>MDC_P014</propertyID>
     <defaultDataSupplier>0112/2///61360_4</defaultDataSupplier>
     <defaultDataVersion>1</defaultDataVersion>
    </property>
   </schemaHeader>
  </header>
  <data defaultSupplier="0112/2///62656_1" defaultVersion="1">
   <operations>
    <operation/>
    <operation/>
    <operation/>
    <operation/>
   </operations>
   <values propertyID="MDC_P001_5">
    <value>AAA001</value>
    <value>AAA002</value>
    <value>AAA003</value>
    <value>AAA013</value>
   </values>
   <values propertyID="MDC_P004_1.en">
    <value>component</value>
    <value>electric/electronic component</value>
    <value>amplifier</value>
    <value>antenna</value>
   </values>
```

IEC

Figure C.3 – Example of data representation in vertical XML notation *(1 of 2)*

```
      <values propertyID="MDC_P010">
        <value>UNIVERSE</value>
        <value>AAA001</value>
        <value>AAA002</value>
        <value>AAA002</value>
      </values>
      <values propertyID="MDC_P014">
        <value>{AAE001,AAE006,...}</value>
        <value>{AAE002,AAE007,...}</value>
        <value>{AAE697,AAE969,...}</value>
        <value>{AAE340,AAE341,...}</value>
      </values>
    </data>
  </parcel>
</conjunctiveParcels>
```

IEC

Figure C.3 *(2 of 2)*

C.3.2 Example of data representation in lateral XML notation

An example of data representation in the lateral XML notation with respect to the data parcel in Table C.1 is depicted in Figure C.4.

```
<?xml version="1.0" encoding="utf-8"?>
<conjunctiveParcels ontLayer="DO"
 xmlns="urn:iec:std:iec:62656:-8:ed-1:xml-schema:lateral"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="urn:iec:std:iec:62656:-8:ed-1:xml-schema:lateral lateral.xsd">
 <parcel>
  <header>
   <classHeader>
    <classID>MDC_C002</classID>
    <defaultSupplier>0112/2///62656_1</defaultSupplier>
    <defaultVersion>1</defaultVersion>
    <intendedLanguage>en</intendedLanguage>
    <pwsCodificationMode>LATERAL</pwsCodificationMode>
   </classHeader>
   <schemaHeader>
    <property>
     <propertyID>MDC_P001_5</propertyID>
     <defaultDataSupplier>0112/2///61360_4</defaultDataSupplier>
     <defaultDataVersion>1</defaultDataVersion>
    </property>
    <property>
     <propertyID>MDC_P004_1.en</propertyID>
    </property>
    <property>
     <propertyID>MDC_P010</propertyID>
     <defaultDataSupplier>0112/2///61360_4</defaultDataSupplier>
     <defaultDataVersion>1</defaultDataVersion>
     <defaultValue>UNIVERSE</defaultValue>
    </property>
    <property>
     <propertyID>MDC_P014</propertyID>
     <defaultDataSupplier>0112/2///61360_4</defaultDataSupplier>
     <defaultDataVersion>1</defaultDataVersion>
    </property>
   </schemaHeader>
  </header>
  <data defaultSupplier="0112/2///62656_1" defaultVersion="1">
   <instance>
    <operation/>
    <value propertyID="MDC_P001_5">AAA001</value>
    <value propertyID="MDC_P004_1.en">component</value>
    <value propertyID="MDC_P010">UNIVERSE</value>
    <value propertyID="MDC_P014">{AAE001, AAE006,...}</value>
   </instance>
   <instance>
    <operation/>
    <value propertyID="MDC_P001_5">AAA002</value>
    <value propertyID="MDC_P004_1.en">electric/electronic component</value>
    <value propertyID="MDC_P010">AAA001</value>
    <value propertyID="MDC_P014">{AAE002, AAE007,...}</value>
   </instance>
```

IEC

Figure C.4 – Example of data representation in lateral XML notation *(1 of 2)*

```
      <instance>
        <operation/>
        <value propertyID="MDC_P001_5">AAA003</value>
        <value propertyID="MDC_P004_1.en">amplifier</value>
        <value propertyID="MDC_P010">AAA002</value>
        <value propertyID="MDC_P014">{AAE697, AAE969,...}</value>
      </instance>
      <instance>
        <operation/>
        <value propertyID="MDC_P001_5">AAA013</value>
        <value propertyID="MDC_P004_1.en">antenna</value>
        <value propertyID="MDC_P010">AAA002</value>
        <value propertyID="MDC_P014">{AAE340, AAE341,...}</value>
      </instance>
    </data>
  </parcel>
</conjunctiveParcels>
```

IEC

Figure C.4 *(2 of 2)*

Annex D
(informative)

Descriptions of the instructions of "optional – informative"

The descriptions of the instructions of "optional – informative" are listed in Table D.1. Each description follows the naming convention for instruction keywords which is specified in 6.4.

Table D.1 – Descriptions of the instructions of "optional – informative"

Instruction keyword specified in IEC 62656-1	Description in XML and JSON
#CLASS_NAME.<lang>	className.<lang>
#CLASS_DEFINITION.<lang>	classDefinition.<lang>
#CLASS_NOTE.<lang>	classNote.<lang>
#ALTERNATE_CLASSID	alternateClassID
#SUPER_ALT_CLASSID	superAltClassID
#SUB_ALT_CLASSID	subAltClassID
#SOURCE_LANGUAGE	sourceLanguage
#OBJECT_ID_NAME	objectIDName
#PROPERTY_NAME.<lang>	propertyName.<lang>
#DEFINITION.<lang>	definition.<lang>
#NOTE.<lang>	note.<lang>
#DATATYPE	datatype
#UNIT	unit
#ALTERNATIVE_UNITS	alternativeUnits
#VARIABLE_PREFIX_UNIT	variablePrefixUnit
#SUPER_PROPERTY	superProperty
#ALTERNATE_ID	alternateID
#SUPER_ALTERNATE_ID	superAlternateID
#EQUIVALENT_ID	equivalentID
#UNIT_ID	unitID
#VALUE_FORMAT	valueFormat
#DELIMITER	delimiter
#DECIMAL	decimal
#PATTERN	pattern
#RELATION	relation

Bibliography

[1] IEC 62656-2:2013, *Standardized product ontology register and transfer by spreadsheets – Part 2: Application guide for use with the IEC common data dictionary (CDD)*

[2] IEC 61360-2:2012, *Standard data element types with associated classification scheme for electric components – Part 2: EXPRESS dictionary schema*

[3] ISO 13584-42:2010, *Industrial automation systems and integration – Parts library – Part 42: Description methodology: Methodology for structuring parts families*

[4] Internet Engineering Task Force (IETF). *JSON Schema: A media type for describing JSON documents* [online]. Edited by H. Andrews and A. Wright. March 2018 [viewed 2018-09-01].
Available at http://json-schema.org/latest/json-schema-core.html

[5] W3C Recommendation. *Extensible Markup Language (XML) 1.1 (Second Edition)* [online]. Edited by T. Bray et al. September 2016 [viewed 2018-09-01].
Available at https://www.w3.org/TR/xml11/

[6] HADLEY, M., Sun Microsystems, Inc., W3C Member Submission. *Web Application Description Language* [online]. August 2009 [viewed 2018-09-01]. Available at https://www.w3.org/Submission/wadl/

[7] W3C Recommendation. *Web Services Description Language (WSDL) Version 2.0 Part 1: Core Language* [online]. Edited by R. Chinnici et al. June 2007 [viewed 2018-09-01]. Available at https://www.w3.org/TR/wsdl

[8] W3C Ubiquitous Web domain. *XML Schema* [online]. April 2000 [viewed 2018-09-01]. Available at https://www.w3.org/XML/Schema

[9] Internet Engineering Task Force (IETF). RFC 7231 – *Hypertext Transfer Protocol (HTTP/1.1): Semantics and Content* [online]. Edited by R. Fielding and J. Reschke June 2014 [viewed 2018-09-01]. Available at https://tools.ietf.org/html/rfc7231

[10] ISO 13584-24, *Industrial automation systems and integration – Parts library – Part 24: Logical resource: Logical model of supplier library*

SOMMAIRE

COMMISSION ÉLECTROTECHNIQUE INTERNATIONALE

_____

ENREGISTREMENT D'ONTOLOGIE DE PRODUITS NORMALISES ET TRANSFERT PAR PAQUETS DE DONNÉES –

Partie 8: Interface de service Web pour les paquets de données

AVANT-PROPOS

1) La Commission Électrotechnique Internationale (IEC) est une organisation mondiale de normalisation composée de l'ensemble des comités électrotechniques nationaux (Comités nationaux de l'IEC). L'IEC a pour objet de favoriser la coopération internationale pour toutes les questions de normalisation dans les domaines de l'électricité et de l'électronique. À cet effet, l'IEC – entre autres activités – publie des Normes internationales, des Spécifications techniques, des Rapports techniques, des Spécifications accessibles au public (PAS) et des Guides (ci-après dénommés "Publication(s) de l'IEC"). Leur élaboration est confiée à des comités d'études, aux travaux desquels tout Comité national intéressé par le sujet traité peut participer. Les organisations internationales, gouvernementales et non gouvernementales, en liaison avec l'IEC, participent également aux travaux. L'IEC collabore étroitement avec l'Organisation Internationale de Normalisation (ISO), selon des conditions fixées par accord entre les deux organisations.
2) Les décisions ou accords officiels de l'IEC concernant les questions techniques représentent, dans la mesure du possible, un accord international sur les sujets étudiés, étant donné que les Comités nationaux de l'IEC intéressés sont représentés dans chaque comité d'études.
3) Les Publications de l'IEC se présentent sous la forme de recommandations internationales et sont agréées comme telles par les Comités nationaux de l'IEC. Tous les efforts raisonnables sont entrepris afin que l'IEC s'assure de l'exactitude du contenu technique de ses publications; l'IEC ne peut pas être tenue responsable de l'éventuelle mauvaise utilisation ou interprétation qui en est faite par un quelconque utilisateur final.
4) Dans le but d'encourager l'uniformité internationale, les Comités nationaux de l'IEC s'engagent, dans toute la mesure possible, à appliquer de façon transparente les Publications de l'IEC dans leurs publications nationales et régionales. Toutes divergences entre toutes Publications de l'IEC et toutes publications nationales ou régionales correspondantes doivent être indiquées en termes clairs dans ces dernières.
5) L'IEC elle-même ne fournit aucune attestation de conformité. Des organismes de certification indépendants fournissent des services d'évaluation de conformité et, dans certains secteurs, accèdent aux marques de conformité de l'IEC. L'IEC n'est responsable d'aucun des services effectués par les organismes de certification indépendants.
6) Tous les utilisateurs doivent s'assurer qu'ils sont en possession de la dernière édition de cette publication.
7) Aucune responsabilité ne doit être imputée à l'IEC, à ses administrateurs, employés, auxiliaires ou mandataires, y compris ses experts particuliers et les membres de ses comités d'études et des Comités nationaux de l'IEC, pour tout préjudice causé en cas de dommages corporels et matériels, ou de tout autre dommage de quelque nature que ce soit, directe ou indirecte, ou pour supporter les coûts (y compris les frais de justice) et les dépenses découlant de la publication ou de l'utilisation de cette Publication de l'IEC ou de toute autre Publication de l'IEC, ou au crédit qui lui est accordé.
8) L'attention est attirée sur les références normatives citées dans cette publication. L'utilisation de publications référencées est obligatoire pour une application correcte de la présente publication.
9) L'attention est attirée sur le fait que certains des éléments de la présente Publication de l'IEC peuvent faire l'objet de droits de brevet. L'IEC ne saurait être tenue pour responsable de ne pas avoir identifié de tels droits de brevets et de ne pas avoir signalé leur existence.

La Norme internationale IEC 62656-8 a été établie par le sous-comité 3D: Classes, propriétés et identification des produits – Dictionnaire de données commun (CDD), du comité d'études 3 de l'IEC: Structures d'informations, documentation et symboles graphiques.

Le texte de cette Norme internationale est issu des documents suivants:

FDIS	Rapport de vote
3D/342/FDIS	3D/346/RVD

Le rapport de vote indiqué dans le tableau ci-dessus donne toute information sur le vote ayant abouti à l'approbation de cette Norme internationale.

Ce document a été rédigé selon les Directives ISO/IEC, Partie 2.

Une liste de toutes les parties de la série IEC 62656, publiées sous le titre général *Enregistrement d'ontologie de produits normalisés et transfert par paquets de données*, peut être consultée sur le site web de l'IEC.

Les futures normes de cette série porteront le nouveau titre général susmentionné. Les titres des normes existantes de cette série seront actualisés à la prochaine édition.

Le comité a décidé que le contenu de ce document ne sera pas modifié avant la date de stabilité indiquée sur le site web de l'IEC sous "http://webstore.iec.ch" dans les données relatives au document recherché. À cette date, le document sera

- reconduit,
- supprimé,
- remplacé par une édition révisée, ou
- amendé.

IMPORTANT – Le logo "colour inside" qui se trouve sur la page de couverture de cette publication indique qu'elle contient des couleurs qui sont considérées comme utiles à une bonne compréhension de son contenu. Les utilisateurs devraient, par conséquent, imprimer cette publication en utilisant une imprimante couleur.

INTRODUCTION

En matière de description des produits et services tout au long de leur cycle de vie, une interopérabilité améliorée des données avec des interventions humaines limitées constitue l'objectif final du développement de normes internationales pour les systèmes de production intelligents. Pour atteindre cet objectif, une ontologie industrielle est destinée à jouer un rôle essentiel en permettant aux composants des systèmes de communiquer entre eux (compréhension machine-machine) sur leurs fonctions, leurs capacités, leurs structures et leurs configurations.

Le modèle d'ontologie par paquet défini dans l'IEC 62656-1 (connu sous l'acronyme "POM" – *parcellized ontology model*) est un modèle d'ontologie générique à quatre couches permettant de capturer différents types de modèles d'ontologie en triant les éléments en catégories de collecte homogène d'entités ontologiques comme les classes (concepts), les propriétés, les relations, les énumérations, les termes (constantes), les types de données, etc. 11 types de catégories sont définis au niveau de la deuxième couche en partant du haut appelée couche de métaontologie (MO – *meta-ontology*). Chaque couche est un ensemble de catégories, chaque catégorie étant représentée par une matrice s'apparentant à un tableau relationnel appelée "paquet de données" dont les métadonnées (attributs) sont intégrées dans une sélection d'instances de la couche immédiatement supérieure. La couche supérieure du POM, appelée couche d'ontologie axiomatique (AO – *axiomatic ontology*), est composée de deux paquets de données uniquement qui définissent conjointement le "concept des concepts" par classes et propriétés, qui est une intégration de technologie d'informations (IT - *information technology*) de la notion de logique mathématique de la classe (c'est-à-dire le "concept") elle-même.

D'autres parties de la série de normes IEC 62656 qui sont désignées collectivement par l'appellation "Normes de paquet", portent sur la spécialisation du POM à des fins bien précises.

L'IEC 62656-2 [1][1] est un guide destiné aux experts du domaine permettant d'appliquer le POM afin de capturer un dictionnaire de données à partir des définitions disponibles dans les normes de produits, le tout dans un format conforme au schéma de dictionnaire IEC 61360-2 [2] et ISO 13584-42 [3] (c'est-à-dire un modèle commun de dictionnaire de données ou CDDM - *common data dictionary model*), en utilisant la spécification d'une partie de l'IEC 62656-1 en tant qu'interface de données officielle pour l'IEC 61360-4 DB appelé IEC CDD (dictionnaire de données commun de l'IEC) et en permettant de charger et de télécharger le dictionnaire au départ et à l'arrivée du CDD de l'IEC. Une mise en œuvre référentielle de l'IEC 62656-1 est disponible sous la forme d'un outil gratuit pour répondre aux besoins de normalisation.

L'IEC 62656-3 est une spécification de mise en correspondance entre un modèle de données normalisé du "Smart-Grid" présenté sous l'acronyme de domaine CIM (*common information model* – modèle d'informations commun) et un modèle de données étendu ou plutôt généralisé du CDD de l'IEC, à savoir le POM. Le CIM est composé des séries de normes IEC 61968/IEC 61970/IEC 62325. Par conséquent, le CDD de l'IEC peut être compatible avec le CIM, à condition que le CDD de l'IEC mette suffisamment en œuvre le POM en tant qu'interface de données ou que base de données. Par ailleurs, cette mise en correspondance induit inévitablement une petite extension du CDD de l'IEC mais qui n'est pas négligeable, sans laquelle la compatibilité du CIM avec le CDD de l'IEC est impossible. Toutefois, il n'est pas nécessaire d'ajouter ou de retirer un élément de l'outil actuellement utilisé en tant qu'interface de données pour le CDD de l'IEC et qui intègre totalement l'IEC 62656-1.

[1] Les numéros entre crochets font référence à la Bibliographie.

L'IEC 62656-5 porte sur une interface de description des activités sous la forme d'une ontologie conforme à l'IEC 62656-1 et permet donc de stocker les définitions disponibles dans les normes internationales centrées sur les activités (l'IEC 62224-3, par exemple) en tant qu'ontologie. La présente partie peut également s'appliquer à la description de scénarios d'utilisation non liée à la production (la description des activités de management des dangers naturels ou du guidage ou de la navigation touristique électronique, par exemple) avec une intégration harmonieuse des activités avec les produits et services connexes.

Cela signifie qu'un référentiel d'ontologie commun (COR – *common ontology repository*) reposant sur le POM peut stocker le CDD de l'IEC et les types CIM des dictionnaires de données et des ontologies. De plus, il peut réduire sans problème les différences et combler le fossé en couvrant les ontologies de différentes provenances.

Les futures parties de la série IEC 62656 sont destinées à apporter un nouvel éclairage sur un éventail d'applications pour le COR reposant sur le POM.

Par-dessus tout, le présent document donne une description des services Web de base des référentiels sémantiques reposant sur le POM, un type avancé d'interface Web incluant une question complexe relative au produit et une interrogation transmise à un autre référentiel étant prévu dans le cadre d'une partie de la série qui sera élaborer ultérieurement.

ENREGISTREMENT D'ONTOLOGIE DE PRODUITS NORMALISES ET TRANSFERT PAR PAQUETS DE DONNÉES –

Partie 8: Interface de service Web pour les paquets de données

1 Domaine d'application

La présente partie de l'IEC 62656 spécifie une interface de service Web d'échange de paquet(s) de données conformes à l'IEC 62656-1 entre un serveur de paquets et un client de paquets ou entre des serveurs de paquets. Cette interface est composée de trois services de base: un service d'enregistrement, un service de résolution et un service d'abonnement.

Le présent document inclut:

– un scénario d'utilisation holistique;
– la spécification détaillée des trois services de base;
– les schémas de notation JSON [1] et XML [5] correspondant à un ou des paquets de données.

Le présent document ne s'applique pas:

– à l'identification et l'autorisation des utilisateurs;
– au langage d'interrogation pour un paquet de données;
– au protocole de transport;
– aux données et techniques de sécurité de communication.

2 Références normatives

Les documents suivants sont cités dans le texte de sorte qu'ils constituent, pour tout ou partie de leur contenu, des exigences du présent document. Pour les références datées, seule l'édition citée s'applique. Pour les références non datées, la dernière édition du document de référence s'applique (y compris les éventuels amendements).

IEC 62656-1:2014, *Enregistrement d'ontologie de produits normalisés et transfert par tableurs – Partie 1 : Structure logique pour les paquets de données*

ISO/IEC 21778, *Information technology – The JSON data interchange syntax* (disponible en anglais seulement)

ISO 639-1, *Codes pour la représentation des noms de langue – Partie 1: Code Alpha-2*

ISO 3166-1, *Codes pour la représentation des noms de pays et de leurs subdivisions – Partie 1: Codes de pays*

ISO 8601-1, *Date and time – Representations for information interchange – Part 1: Basic rules* (disponible en anglais seulement)

ISO 8601-2, *Date and time – Representations for information interchange – Part 2: Extensions* (disponible en anglais seulement)

ISO 13584-32, *Industrial automation systems and integration – Parts library – Part 32: Implementation resources: OntoML: Product ontology markup language* (disponible en anglais seulement)

3 Termes, définitions et termes abrégés

3.1 Termes et définitions

Pour les besoins du présent document, les termes et définitions suivants s'appliquent.

L'ISO et l'IEC tiennent à jour des bases de données terminologiques destinées à être utilisées en normalisation, consultables aux adresses suivantes:

- IEC Electropedia: disponible à l'adresse http://www.electropedia.org/
- ISO Online browsing platform: disponible à l'adresse http://www.iso.org/obp

3.1.1
référentiel d'ontologie d'application
AOR
référentiel d'ontologie dans lequel les ontologies propriétaires et leurs instances peuvent être stockées

Note 1 à l'article: Une ontologie propriétaire peut être une modification et/ou une extension d'une ontologie normalisée.

Note 2 à l'article: Les ontologies normalisées peuvent également être stockées dans un AOR.

Note 3 à l'article: Le terme abrégé "AOR" est dérivé du terme anglais développé correspondant "application ontology repository"

3.1.2
référentiel d'ontologie commun
COR
référentiel d'ontologie partagé dans lequel peuvent être stockées les ontologies normalisées de différents modèles de base

Note 1 à l'article: Le terme abrégé "COR" est dérivé du terme anglais développé correspondant "common ontology repository"

3.1.3
paquets conjonctifs
feuilles de paquets qui sont utilisées ensemble pour définir une bibliothèque, un dictionnaire de référence ou un métadictionnaire

[SOURCE: IEC 62656-1:2014, 3.6]

3.1.4
paquet de données
paquet
structure d'information basée sur une paire d'informations, comprenant un ensemble de propriétés et un ensemble de tuples de valeurs de l'ensemble de propriétés, ayant pour but de décrire un dictionnaire de données de domaine, une bibliothèque de données de domaine, ou un concept de modélisation ontologique

Note 1 à l'article: Un paquet de données est typiquement mis en œuvre et échangé comme un ensemble de tableurs, mais le support de mise en œuvre ou d'échange ne se limite pas aux tableurs. Il peut être de n'importe quelle autre forme.

[SOURCE: IEC 62656-1:2014, 3.8]

3.1.5
dictionnaire
dictionnaire de données
ensemble de termes avec des identificateurs respectifs exprimés dans une syntaxe canonique et avec des définitions communément admises conçues pour fournir un cadre lexical ou taxonomique pour une représentation des connaissances sous une forme informatique facile à interpréter, pouvant être partagée par différents systèmes d'information et différentes communautés

[SOURCE: IEC 62656-1:2014, 3.10]

3.1.6
données JSON
données représentées selon la spécification ISO/IEC 21778

3.1.7
nom JSON
série de caractères attribuée à une valeur ou un objet pour y faire référence dans les données JSON

3.1.8
entité ontologique
artefact utilisé pour représenter une catégorie d'être de choses ou relation entre elles

[SOURCE: IEC 62656-1:2014, 3.36]

3.1.9
référentiel d'ontologie
référentiel de données dans lequel sont stockées les ontologies ou les entités ontologiques

3.1.10
client de paquet
système client ou application qui peut lire ou écrire en général des feuilles de paquetage et qui peut avoir une capacité facultative à les envoyer à un système serveur ou à les recevoir de ce dernier

[SOURCE: IEC 62656-1:2014, 3.37]

3.1.11
couche d'ontologie de paquet
couche d'abstraction intégrée comme un ensemble de paquets de données sur le même niveau

Note 1 à l'article: L'IEC 62656-1:2014 classe les couches d'ontologie de paquet en ontologie axiomatique (AO), métaontologie (MO), ontologie de domaine (DO) et bibliothèque de domaine (DL).

3.1.12
serveur de paquet
système serveur ou application qui peut fournir en général des tableurs de paquets sur Internet

[SOURCE: IEC 62656-1:2014, 3.41]

3.1.13
enregistrement de paquet
opération de saisie d'un nouvel enregistrement d'informations par paquets de données sur un serveur de paquet

Note 1 à l'article: Les opérations d'enregistrement sont classées en opérations d'ajout, de modification et de suppression.

3.1.14
résolution de paquet
opération de réception d'informations par paquets de données à l'aide du ou des paramètres spécifiques en tant que condition(s) de recherche

3.1.15
abonnement de paquet
accord de réception de l'état ou du changement d'informations par les paquets de données

3.2 Termes abrégés

AOR Application ontology repository (référentiel d'ontologie d'application)
CDD Common data dictionary (dictionnaire de données commun)
COR Common Ontology Repository (référentiel d'ontologie commun)
HTTP Hypertext Transfer Protocol (protocole de transfert hypertexte)
HTTPS Hypertext Transfer Protocol Secure (protocole sécurisé de transfert hypertexte)
ICID International Concept Identifier (identificateur de concept international)
JSON JavaScript Object Notation (notation d'objet JavaScript)
RAI Registration Authority Identifier (identificateur d'autorité d'enregistrement)
SOAP Simple Object Access Protocol (protocole SOAP)
VI Version Identifier (identificateur de version)
XML eXtensible Markup Language (langage de balisage extensible)
WADL Web Application Description Language (langage de description d'application Web)
WSDL Web Services Description Language (langage de description de services Web)

4 Scénarios d'utilisation

4.1 Scénario d'utilisation holistique

Une ontologie normalisée comme l'IEC 61360-4 est composée d'un ensemble normalisé de classes et de propriétés qu'il convient d'utiliser pour la compréhension commune de la signification des données entre les utilisateurs. Dans la plupart des cas, ce type d'ontologie normalisée contient un ensemble convenu minimal de classes et de propriétés exprimées en anglais. Il est donc souvent nécessaire de le personnaliser dans le cadre d'une utilisation réelle. Par exemple, une traduction peut être importante pour utiliser une ontologie destinée à des utilisateurs dont l'anglais n'est pas la langue maternelle dans une entreprise régionale. Par exemple encore, des extensions de classes et propriétés (des données commerciales ou des informations administratives, par exemple) sont essentielles à la gestion des informations sur le produit de chaque entreprise. L'approche en quatre couches de modélisation définie dans l'IEC 62656-1 permet de représenter ce type d'extension et de modification dans les paquets de données.

Ce type d'ontologie personnalisée est en général conservé et publié dans un serveur (le référentiel d'ontologie d'application, par exemple) tel qu'un serveur national et un serveur d'entreprise, séparément du serveur (le référentiel d'ontologie commun, par exemple) dans lequel l'ontologie normalisée est stockée. En présence d'un mécanisme de téléchargement et de chargement des paquets de données, mais également d'abonnement à une ontologie à l'aide de la technique Web, l'utilisation d'une ontologie normalisée et de son extension est améliorée et facilitée. Les services Web spécifiés dans le présent document contribuent à l'échange du ou des paquets de données d'une ontologie entre un serveur et un client ou entre des serveurs.

Dans de nombreux cas, l'échange de données tant entre les serveurs qu'entre un serveur et un client est assuré par l'intermédiaire du protocole HTTP(S), car il existe de nombreuses techniques matérielles, logicielles et de sécurité pour cela. Par conséquent, l'utilisation du

protocole HTTP(S) est recommandée pour un service Web de paquet, mais d'autres protocoles sont également disponibles pour l'échange de données au moyen d'un service Web de paquet pour satisfaire à une exigence particulière (la vitesse de communication, par exemple).

Les services Web correspondant à l'échange ou à l'abonnement aux paquets de données définis dans le présent document prennent pour hypothèse, entre autres, les scénarios d'utilisation suivants décrits à la Figure 1.

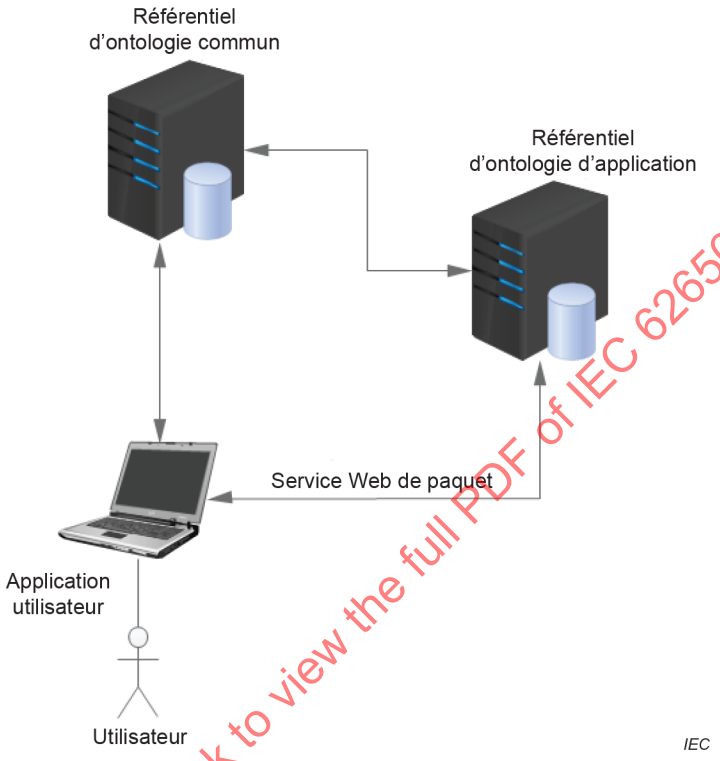


Figure 1 – Scénario d'utilisation holistique des services Web de paquet

4.2 Scénario d'utilisation entre le serveur et le client

Un scénario d'utilisation détaillé entre un référentiel d'ontologie et un client de paquet est décrit à la Figure 2 avec les activités et flux d'informations associés suivants:

– A1: Une application client (un éditeur d'ontologie, par exemple) enregistre le contenu d'un ensemble de paquets de données dans un référentiel d'ontologie sans intervenir sur la page Web du référentiel d'ontologie.

– A2: Une application client (un contrôleur, une application CAD ou un système d'achat, par exemple) demande la résolution de l'identificateur d'un élément de dictionnaire et obtient tout ou partie de l'ensemble des informations relatives à un élément de dictionnaire désigné par l'identificateur, sans intervenir sur la page Web d'un référentiel d'ontologie.

NOTE Le contenu renvoyé par un ensemble de paquets de données est un dictionnaire de données ou une mise à jour de ce dernier.

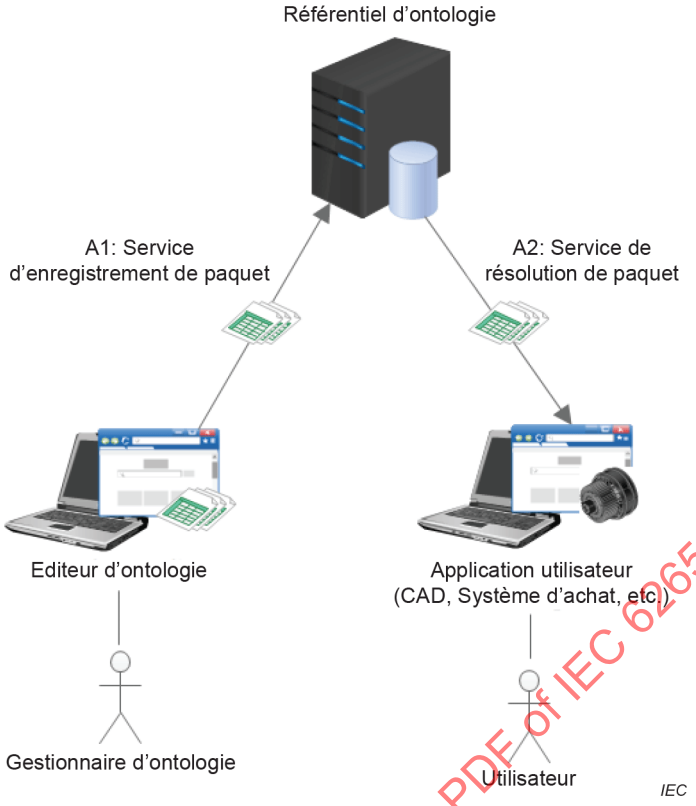


Figure 2 – Services de résolution et d'enregistrement de paquet entre un serveur et un client

4.3 Scénario d'utilisation entre des serveurs

Un scénario d'utilisation détaillé entre des serveurs est décrit à la Figure 3 avec les activités et flux d'informations associés suivants:

– A3: Un référentiel d'ontologie d'application demande à s'abonner à un élément de dictionnaire stocké dans un référentiel d'ontologie commun ou un autre référentiel d'ontologie d'application pour obtenir une mise à jour du contenu stocké dans ce dernier référentiel, lequel contient une nouvelle instance enregistrée.

– A4: Un référentiel d'ontologie d'application qui s'abonne à un élément de dictionnaire stocké dans le référentiel d'ontologie commun obtient une mise à jour du contenu, sur un laps de temps régulier ou en fonction des événements, c'est-à-dire lors de l'enregistrement d'une nouvelle instance dans ce dernier référentiel. Les serveurs de référentiel d'ontologie peuvent procéder à une connexion en cascade afin de délivrer le contenu entre un serveur en amont et un serveur en aval.

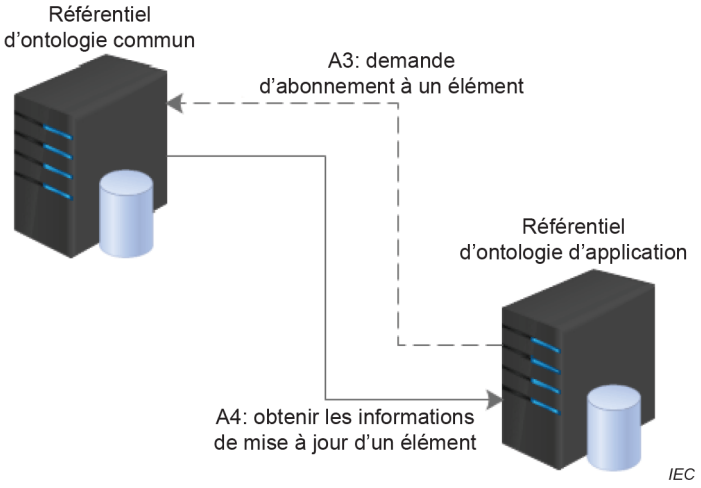


Figure 3 – Service d'abonnement de paquet entre des registres

5 Spécification de service Web de paquet

5.1 Généralités

Le présent document spécifie les trois services Web de base suivants, sur lesquels des applications utilisateur plus sophistiquées ou avancées peuvent reposer:

– service d'enregistrement de paquet (voir 5.4);
– service de résolution de paquet (voir 5.5);
– service d'abonnement de paquet (voir 5.6).

Comme d'autres services Web en général, chaque service Web de base spécifié dans le présent document comporte également des mécanismes de demande, de réponse et d'exception.

Une demande est un message envoyé par un client de paquet à un serveur de paquet. Le message peut contenir un ou plusieurs paramètres, chacun étant utilisé pour exécuter un service sur un serveur de paquet.

Une réponse est un message envoyé par un serveur de paquet à un client de paquet par suite de l'exécution d'un service. Le message peut contenir le résultat de la demande d'informations formulée par un client de paquet.

Une exception est un événement imprévu qui se produit lors de l'exécution d'un service Web de paquet. Si ce type d'événement se produit sur un serveur de paquet et que les informations relatives à l'événement sont utiles ou significatives pour un client de paquet, le serveur de paquet envoie un message exceptionnel au client de paquet. Cela permet à un client de paquet de poursuivre son flux de fonctions en cas de défaillance d'un service. Si une exception se produit lors du traitement d'une fonction dont un client de paquet demande l'exécution, l'exécution se termine et le serveur de paquet envoie l'exception au client de paquet.

En tant que description lisible par une machine des services Web, WADL (Web Application Description Language – langage de description d'application Web) [6] et WSDL (Web Service Description Language – langage de description de services Web) [7] sont largement utilisés. Une description WADL des services spécifiés à l'Article 5 est fournie à l'Annexe B, Article B.1, leur description WSDL étant fournie à l'Article B.2.

5.2 Exception

5.2.1 Généralités

Comme cela est indiqué en 5.1, une exception est un événement imprévu qui se produit lors de l'exécution d'un service Web de paquet.

Le présent document spécifie les types d'exceptions suivants pour un événement imprévu.

– Exception définie par la norme
– Exception définie par l'utilisateur

Une "exception définie par la norme" est le super-type d'exceptions défini dans le présent document. La spécification de ces exceptions est donnée en 5.2.3.

Une "exception définie par l'utilisateur" est le super-type d'exceptions à utiliser pour définir des exceptions qui sont définies dans les applications utilisateur pour un objectif précis. Pour un service Web de paquet entre des serveurs de paquet ou entre un serveur de paquet et un client de paquet, l'utilisation d'une exception définie par l'utilisateur est admise, mais il est possible qu'elle ne soit pas traitée par un récepteur de ce type d'exception. En d'autres termes, ce type d'exceptions peut être reconnu comme étant de simples informations textuelles par un récepteur.

La Figure 4 représente une structure d'exceptions, composée d'exceptions générales et d'exceptions définies dans le Tableau 1.

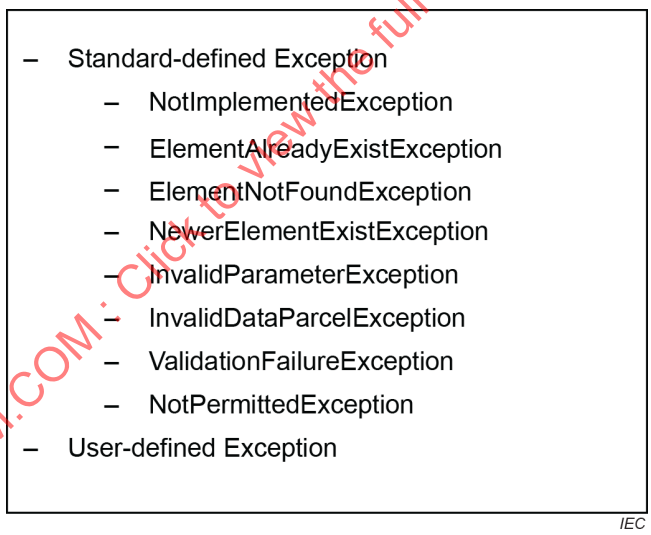


Figure 4 – Structure arborescente des exceptions

5.2.2 Convention de dénomination d'une exception

Pour la mise en œuvre robuste d'une exception définie par l'utilisateur tant sur un serveur de paquet que sur un client de paquet, il est nécessaire de spécifier les caractères qui peuvent être utilisés pour donner un nom à l'exception. En tant que conventions de dénomination, les règles suivantes doivent s'appliquer pour nommer des exceptions définies par l'utilisateur.

– Chaque valeur doit contenir uniquement des caractères alphabétiques ou numériques;
– Chaque valeur doit commencer par une majuscule;
– Chaque valeur doit se terminer par "Exception".

5.2.3 Exceptions définies par la norme

Les exceptions définies par la norme qui peuvent être envoyées ou retournées par un serveur de paquet ou un client de paquet par l'intermédiaire d'un service Web de paquet sont décrites dans le Tableau 1.

Le nom et la définition de chaque exception sont donnés pour une bonne compréhension par l'homme. Le symbole alphabétique préférentiel de chaque exception, qui suit la convention de dénomination spécifiée en 5.2.2, est donné pour la mise en œuvre qui peut être utilisée comme étiquette de l'exception dans un langage de programmation.

Tableau 1 – Exceptions définies par la norme pour les services Web de paquet

Nom préférentiel	Définition	Symbole alphabétique préférentiel
pas de mise en œuvre.	exception définie par la norme indiquant que la fonction spécifiée n'est pas mise en œuvre sur le serveur de référentiel d'ontologie	NotImplementedException
élément déjà existant	exception définie par la norme indiquant que la fonction spécifiée n'est pas exécutée, car le même élément existe déjà dans le référentiel d'ontologie	ElementAlreadyExistException
élément introuvable	exception définie par la norme indiquant que la demande spécifiée ne correspond à aucun élément dans le référentiel d'ontologie	ElementNotFoundException
élément nouveau existant	exception définie par la norme indiquant qu'une nouvelle version d'un élément a été détectée dans le référentiel d'ontologie	NewerElementExistException
paramètre non valide	exception définie par la norme indiquant qu'un paramètre imprévu est spécifié dans le message de demande	InvalidParameterException
valeur imprévue	exception définie par la norme indiquant qu'une valeur imprévue a été est spécifiée dans le message de demande	InvalidDataParcelException
échec de validation	exception définie par la norme indiquant qu'une ou que plusieurs erreurs se sont produites lors d'un processus de validation sur les paquets de données spécifiés	ValidationFailureException
pas de permission	exception définie par la norme indiquant que l'utilisateur n'est pas autorisé à exécuter la fonction spécifiée	NotPermittedException

5.3 Domaine de recherche

Dans le cadre du service de résolution de paquet spécifié en 5.5, si une ligne correspond à un identificateur ou un nom spécifié pour le paramètre "keyword" d'un message de demande, la ligne est renvoyée. Par exemple, si l'identificateur d'une classe est spécifié dans une demande, la ligne de définition de cette classe est renvoyée.

Comme pour le service de résolution de paquet, dans le cadre du service d'abonnement de paquet spécifié en 5.6, si un élément correspond à un identificateur spécifié pour le paramètre "identifiers" d'un message de demande, l'élément est souscrit ou non souscrit. Par exemple, si l'identificateur d'une classe est spécifié dans une demande, la classe est souscrite ou non souscrite. Si une sous-classe est ajoutée à la classe, c'est-à-dire si la structure de la branche est modifiée, les informations de la classe elle-même ne sont pas modifiées, car les informations de la relation "est-une" sont en général décrites du côté d'une sous-classe à l'aide de l'attribut MDC_P010 (superclasse). Il est possible que les abonnés de la classe ne soient donc pas informés de la modification.

Pour un abonnement/désabonnement simultané ou pour obtenir les informations d'une classe et celles de ses superclasses ou sous-classes dans le cadre d'une demande, le présent document spécifie des modificateurs de domaine de recherche pour le paramètre de demande "keyword" du service de résolution de paquet. Le Tableau 2 donne la spécification de ces modificateurs.

Tableau 2 – Spécification des modificateurs du domaine de recherche

Modificateur de domaine de recherche	Description	Exemple
*	Ce modificateur étend le domaine à toutes les sous-classes d'une classe donnée, y compris les classes qui importent leurs propriétés de la classe donnée ou de ses sous-classes.	AAA004* 0112/2///61360_4#AAA004##001*
$	Ce modificateur étend le domaine à toutes les sous-classes directes. Le cas de la relation n'est pas pris en compte.	AAA004$ 0112/2///61360_4#AAA004##001$
%	Ce modificateur étend le domaine à toutes les superclasses, y compris les classes à partir desquelles ces sous-classes importent des propriétés.	AAA004% 0112/2///61360_4#AAA004##001%
!	Ce modificateur étend le domaine à toutes les superclasses directes uniquement.	AAA004! 0112/2///61360_4#AAA004##001!

La Figure 5 représente le fonctionnement de chaque modificateur de domaine de recherche pour étendre un domaine de recherche à l'aide d'un arbre de classification. Si AAA004 du paramètre "keyword" est spécifié dans un message de demande d'un service de résolution de paquet, la ligne qui correspond à AAA004 est renvoyée. Si AAA004% ou AAA004! du paramètre "keyword" est spécifié, les lignes qui correspondent à AAA004 ou aux identificateurs de sa superclasse (AAA001, AAA002 et AAA003) sont renvoyées. Si AAA004* ou AAA004$ du paramètre "keyword" est spécifié, les lignes qui correspondent à AAA004 ou aux identificateurs de ses sous-classes (AAA005, AAA006, AAA007, AAA008, AAA009 et AAA010) sont renvoyées.

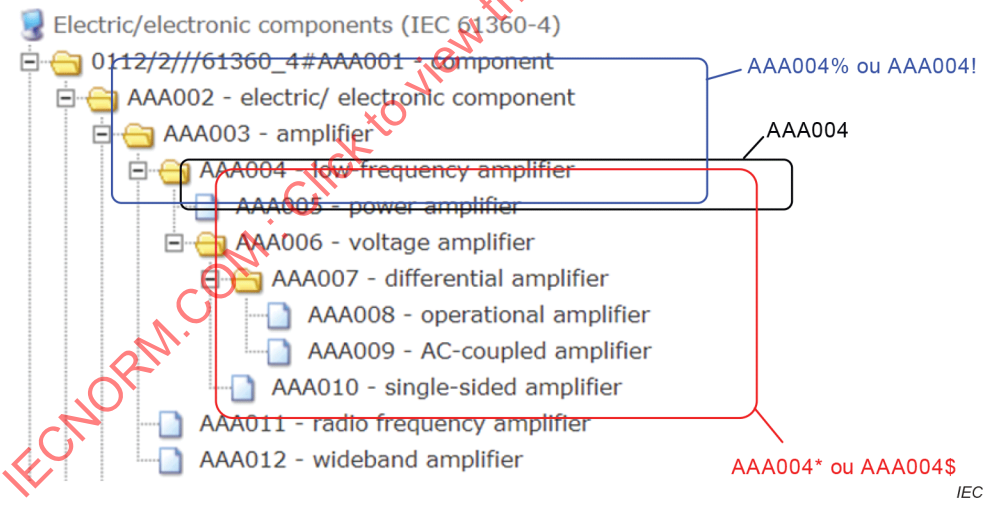


Figure 5 – Exemple de vue structurelle de l'utilisation des modificateurs de domaine de recherche

La Figure 6 donne un exemple de feuille de classe de l'arborescence de classes décrite à la Figure 5. Chaque case en lignes pleines accompagnée d'une étiquette d'un identificateur avec ou sans modificateur de domaine de recherche indique que les lignes sont renvoyées en spécifiant l'identificateur avec ou sans le modificateur de domaine de recherche au niveau du paramètre "keyword" dans un message de demande d'un service de résolution de paquet.

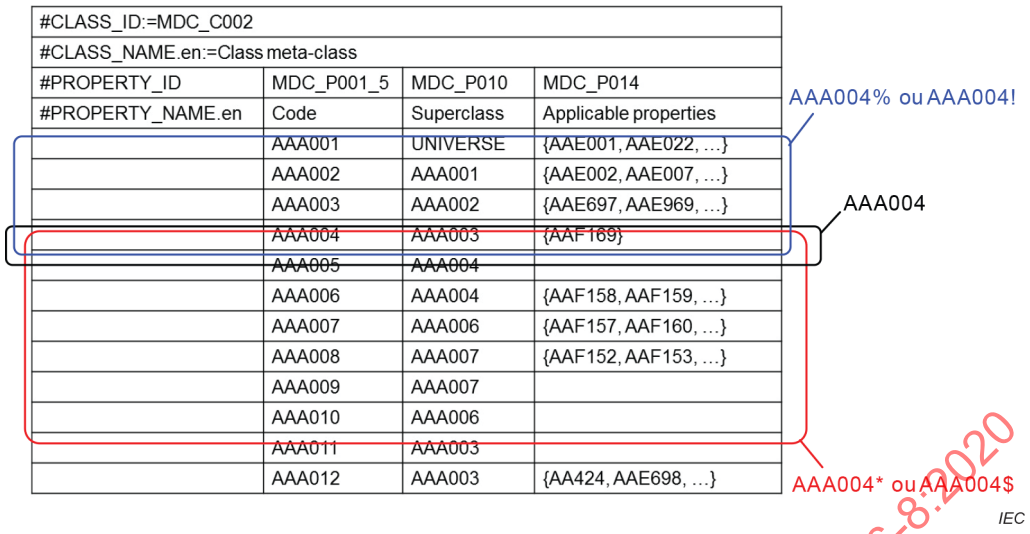


Figure 6 – Exemple de vue de feuille de paquet de l'utilisation des modificateurs de domaine de recherche

5.4 Service d'enregistrement de paquet

5.4.1 Généralités

Le service d'enregistrement de paquet permet à un client de paquet d'enregistrer des éléments dans des paquets de données sur un serveur de paquet. Lorsqu'un serveur de paquet reçoit une demande de la part d'un client de paquet autorisé à enregistrer, le serveur de paquet vérifie la validité de tous les éléments des paquets de données de la demande, puis met à jour le contenu dans sa base de données. En cas d'erreur lors de l'exécution d'une demande provenant d'un client (si le serveur de paquet détecte un ou plusieurs éléments non valides, par exemple), il convient que le serveur de paquet mette fin à l'exécution de la demande, puis envoie une exception au client, contenant des informations détaillées relatives aux erreurs. Dans ce cas, aucun élément des paquets de données de la demande n'est ajouté, mis à jour ou supprimé sur le serveur de paquet.

Le service d'enregistrement de paquet est utilisé pour enregistrer une ontologie complète à partir de rien sur un serveur de paquet. Il permet également de modifier le contenu existant d'une ontologie enregistrée sur un serveur de paquet. La modification du contenu consiste à ajouter un élément, modifier l'élément existant ou le supprimer.

5.4.2 Message de demande

5.4.2.1 Généralités

Le message de demande du service d'enregistrement de paquet doit contenir le paramètre "conjunctiveParcels". La valeur de chaîne de ce paramètre est composée d'un ensemble conjonctif de paquets de données en langage JSON ou XML, dont chaque instance doit être enregistrée sur un serveur de paquet. Le Tableau 3 récapitule ce paramètre.

Tableau 3 – Structure d'un message de demande du service d'enregistrement de paquet

Nom	Type de données	Obligation	Exemple
conjunctiveParcels	Chaîne	Obligatoire	<?xml version="1.0" encoding="utf-8"?> <conjunctiveParcels ontoLayer="DO"> <parcel> ... </parcel> </conjunctiveParcels>

5.4.2.2 Exigences relatives aux paquets de données d'entrée

5.4.2.2.1 Généralités

L'instruction #PARCEL_MODE spécifiée dans l'IEC 62656-1 doit être indiquée dans tous les paquets de données envoyés par un client de paquet à un serveur de paquet par l'intermédiaire d'un service d'enregistrement de paquet.

Concernant la spécification de #PARCEL_MODE, FULL et UPDATE impliquent que l'opération contient un ensemble complet de paquets de données à valider en tant qu'ontologie, alors que PARTIAL implique, par défaut, que le résultat de l'opération est un ensemble incomplet de paquets de données. Par conséquent, en cas de validation, le résultat est souvent une erreur.

L'usage des instructions dans les services Web de paquet est le suivant:

– FULL, UPDATE ou PARTIAL peut être utilisé comme valeur pour cette instruction.
– Dans un ensemble conjonctif de paquets de données, la valeur de l'instruction #PARCEL_MODE de chaque paquet doit être cohérente.
– Si FULL ou UPDATE est spécifié, il convient de procéder à une validation sur le serveur.

Si UPDATE ou PARTIAL est spécifié pour #PARCEL_MODE, l'une des trois opérations suivantes doit être utilisée pour chaque instance dans un paquet de données:

– #ADD (voir 5.4.2.2.2);
– #MOD (voir 5.4.2.2.3);
– #DEL (voir 5.4.2.2.4).

Si FULL est spécifié pour #PARCEL_MODE, il n'est pas exigé d'utiliser les trois opérations ci-dessus. Si ces opérations apparaissent dans un paquet de données du mode FULL, elles doivent être ignorées.

5.4.2.2.2 Opération d'ajout

Mot-clé: #ADD
Nom: Opération d'ajout
Définition: opération des modes UPDATE et PARTIAL indiquant qu'une instance est destinée à être ajoutée au contenu stocké dans un serveur de paquet
Description: Dans le mode UPDATE, un serveur de paquet doit vérifier la duplication de l'instance avec la ou les valeurs de la clé (composite) ou de l'identificateur d'objet de données avec MDC_P066 avant de mettre à jour le contenu. Si la duplication est trouvée, le serveur de paquet rejette la demande et renvoie l'exception définie par la norme "ValidationFailureException" au client de paquet. Si la configuration de la clé (composite) présente une incohérence entre les paquets de données chargés par l'intermédiaire du service et le contenu stocké dans le serveur de paquet, ce dernier doit renvoyer au client de paquet l'exception définie par la norme "ValidationFailureException", qui contient une liste d'identificateurs des éléments dont la validation n'a pas abouti.

5.4.2.2.3 Opération de modification

Mot-clé: #MOD
Nom: Opération de modification
Définition: opération des modes UPDATE et PARTIAL indiquant que l'instance du contenu stocké dans un serveur de paquet, qui correspond à la ou aux valeurs de la clé (composite) ou de l'identificateur d'objet de données identifié par MDC_P066, est destinée à être modifiée
Description: Dans le mode UPDATE, un serveur de paquet doit vérifier l'existence de l'instance avant de procéder à sa mise à jour. S'il n'est pas trouvé, le serveur de paquet rejette la demande et renvoie l'exception définie par la norme "ValidationFailureException" au client de paquet. Si la configuration de la clé (composite) présente une incohérence entre les paquets de données chargés par l'intermédiaire du service et le contenu stocké dans le serveur de paquet, ce dernier doit renvoyer au client de paquet l'exception définie par la norme "ValidationFailureException", qui contient une liste d'identificateurs des éléments dont la validation n'a pas abouti.

5.4.2.2.4 Opération de suppression

Mot-clé: #DEL
Nom: Opération de suppression
Définition: opération des modes UPDATE et PARTIAL indiquant que l'instance du contenu stocké dans un serveur de paquet, qui correspond à la ou aux valeurs de la clé (composite) ou de l'identificateur d'objet de données identifié par MDC_P066, est destinée à être supprimée
Description: Dans le mode UPDATE, un serveur de paquet doit vérifier l'existence de l'instance avant de procéder à sa suppression. S'il n'est pas trouvé, le serveur de paquet rejette la demande et renvoie l'exception définie par la norme "ValidationFailureException" au client de paquet. Il n'est pas nécessaire qu'une instance avec cette opération dispose soit de la ou des valeurs de clé (composite) soit de l'identificateur d'objet de données identifié par MDC_P066. Si une instance à supprimer ne dispose ni de la ou des valeurs de clé (composite) ni de l'identificateur d'objet de données, elle est ignorée par le processus d'exécution du service de résolution de paquet. Si la configuration de la clé (composite) présente une incohérence entre les paquets de données chargés par l'intermédiaire du service et le contenu stocké dans le serveur de paquet, ce dernier doit renvoyer au client de paquet l'exception définie par la norme "ValidationFailureException", qui contient une liste d'identificateurs des éléments dont la validation n'a pas abouti.

5.4.3 Message de réponse

Le message de réponse du service d'enregistrement de paquet doit contenir le paramètre "operationResult". Ce paramètre comporte une valeur booléenne indiquant que l'exécution d'une demande d'enregistrement a abouti ou échoué. Si l'exécution a abouti, le référentiel d'ontologie renvoie une indication "true" au client de paquet. Si l'exécution n'a pas abouti, l'une des exceptions est renvoyée au client de paquet, outre le renvoi d'une indication "false". Le Tableau 4 récapitule la structure d'un message de réponse.

Tableau 4 – Structure d'un message de réponse du service d'enregistrement de paquet

Nom	Type de données	Obligation	Exemple
operationResult	Booléen	Obligatoire	true

5.4.4 Exception

Les exceptions définies par la norme suivantes spécifiées en 5.2 s'appliquent au service d'enregistrement de paquet.

– pas de mise en œuvre.
– l'élément existe déjà pour l'opération #ADD
– aucun élément trouvé pour les opérations #MOD et #DEL
– nouvel élément de version existant
– paramètre imprévu
– valeur imprévue
– échec de validation
– pas de permission

5.5 Service de résolution de paquet

5.5.1 Généralités

Le service de résolution de paquet permet à un client de paquet de résoudre les entités ontologiques stockées sur un serveur de paquet. Une résolution a lieu sur un serveur de paquet à l'aide des identificateurs ou des noms spécifiés dans un message de demande provenant d'un client de paquet. Si des entités ontologiques correspondent à ces identificateurs ou noms sur un serveur de paquet, ce dernier renvoie ces entités ontologiques au format XML ou JSON conformément au présent document. En cas d'erreur lors de l'exécution d'une demande provenant d'un client, il convient que le serveur de paquet mette fin à l'exécution de la demande, puis envoie une exception au client, contenant des informations détaillées relatives aux erreurs. Dans ce cas, aucun contenu n'est renvoyé au client de paquet.

Le service de résolution de paquet offre deux types de résolutions pour spécifier l'identificateur d'une entité ontologique, comme suit:

– résolution de définition, qui vise à résoudre les définitions des entités ontologiques correspondant aux identificateurs ou noms spécifiés dans un message de demande.
– résolution d'instance, qui vise à résoudre les instances des entités ontologiques correspondant aux identificateurs ou noms spécifiés dans un message de demande.

Le type de résolutions choisi est déterminé par une valeur de paramètre du message de demande.

EXEMPLE 1 Dans une résolution de définition, si l'identificateur d'une classe est spécifié dans un message de demande, toutes les informations de la classe (le code, le nom et la définition, par exemple) sont renvoyées. Ces informations sont composées de paires d'attributs de classe (définis dans l'IEC 62656-1) et de sa valeur.

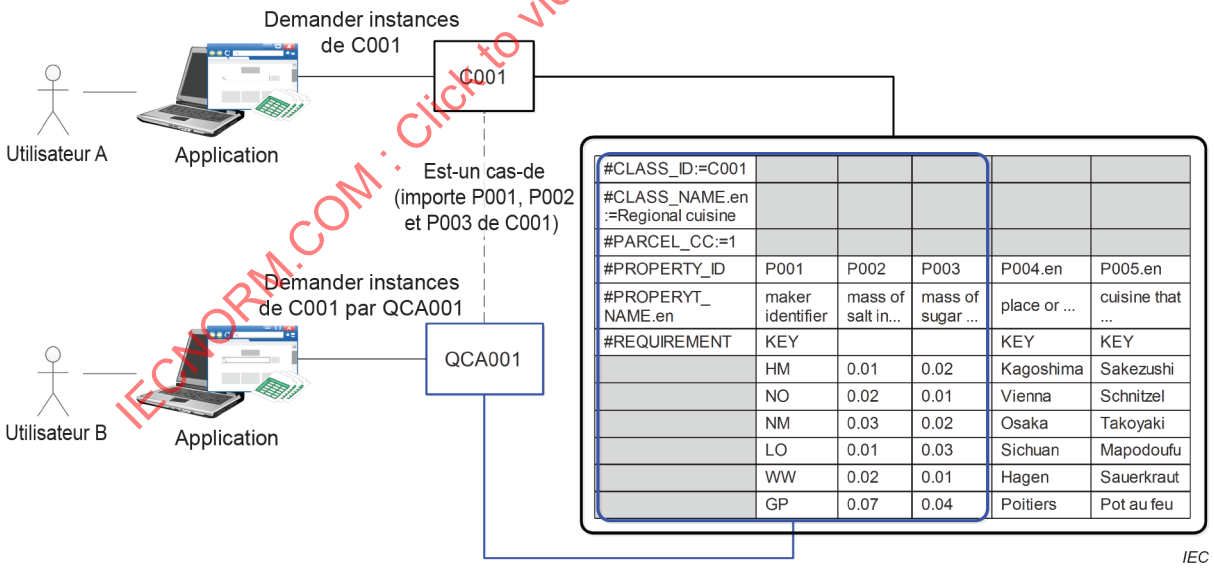
EXEMPLE 2 Dans une résolution d'instance, si l'identificateur d'une classe est spécifié dans un message de demande, toutes les instances de la classe sont renvoyées.

Si plusieurs identificateurs sont spécifiés en même temps dans un message de demande, le résultat peut comporter un ensemble conjonctif de paquets de données. Par exemple, si les identificateurs de la métaclasse class et de la métaclasse property sont spécifiés dans un message de demande, le message de réponse contient un ensemble conjonctif de paquets de données comportant des instances de ces métaclasses.

NOTE Un mécanisme complexe d'interrogation (extraction des classes au moyen d'une hiérarchie de classes, par exemple) est spécifié dans une autre norme ou une autre partie de l'IEC 62656.

Si un identificateur d'une classe d'éléments est spécifié dans un message de demande, une instance contenue dans un message de réponse à la demande contient les valeurs de toutes les propriétés de la classe d'éléments. Si un serveur de paquet prépare une classe en tant qu'aperçu d'une classe d'éléments, l'identificateur de ladite classe est disponible au lieu de spécifier l'identificateur de la classe d'éléments. En d'autres termes, ce type de classes en tant qu'aperçu est utilisé pour spécifier un ensemble prédéfini d'enquêtes utiles ou significatives relatives aux propriétés d'un produit. Ce type de classes en tant qu'aperçu peut être modélisé en s'appuyant sur le mécanisme d'extension de dictionnaire spécifié dans l'ISO 13584-24:2003. Il peut être considéré comme un ensemble d'enquêtes classiques relatives à une classe de produits s'apparentant à des FAQ (Questions fréquemment posées).

La Figure 7 donne un exemple de service de résolution d'instance. Le tableau de la Figure 7 contient les instances d'une classe d'éléments C001 d'un tableur IEC 62656-1. Une classe QCA001 est une classe en tant qu'aperçu de la classe d'éléments C001, qui importe les propriétés P001, P002 et P003 depuis la classe d'éléments C001. Lorsqu'un utilisateur A envoie une demande en spécifiant C001, il obtient toutes les instances de la classe. Pendant ce temps, l'utilisateur B envoie une demande en spécifiant QCA001 au lieu de C001 pour obtenir les instances de la classe d'éléments C001, chacune d'elles comportant uniquement les valeurs de P001, P002 et P003.



#CLASS_ID:=C001					
#CLASS_NAME.en :=Regional cuisine					
#PARCEL_CC:=1					
#PROPERTY_ID	P001	P002	P003	P004.en	P005.en
#PROPERYT_ NAME.en	maker identifier	mass of salt in...	mass of sugar ...	place or ...	cuisine that ...
#REQUIREMENT	KEY			KEY	KEY
	HM	0.01	0.02	Kagoshima	Sakezushi
	NO	0.02	0.01	Vienna	Schnitzel
	NM	0.03	0.02	Osaka	Takoyaki
	LO	0.01	0.03	Sichuan	Mapodoufu
	WW	0.02	0.01	Hagen	Sauerkraut
	GP	0.07	0.04	Poitiers	Pot au feu

Figure 7 – Aperçu du service de résolution de paquet

5.5.2 Message de demande

5.5.2.1 Généralités

Le message de demande du service de résolution de paquet doit contenir les paramètres "requestKind", "keywordKind", "keyword", "language", "pwsCodificationMode", "startPoint" et "endPoint". Le Tableau 5 récapitule ces paramètres.

Tableau 5 – Structure d'un message de demande du service de résolution de paquet

Nom	Type de données	Obligation	Exemple
requestKind	Chaîne	Obligatoire	DEFINITION
keywordKind	Chaîne	Obligatoire	ID
keyword	Chaîne	Obligatoire	0112/2///61360_4#AAA013 0112/2///61360_4#AAA013* tension assignée
dictionaryId	Chaîne	Facultatif	0112/2///61360_4
language	Chaîne	Facultatif	en en, fr
pwsCodificationMode	Chaîne	Obligatoire	VERTICAL
notationFormat	Chaîne	Facultatif	XML JSON
startPoint	Entier	Facultatif	10
endPoint	Entier	Facultatif	101

5.5.2.2 requestKind

Le paramètre "requestKind" est associé à une valeur de chaîne, "DEFINITION" ou "INSTANCE".

Si "DEFINITION" est spécifié, il est considéré que les informations de l'entité ontologique elle-même, qui correspond à l'identificateur ou au nom spécifié dans un message de demande, sont résolues. Il s'agit d'extraire toutes les informations d'une classe en spécifiant son identificateur, y compris le code, le nom préférentiel et les définitions.

EXEMPLE 1 Dans le mode "DEFINITION", si l'identificateur d'une classe est spécifié, les informations de la classe (y compris le code, le nom et la définition) sont extraites.

Si "INSTANCE" est spécifié, il est considéré que les instances des entités ontologiques, qui correspondent aux identificateurs ou aux noms spécifiés dans un message de demande, sont résolues. Il s'agit, par exemple, d'obtenir les codes de toutes les classes d'une ontologie en spécifiant l'identificateur de la métaclasse class.

EXEMPLE 2 Dans le mode "INSTANCE", si l'identificateur d'une métaclasse class est spécifié, les informations de chaque classe d'une ontologie (y compris le code, le nom et la définition) sont extraites.

5.5.2.3 keywordKind

Le paramètre "keywordKind" est associé à une valeur de chaîne, "ID" ou "NAME". Si "ID" est spécifié, une valeur du paramètre "keyword" doit contenir une liste d'identificateurs des entités ontologiques. Si "NAME" est spécifié, une valeur du paramètre "keyword" doit contenir une liste de noms des entités ontologiques.

5.5.2.4 keyword

Le paramètre "keyword" est associé à une valeur de chaîne composée de l'un des éléments suivants:

– une liste d'identificateurs d'entités ontologiques avec/sans modificateur de domaine de recherche spécifié en 5.3;

– une liste de noms d'entités ontologiques.

Si un serveur de paquet reçoit des mots-clés par l'intermédiaire d'un service Web de paquet, il renvoie à son client un résultat correspondant à l'un des mots-clés.

Les règles de description d'une valeur du paramètre "keyword" sont récapitulées dans la liste suivante.

– Si plusieurs éléments sont spécifiés, chacun d'eux doit être séparé par une virgule ",".
– Si un nom contient des guillemets, ces derniers doivent être codés par d'autres guillemets.
– Si un nom contient une virgule ou des guillemets, il doit être placé entre un guillemet ouvrant et un guillemet fermant.

EXEMPLE 1 Si "0112/2///62656_1#MDC_C002##001,0112/2///62656_1#MDC_C003##001" en tant que valeur du paramètre "keyword" et "INSTANCE" en tant que valeur du paramètre "requestKind" sont spécifiés dans un message de demande du service de résolution de paquet, l'ensemble des classes et des propriétés enregistrées dans le serveur de paquet est par hypothèse renvoyé au client de paquet. Ici, "0112/2///62656_1#MDC_C002##001" et "0112/2///62656_1#MDC_C003##001" sont des ICID de la classe meta-class et de la propriété meta-class, respectivement, qui sont définies dans l'IEC 62656-1:2014.

EXAMPLE 2 Si "0112///61360_4#AAA001##001*" en tant que valeur du paramètre "keyword" et "DEFINITION" en tant que valeur du paramètre "requestKind" sont spécifiés dans un message de demande du service de résolution de paquet, l'ensemble des éléments de dictionnaire dans la classe spécifiée est par hypothèse renvoyé au client de paquet.

5.5.2.5 dictionaryId

Le paramètre facultatif "dictionaryId" comporte une valeur de chaîne qui indique l'identificateur d'un dictionnaire de données permettant de limiter un domaine de recherche à des éléments du dictionnaire de données. En d'autres termes, les éléments d'autres dictionnaires de données enregistrés dans un référentiel sont ignorés dans la demande.

5.5.2.6 language

Le paramètre "language" est associé à une valeur de chaîne composée d'une liste de langues, chaque langue étant l'une des suivantes en fonction de l'ICID spécifié dans l'IEC 62656-1:

– code de langue à 2 lettres conformément à l'ISO 639-1 ("fr", par exemple);
– combinaison d'un code de langue à 2 lettres conformément à l'ISO 639-1 et d'un code de pays à 2 lettres conformément à l'ISO 3166-1, séparés par un tiret ("fr-FR", par exemple).

Si au moins une langue est spécifiée, un serveur de paquet renvoie à son client le contenu dans les langues spécifiées.

NOTE 1 Si plusieurs langues sont spécifiées, chacune d'elles est séparée par une virgule (",").

NOTE 2 Si aucune langue n'est spécifiée dans une demande, le contenu d'une langue normalisée est renvoyé.

5.5.2.7 pwsCodificationMode

Le paramètre "pwsCodificationMode" est associé à une valeur de chaîne, "VERTICAL" ou "LATERAL". Si "VERTICAL" est spécifié, un ensemble conjonctif de paquets de données renvoyés dans le cadre du service de résolution de paquet doit être structuré dans la codification verticale spécifiée à l'Article 5. Si "LATERAL" est spécifié, un ensemble conjonctif de paquets de données renvoyés dans le cadre du service de résolution de paquet doit être structuré dans la codification latérale spécifiée à l'Article 5.

5.5.2.8 notationFormat

Le paramètre "notationFormat" comporte une chaîne de valeur "XML" ou "JSON". Si "JSON" est spécifié, un ensemble conjonctif de paquets de données renvoyés dans le cadre du service de résolution de paquet doit être présenté au format JSON spécifié à l'Article 7. Si "XML" est spécifié, un ensemble conjonctif de paquets de données renvoyés dans le cadre du service de résolution de paquet doit être présenté au format XML spécifié à l'Article 8.

Si un serveur de paquets prend en charge la transaction des demandes, un format de notation peut être négocié entre le serveur de paquet et un client de paquet avant le début de la transaction. Dans ce cas, ce paramètre peut être ignoré dans chaque message de demande.

5.5.2.9 startPoint

Le paramètre "startPoint" indique un point de départ ou un point de continuation d'un (ensemble conjonctif de) paquet(s) de données par suite de l'exécution d'une demande de résolution. "startPoint" est associé à une valeur entière positive. Sa valeur par défaut est 1. Si ce paramètre n'est pas spécifié, la valeur par défaut est appliquée.

Pour chaque instance contenue dans un (ensemble conjonctif de) paquet(s) de données par suite de l'exécution d'une demande de résolution, le numéro doit être virtuellement attribué.

5.5.2.10 endPoint

Le paramètre "endPoint" indique un point final d'un service de résolution de paquet. "endPoint" est associé à une valeur entière positive qui doit être supérieure ou égale à la valeur du paramètre "startPoint". Si ce paramètre n'est pas spécifié, tous les éléments qui se trouvent après le point de départ sont renvoyés par suite de la demande.

Pour chaque instance contenue dans un (ensemble conjonctif de) paquet(s) de données par suite de l'exécution d'une demande de résolution, le numéro doit être virtuellement attribué.

5.5.3 Message de réponse

Le message de réponse du service de résolution de paquet doit contenir le paramètre "conjunctiveParcels". Ce paramètre est associé à une valeur de chaîne composée d'un ensemble conjonctif de paquets de données qui correspond aux conditions spécifiées dans un message de demande. Le Tableau 6 récapitule la structure d'un message de réponse.

Tableau 6 – Structure d'un message de réponse du service de résolution de paquet

Nom	Type de données	Obligation	Exemple
conjunctiveParcels	Chaîne	Obligatoire	<?xml version="1.0" encoding="utf-8"?> <conjunctiveParcels ontoLayer="DO"> <parcel> ... </parcel> </conjunctiveParcels>

5.5.4 Exception

Les exceptions suivantes définies par la norme spécifiées en 5.2 s'appliquent au service de résolution de paquet:

– pas de mise en œuvre
– élément introuvable
– paramètre imprévu
– valeur imprévue
– pas de permission.

5.6 Service d'abonnement de paquet

5.6.1 Généralités

Le service d'abonnement de paquet permet à un client de paquet de recevoir ou de ne plus recevoir une notification l'informant de la modification de l'état, des informations ou de la valeur de l'élément spécifié. Ici, la modification d'un élément signifie qu'un élément a été ajouté, modifié ou supprimé. En cas d'erreur lors de l'exécution d'une demande provenant d'un client (si le serveur de paquet détecte un ou plusieurs identificateurs non valides, par exemple), il convient que le serveur de paquet mette fin à l'exécution de la demande, puis envoie une exception au client, contenant des informations détaillées relatives aux erreurs. Dans ce cas, l'état de l'abonnement ne change pas sur le serveur de paquet.

Le service d'abonnement de paquet offre des abonnements par émission. En effet, en cas de modification de l'état ou d'une valeur d'un élément, le serveur de paquet envoie une notification à ses abonnés.

5.6.2 Message de demande

5.6.2.1 Généralités

Le message de demande du service d'abonnement de paquet doit contenir les paramètres "requestKind", "identifiers" et "subscriptionOperationKind". Le Tableau 7 récapitule la structure d'un message de demande.

Tableau 7 – Structure d'un message de demande du service d'abonnement de paquet

Nom	Type de données	Obligation	Exemple
requestKind	Chaîne	Obligatoire	DEFINITION
identifiers	Chaîne	Obligatoire	0112/2///61360_4#AAA013 0112/2///61360_4#AAA013* AAA013
subscriptionOperationKind	Chaîne	Obligatoire	DÉBUT
expirationDate	Chaîne	Facultatif	20200131 2020-01-31

5.6.2.2 requestKind

Le paramètre "requestKind" est associé à une valeur de chaîne, "DEFINITION" ou "INSTANCE".

Si "DEFINITION" est spécifié, il est considéré qu'un abonnement aux informations d'une entité ontologique elle-même, qui correspond à l'identificateur spécifié dans un message de demande, commence ou se termine.

Si "INSTANCE" est spécifié, il est considéré qu'un abonnement aux instances d'entités ontologiques, qui correspond aux identificateurs spécifiés dans un message de demande, commence ou se termine.

5.6.2.3 identifiers

Le paramètre "identifiers" est associé à une valeur de chaîne composée d'une liste d'identificateurs d'entités ontologiques avec/sans les modificateurs de domaine de recherche spécifiés en 5.3, qu'un client de paquet ou un utilisateur demande pour s'y abonner.

5.6.2.4 subscriptionOperationKind

Le paramètre "subscriptionOperationKind" est associé à une valeur de chaîne, "START" ou "END". Si "START" est spécifié, un abonnement à des entités ontologiques des identificateurs spécifiés dans le paramètre "identifiers" commence. Si "END" est spécifié, un abonnement à des entités ontologiques des identificateurs spécifiés dans le paramètre "identifiers" se termine.

5.6.2.5 expirationDate

Le paramètre "expirationDate" comporte une valeur de chaîne permettant de spécifier une date calendaire au format "AAAAMMJJ" ou "AAAA-MM-JJ" conforme à l'ISO 8601-1 et à l'ISO 8601-2. Si une date est spécifiée dans un message de demande, l'abonnement à des entités ontologiques, qui a commencé par cette demande d'abonnement, se termine automatiquement sur le serveur de paquet à la date spécifiée.

5.6.3 Message de réponse

Le message de réponse doit contenir le paramètre "operationResult". Ce paramètre comporte une valeur booléenne indiquant si l'exécution d'une demande d'abonnement a abouti ou échoué. Si l'exécution a abouti, le serveur d'ontologie renvoie une indication "true" au client de paquet. Si l'exécution n'a pas abouti, l'une des exceptions est envoyée ou renvoyée à un client de paquet, outre le renvoi d'une indication "false". Le Tableau 8 récapitule la structure d'un message de réponse.

Tableau 8 – Structure d'un message de réponse du service d'abonnement de paquet

Nom	Type de données	Obligation	Exemple
operationResult	Booléen	Obligatoire	true

5.6.4 Exception

Les exceptions suivantes définies par la norme spécifiées en 5.2 s'appliquent au service d'abonnement de paquet:

– pas de mise en œuvre
– élément introuvable
– paramètre imprévu
– valeur imprévue
– pas de permission.

5.6.5 Spécification de la notification de modification

Une notification envoyée à un client de paquet est composée d'un paramètre obligatoire "identifiers" et de deux paramètres facultatifs "defaultDataSupplier" et "defaultDataVersion". Le Tableau 9 récapitule la spécification d'un message d'une notification.

Le paramètre "identifiers" permet de spécifier une liste des identificateurs d'éléments présentant un intérêt pour le client de paquet. Ces identificateurs sont décrits en utilisant la virgule comme séparateur.

Les paramètres facultatifs peuvent être utilisés pour éviter de répéter le même RAI ou VI, ce qui permet de réduire la taille des données de la notification. Si "defaultDataSupplier" est présent dans une notification, un RAI peut être omis de certains identificateurs dans la notification. Dans ce cas, il convient d'appliquer la valeur de "defaultDataSupplier" à ces identificateurs. De la même manière, si "defaultDataVersion" est présent dans une notification, un VI peut être omis de certains identificateurs dans la notification. Dans ce cas, il convient d'appliquer la valeur de "defaultDataVersion" à ces identificateurs.

Tableau 9 – Spécification d'une notification

Nom	Type de données	Obligation	Exemple
identifiers	Chaîne	Obligatoire	
defaultDataSupplier	Chaîne	Facultatif	
defaultDataVersion	Chaîne	Facultatif	

6 Spécification de la représentation de données de paquet dans un message de service Web

6.1 Généralités

Il convient qu'un format de données échangées par l'intermédiaire d'un service Web satisfasse aux critères suivants:

– la spécification est accessible au public;
– le format est largement utilisé dans les systèmes et dispositifs d'entreprise, y compris les dispositifs et applications mobiles;
– des bibliothèques de logiciels informatiques pour différents langages de programmation sont fournies.

Conformément à ces critères, XML et JSON sont adoptés dans le présent document.

Dans le secteur, XML a été traditionnellement utilisé comme format de données pour l'échange de données entre les systèmes industriels. XML présente certains avantages, comme un schéma bien défini et une technique de sécurité sur le Web. L'ISO 13584-32 spécifie le schéma XML [6] pour représenter les données conformes à l'ISO 13584-42 au format XML. Par ailleurs, XML présente certains inconvénients, comme la taille importante des données, car il convient de placer chaque donnée entre une paire de balises XML.

Une autre notation de données, JSON, est également utilisée, particulièrement dans les applications mobiles. JSON donne une description simple et légère des données en s'appuyant sur la notation JavaScript. Compte tenu de sa facilité d'utilisation et de compréhension, JSON est utilisé non seulement dans JavaScript, mais également dans d'autres langages de programmation populaires comme Java, C#, PHP et python, en tant que format de données échangé entre les systèmes et les dispositifs et applications (mobiles).

6.2 Représentation des données de base

Les paquets de données sont échangés entre un serveur de paquet et un client de paquet, et sont placés dans un message de demande ou de réponse au moyen d'un service Web. Le présent document spécifie la représentation des données des paquets de données dans un message.

La Figure 8 représente la structure de données conceptuelles d'un ensemble conjonctif de paquets de données spécifié dans l'IEC 62656-1. Un ensemble conjonctif de paquets de données appelés "paquets conjonctifs" est composé d'au moins un paquet de données, chacun étant appelé "paquet". Chaque paquet de données est composé de deux sections: une section d'en-tête et une section de données, dans l'ordre. La section d'en-tête est par ailleurs composée de deux sous-sections: une section en-tête de classe et une section en-tête de schéma. La section en-tête de classe décrit les informations relatives au paquet de données, alors que la section en-tête de schéma décrit le schéma de données du paquet. La section de données décrit les instances conformément au schéma décrit dans la section en-tête de schéma.

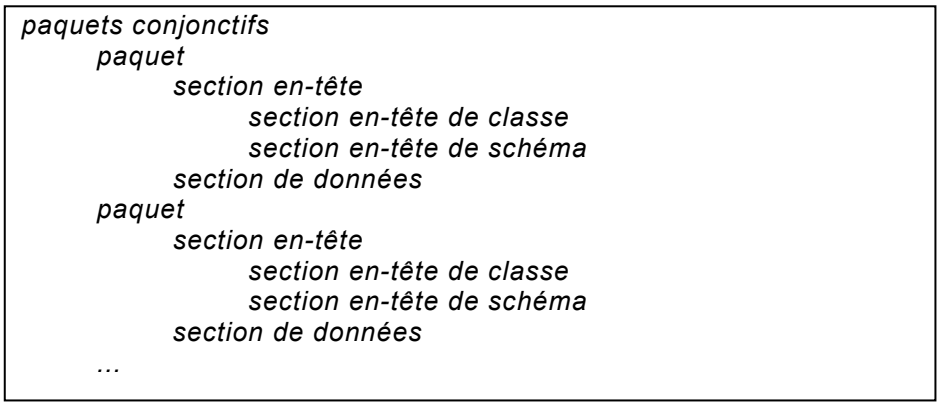


Figure 8 – Structure de base d'une représentation de données pour un ensemble conjonctif de paquets de données

Le paragraphe 6.3 spécifie les mots-clés réservés pour la représentation de données en langage JSON et XML. Pour la représentation de données en langage JSON, chaque mot-clé est utilisé comme un nom JSON. Pour la représentation de données en langage XML, chaque mot-clé est utilisé comme un nom d'élément ou nom d'attribut XML.

6.3 Mots-clés réservés

6.3.1 Mot-clé indiquant les paquets conjonctifs

Mot-clé: conjunctiveParcels
Nom: paquets conjonctifs
Définition: mot-clé réservé pour la collecte d'un ou de plusieurs paquets de données, chacun étant placé dans le mot-clé "parcel"
Occurrence: 1 pour un message de demande ou de réponse

6.3.2 Mot-clé indiquant la couche d'ontologie de paquet d'un ensemble de paquets de données

Mot-clé: ontoLayer
Nom: couche d'ontologie de paquet
Définition: mot-clé réservé pour décrire une couche d'ontologie de paquet d'un ensemble de paquets de données collectées par le mot-clé "conjunctiveParcels"
Description: La valeur doit être "AO", "MO", "DO" ou "DL"
Occurrence: 1 pour une occurrence du mot-clé "conjunctiveParcels"

6.3.3 Mot-clé indiquant la section d'en-tête

Mot-clé: header
Nom: section en-tête
Définition: mot-clé réservé pour la collecte de la section en-tête de classe placée dans le mot-clé "classHeader" et la section en-tête de schéma placée dans le mot-clé "schemaHeader"
Occurrence: 1 pour une occurrence du mot-clé "parcel"

6.3.4 Mot-clé indiquant la section en-tête de classe

Mot-clé: classHeader
Nom: section en-tête de classe
Définition: mot-clé réservé pour la collecte des instructions de la section en-tête de classe
Occurrence: 1 pour une occurrence du mot-clé "header"

6.3.5 Mot-clé indiquant la section en-tête de schéma

Mot-clé: schemaHeader
Nom: section en-tête de schéma
Définition: mot-clé réservé pour la collecte des instructions de la section en-tête de schéma
Occurrence: 1 pour une occurrence du mot-clé "header"

6.3.6 Mot-clé indiquant la section de données

Mot-clé: data
Nom: section de données
Définition: mot-clé réservé pour la collecte d'instances
Occurrence: 1 pour une occurrence du mot-clé "conjunctiveParcels"

6.3.7 Mot-clé indiquant le fournisseur par défaut dans la section de données

Mot-clé: defaultSupplier
Nom: default supplier
Définition: mot-clé réservé pour spécifier le fournisseur par défaut des identificateurs de propriété dans la section de données, chacun faisant office de clé pour ses valeurs
Occurrence: 1 pour une occurrence du mot-clé "data"

6.3.8 Mot-clé indiquant la version par défaut dans la section de données

Mot-clé: defaultVersion
Nom: default version
Définition: mot-clé réservé pour spécifier la version par défaut des identificateurs de propriété dans la section de données, chacun faisant office de clé pour ses valeurs
Occurrence: 1 pour une occurrence du mot-clé "data"

6.4 Instructions supplémentaires aux paquets de données pour les services Web de paquet

6.4.1 Mode de codification

En langage JSON et XML, deux modes de codification sont fournis. L'un est la codification verticale, et l'autre la codification latérale.

La codification verticale est un mode de collecte des valeurs par propriété. La codification verticale permet d'obtenir les valeurs d'une propriété (pour vérifier si une valeur spécifiée existe dans une liste de valeurs d'une propriété, par exemple) et pour agréger et calculer les valeurs d'une propriété.

La codification latérale est un mode de collecte des valeurs par instance. La codification latérale permet d'obtenir les paires propriété-valeur d'une instance.

Ces deux modes offrent l'avantage d'assurer la continuité d'un processus. Si une erreur se produit pendant une communication entre un serveur de paquet et un client de paquet, le serveur de paquet (ou le client de paquet) ne peut recevoir qu'un sous-ensemble de paquets de données. Dans le cas de la codification verticale, si le serveur de paquet (ou le client de paquet) a besoin de l'ensemble de valeurs d'une propriété et que cet ensemble appartient au sous-ensemble des paquets de données, le serveur de paquet (ou le client de paquet) peut

continuer à exécuter le processus. Dans le cas de la codification latérale, un scénario similaire est applicable.

Le présent document définit l'instruction #PWS_CODIFICATION_MODE ci-dessous afin d'indiquer le mode utilisé pour représenter un ensemble conjonctif de paquets de données échangés par l'intermédiaire d'un service Web de paquet.

Mot-clé: #PWS_CODIFICATION_MODE
Nom: mode de codification
Définition: instruction pour un paquet de données indiquant la codification utilisée pour représenter le paquet de données
Description: Si "LATERAL" est spécifié, le paquet de données est représenté dans la codification latérale. Si "VERTICAL" est spécifié, le paquet de données est représenté dans la codification verticale.
Catégorie: facultative – fonctionnelle
Format: Chaque LATERAL ou VERTICAL peut être attribué en tant que valeur.

6.4.2 Langue prévue

Un paquet de données peut contenir des définitions en plusieurs langues. Au moyen d'un service d'enregistrement de paquet, le contenu de toutes les langues d'un ensemble conjonctif de paquets de données est traité côté réception (en général un serveur de paquet), mais cela s'avère parfois dangereux, car il peut contenir des modifications imprévues dans des langues imprévues. Par exemple, lors d'une traduction de l'anglais vers une autre langue, le contenu en anglais peut être utilisé comme simple référence dans la traduction vers une autre langue appelée "variante de langue". Dans ce cas, même si la définition en anglais est modifiée par inadvertance, la modification doit être ignorée sur le serveur de paquet. Par conséquent, la ou les définitions de langue réellement destinées à la modification doivent être clairement marquées. Pour éviter toute modification intempestive du contenu, le présent document définit l'instruction "#INTENDED_LANGUAGE" pour le service d'enregistrement de paquet.

Mot-clé: #INTENDED_LANGUAGE
Nom: langue(s) prévue(s)
Définition: désignation de la ou des langues conformément à l'ISO 639-1, qui contient (contiennent) la modification prévue des définitions
Description: Si un paquet de données contient une langue qui n'est pas spécifiée comme étant une langue prévue, sa définition doit être ignorée côté réception (un serveur de paquet, en général). Si cette instruction n'est pas spécifiée dans un paquet de données, toutes les langues du paquet de données, y compris la langue source, doivent être considérées comme étant prévues. Comme les autres langues, la langue source peut être spécifiée comme un élément de la valeur de cette commande. Si la langue source est spécifiée, la description en langue source contient certaines modifications prévues des définitions.
Catégorie: facultative – fonctionnelle
Format: Le mot-clé "#INTENDED_LANGUAGE" doit être décrit dans la première colonne et le ou les codes de langue doivent être décrits après le mot-clé séparés par le symbole ":=" (deux points-égal). Le code de langue selon l'ISO 639-1, éventuellement suivi d'un code de pays à deux lettres conformément à l'ISO 3166-1, permet d'identifier la langue. Si plusieurs langues sont spécifiées, ces codes de langue doivent être séparés par des virgules et doivent être placés entre accolades "{" et "}". Les cellules de la deuxième colonne et les suivantes doivent être ignorées.

6.4.3 Valeur par défaut

Pour certains environnements de réseau informatique (un réseau mobile et un réseau à faible bande passante, par exemple), il convient de prendre en compte un moyen de réduire la

quantité totale de données à échanger entre un serveur et client afin d'obtenir une communication de réseau efficace et fiable.

L'IEC 62656-1 spécifie deux instructions #DEFAULT_DATA_SUPPLIER et #DEFAULT_DATA_VERSION pour éviter de répéter le même RAI et le même VI, respectivement, des ICID dans la section de données. De la même manière, un moyen de spécifier une valeur par défaut d'une propriété peut être utile.

Le présent document définit l'instruction #DEFAULT_VALUE suivante pour spécifier la valeur par défaut d'une propriété. Si une valeur par défaut est définie sur une colonne de propriété dans une feuille de paquet, la valeur est automatiquement appliquée à toutes les cellules vierges de la section de données de la colonne de propriété.

Mot-clé: #DEFAULT_VALUE
Nom: valeur par défaut
Définition: valeur prédéfinie de la propriété spécifiée par l'ID de propriété
Description: La valeur par défaut de la propriété doit être appliquée à tous les champs vierges de la propriété dans la section de données, dans laquelle les valeurs ne sont pas spécifiées. Pour conserver une valeur null dans un champ, une valeur particulière "(null)", composée d'une chaîne "null" placée entre parenthèses, doit être explicitement décrite dans le champ. Si un paquet de données contenant une valeur par défaut doit être envoyée à un système externe qui n'est pas en mesure de traiter la valeur par défaut, il convient que le paquet de données soit préalablement traité pour remplir les champs vierges avec la valeur par défaut, sauf dans les cas où le champ est explicitement marqué "(null)", avant de l'envoyer réellement au système externe.
Catégorie: facultative – fonctionnelle
Format: Le mot-clé "#DEFAULT_VALUE" est décrit dans la première colonne. Les valeurs par défaut sont décrites dans les cellules correspondant à la deuxième colonne et aux suivantes. Chaque valeur par défaut de ce type correspond à la propriété décrite dans la ligne #PROPERTY_ID.

La Figure 9 indique comment les valeurs par défaut sont décrites et appliquées aux cellules dans la section de données. Les cellules vierges encerclées d'un trait gras indiquent que les valeurs de ces cellules sont remplacées par leurs valeurs par défaut correspondantes lors du traitement des paquets de données.

#PROPERTY_ID	P001	P002	P003	P004	P005
#DATATYPE	STRING_TYPE	STRING_TYPE	LEVEL(MAX) OF INT_TYPE	BOOLEAN_TYPE	ENUM_REAL_ TYPE(E001 (1.0, 2.0, 3.0)
#DEFAULT_ VALUE	**New York**		**2**	**TRUE**	**1.0**
			1.	FALSE	2.0
	Paris				3.0
			3		2.0

IEC

Figure 9 – Exemple d'utilisation des valeurs par défaut

6.5 Description des instructions

L'IEC 62656-1 spécifie les instructions à utiliser dans la section d'en-tête de chaque paquet de données. Ces instructions sont classées en quatre types: "obligatoire", "facultative – fonctionnelle", "facultative – informative" et "commentaire". Conformément à l'IEC 62656-1, "obligatoire" est une catégorie indiquant qu'une instruction doit être présente dans un paquet de données. "facultative – fonctionnelle" est une catégorie indiquant que si une instruction est

présente dans un paquet de données, sa valeur doit être traitée selon la fonction concernée par le mot-clé. "facultative – informative" est une catégorie indiquant que la valeur d'une instruction est juste fournie aux utilisateurs d'un paquet de données en tant que message informatif. Enfin, "commentaire" est une catégorie indiquant qu'une ligne dans laquelle une instruction est décrite doit être interprétée comme une ligne de commentaire. Le service Web spécifié dans le présent document est destiné à être mis en œuvre pour l'échange de données entre un serveur de paquet et un client de paquet. Par conséquent, les instructions des catégories "obligatoire" et "facultative - fonctionnelle" doivent être traitées dans un paquet de données qu'elles échangent.

La chaîne de chaque instruction commence par un croisillon "#" (#CLASS_ID, par exemple) pour décrire l'identificateur de classe d'un paquet de données. Toutefois, XML interdit d'utiliser un croisillon "#" comme caractère de début d'un élément et d'un attribut. Par conséquent, le présent document spécifie des conventions de dénomination en fonction des instructions spécifiées dans l'IEC 62656-1, ce qui permet de les utiliser sans crainte pour la représentation des données en XML ou JSON.

Les conventions de dénomination sont les suivantes:

– Un croisillon "#" de début doit être retiré.
– Un trait de soulignement "_" doit être retiré.
– Chaque caractère alphabétique placé juste après un trait de soulignement "_" doit être remplacé par sa majuscule.
– Les autres caractères alphabétiques doivent être remplacés par leur minuscule.

Par exemple, l'instruction "#SUPER_ALT_CLASSID" est remplacée par " superAltClassid ".

Les descriptions des instructions spécifiées dans le présent document figurent dans le Tableau 10.

Tableau 10 – Description des instructions spécifiées dans l'IEC 62656-1

Mot-clé d'instruction spécifié dans l'IEC 62656-1	Description en XML et JSON
#CLASS_ID	classID
#PARCEL_MODE	parcelMode
#PARCEL_ID	parcelID
#DEFAULT_SUPPLIER	defaultSupplier
#DEFAULT_VERSION	defaultVersion
#DEFAULT_DATA_SUPPLIER	defaultDataSupplier
#DEFAULT_DATA_VERSION	defaultDataVersion
#OBJECT_ID_NAME	objectIDNameobjectIDName
#PROPERTY_ID	propertyID
#REQUIREMENT	requirement
#ID_ENCODE	idEncoding

NOTE Le présent document ne limite pas l'utilisation des mots-clés d'instruction "facultative - informative" dans un service Web. L'Annexe D, Tableau D.1 donne la description de ces mots-clés.

De plus, la traduction de l'instruction spécifiée dans le présent document figure dans le Tableau 11.

Tableau 11 – Description des instructions spécifiées dans le présent document

Mot-clé d'instruction spécifié dans le présent document	Description en XML et JSON
#PWS_CODIFICATION_MODE	pwsCodificationMode
#INTENDED_LANGUAGE	intendedLanguage
#DEFAULT_VALUE	defaultValue

7 Représentation de données en JSON

7.1 Structure de base de la représentation de données en JSON

Conformément à l'ISO/IEC 21778, une valeur JSON est un objet, un tableau, un nombre, une chaîne et les valeurs true, false et null. Chaque objet JSON est représenté par une paire d'accolades "{" et "}" entourant aucune ou plusieurs paires nom et valeur. Chaque tableau JSON est représenté par une paire de crochets "[" et "]" entourant aucun ou plusieurs objets et/ou valeurs.

Un objet JSON et un tableau JSON sont des données structurées. La différence est que l'objet JSON est un ensemble non ordonné de paires nom/valeur, à l'inverse du tableau JSON.

La structure de base de la représentation de données en JSON est décrite à la Figure 10. Selon les recommandations de l'ISO/IEC 21778, il convient que les noms JSON à l'intérieur d'un objet JSON soient uniques. Dans la structure de base d'un ensemble conjonctif de paquets de données décrit à la Figure 8, l'élément "parcel" est dupliqué dans l'élément "conjunctiveParcels" et ne peut donc pas être représenté à l'aide d'un objet JSON. À la place de l'élément "parcel", le présent document spécifie un nom JSON particulier "parcels", qui contient un tableau d'objets JSON, chacun indiquant un paquet de données. Le nom JSON est défini en 7.2. Les autres éléments de la structure de base d'un ensemble conjonctif de paquets de données sont représentés à l'aide d'objets JSON.

Le nom JSON "classHeader" dans le nom JSON "header" est utilisé pour contenir les informations de la section en-tête de classe d'un paquet de données. La représentation de données de la section en-tête de classe en JSON est composée de paires nom/valeur JSON, chacune indiquant une instruction et sa valeur. La spécification des noms JSON pour décrire la section en-tête de classe est indiquée en 7.3.

Le nom JSON "schemaHeader" dans le nom JSON "header" est utilisé pour contenir une liste des informations relatives aux propriétés d'un paquet de données, qui sont le schéma de la section de données. La spécification des noms JSON pour décrire la section en-tête de schéma est indiquée en 7.4.

Le nom JSON "data" est utilisé pour contenir les données de la section de données en notation JSON verticale ou notation JSON latérale. Ces notations JSON sont spécifiées en 7.5.

```
{
  "conjunctiveParcels": {
    "ontoLayer": <AO, MO, DO ou DL>,
    "parcels": [
      {
        "header": {
          "classHeader":{
            <Représentation de données de la section en-tête de classe en JSON>
          },
          "schemaHeader":[
            <Représentation de données de la section en-tête de schéma en JSON>
          ]
        },
        "data":{
          <Représentation de données de la section de données en JSON>
        }
      },
      {
        "header": {
          "classHeader":{
            <Représentation de données de la section en-tête de classe en JSON>
          },
          "schemaHeader":[
            <Représentation de données de la section en-tête de schéma en JSON>
          ]
        },
        "data":{
          <Représentation de données de la section de données en JSON>
        }
      },
      ...
    ]
  }
}
```

IEC

Figure 10 – Structure de base de la représentation de données en JSON

Le schéma JSON de description des données JSON en notation JSON verticale est indiqué à l'Annexe A, A.1.1, celui utilisé pour décrire les données JSON en notation JSON latérale étant indiqué en A.1.2. Le schéma JSON pour la notation JSON des exceptions est spécifié en A.1.3.

7.2 Nom JSON réservé indiquant un tableau de paquets de données

Nom JSON: parcels
Nom: paquet de données
Définition: mot-clé réservé pour décrire un tableau de paquets de données, chacun étant décrit comme un objet JSON
Occurrence: 1 pour une occurrence du mot-clé "conjunctiveParcels"

7.3 Noms JSON pour la section en-tête de classe

7.3.1 Nom JSON indiquant l'instruction "#CLASS_ID"

Nom JSON: classID
Définition: nom JSON réservé pour décrire une valeur de l'instruction nommée "#CLASS_ID" dans un paquet de données
Type de données: Chaîne
Obligation: Obligatoire
Exemple: "classID": "MDC_C002"

7.3.2 Nom JSON indiquant l'instruction "#PARCEL_MODE"

Nom JSON: parcelMode
Définition: nom JSON réservé pour décrire une valeur de l'instruction nommée "#PARCEL_MODE" dans un paquet de données
Description: Si ce mot-clé est présent, sa valeur peut être FULL, UPDATE ou PARTIAL. Si un paquet de données est utilisé dans un service d'enregistrement de paquet, ce mot-clé doit être présent.
Type de données: Chaîne
Obligation: Facultatif
Exemple: "parcelMode": "PARTIAL"

7.3.3 Nom JSON indiquant l'instruction "#PARCEL_ID"

Nom JSON: parcelID
Définition: nom JSON réservé pour décrire une valeur de l'instruction nommée "#PARCEL_ID" dans un paquet de données
Type de données: Chaîne
Obligation: Facultatif
Exemple: "parcelID": "2006-06-25 08:19:49"

7.3.4 Nom JSON indiquant l'instruction "#DEFAULT_SUPPLIER"

Nom JSON: defaultSupplier
Définition: nom JSON réservé pour décrire une valeur de l'instruction nommée "#DEFAULT_SUPPLIER" dans un paquet de données
Type de données: Chaîne
Obligation: Facultatif
Exemple: "defaultSupplier": "0112/2///62656_1"

7.3.5 Nom JSON indiquant l'instruction "#DEFAULT_VERSION"

Nom JSON: defaultVersion
Définition: nom JSON réservé pour décrire une valeur de l'instruction nommée "#DEFAULT_VERSION" dans un paquet de données
Type de données: Chaîne
Obligation: Facultatif
Exemple: "defaultVersion": "1"

7.3.6 Nom JSON indiquant l'instruction "#OBJECT_ID_NAME"

Nom JSON: objectIDName
Définition: nom JSON réservé pour décrire une valeur de l'instruction nommée "#OBJECT_ID_NAME" dans un paquet de données
Description: Si ce mot-clé est présent, sa valeur peut être "GUID" ou "UUID".
Type de données: Chaîne
Obligation: Facultatif
Exemple: "objectIDName": "GUID"

7.3.7 Nom JSON indiquant l'instruction "#ID_ENCODE"

Nom JSON: idEncode
Définition: nom JSON réservé pour décrire une valeur de l'instruction nommée "#ID_ENCODE" dans un paquet de données
Type de données: Chaîne
Obligation: Facultatif
Exemple: "idEncode": "ICID"

7.3.8 Nom JSON indiquant l'instruction "#PWS_CODIFICATION_MODE"

Nom JSON: pwsCodificationMode
Définition: nom JSON réservé pour décrire une valeur de l'instruction nommée "#PWS_CODIFICATION_MODE" dans un paquet de données
Description: Si ce mot-clé est présent, sa valeur doit être "VERTICAL" ou "LATERAL".
Type de données: Chaîne
Obligation: Obligatoire
Exemple: "pwsCodificationMode": "LATERAL"

7.3.9 Nom JSON indiquant l'instruction "#INTENDED_LANGUAGE"

Nom JSON: intendedLanguage
Définition: nom JSON réservé pour décrire une valeur de l'instruction nommée "#INTENDED_LANGUAGE" dans un paquet de données
Description Si plusieurs langues sont spécifiées, chacune d'elles est séparée par une virgule (",").
Type de données: Chaîne
Obligation: Facultatif
Exemple: "intendedLanguage": "en,fr"

7.4 Noms JSON pour la section en-tête de schéma

7.4.1 Structure de base de la représentation de données d'une section en-tête de schéma en JSON

Le nom JSON "schemaHeader" comporte des objets JSON, chacun donnant des informations relatives à une propriété utilisée comme schéma d'une colonne pour les cellules. La structure de la représentation de données d'une section en-tête de schéma en JSON est décrite à la Figure 11. Les informations d'une propriété contenue dans chaque objet JSON sont spécifiées en 7.4.2.

```
"schemaHeader": [
  {
    <informations relatives à une propriété en tant que schéma d'une
colonne pour les cellules>
  },
  {
    <informations relatives à une propriété en tant que schéma d'une
colonne pour les cellules>
  },
   …
]
```

IEC

Figure 11 – Structure de base de la représentation de données d'une section en-tête de schéma en JSON

Un exemple de représentation de données d'une section en-tête de schéma en JSON est donné à l'Annexe A, Article C.2.

7.4.2 Noms JSON pour la section en-tête de schéma

7.4.2.1 Nom JSON indiquant une valeur de l'instruction "#PROPERTY_ID"

Nom JSON: propertyID
Définition: nom JSON réservé pour décrire une valeur de l'instruction nommée "#PROPERTY_ID" dans un paquet de données
Type de données: Chaîne
Obligation: Obligatoire
Exemple: "propertyID": "MDC_P001_5"

7.4.2.2 Nom JSON indiquant une valeur de l'instruction "#DEFAULT_DATA_SUPPLIER"

Nom JSON: defaultDataSupplier
Définition: nom JSON réservé pour décrire une valeur de l'instruction nommée "#DEFAULT_DATA_SUPPLIER" dans un paquet de données
Type de données: Chaîne
Obligation: Facultatif
Exemple: "defaultDataSupplier": "0112/2///61360_4"

7.4.2.3 Nom JSON indiquant une valeur de l'instruction "#DEFAULT_DATA_VERSION"

Nom JSON: defaultDataVersion
Définition: nom JSON réservé pour décrire une valeur de l'instruction nommée "#DEFAULT_DATA_VERSION" dans un paquet de données
Type de données: Chaîne
Obligation: Facultatif
Exemple: "defaultDataVersion": "1"

7.4.2.4 Nom JSON indiquant l'instruction "#DEFAULT_VALUE"

Nom JSON: defaultValue
Définition: nom JSON réservé pour décrire une valeur de l'instruction nommée "#DEFAULT_VALUE" dans un paquet de données
Type de données: Chaîne
Obligation: Facultatif
Exemple: "defaultValue": "New York"

7.4.2.5 Nom JSON indiquant une valeur de l'instruction "#REQUIREMENT"

Nom JSON: Requirement
Définition: nom JSON réservé pour décrire une valeur de l'instruction nommée "#REQUIREMENT" dans un paquet de données
Description: Si ce mot-clé est présent, sa valeur doit être vide ou l'une des suivantes: "CONST", "KEY", "NOT_NULL", "MANDATORY", "OPTIONAL" ou "OBSOLETE". Si sa valeur est vierge, cela équivaut à la désignation "OPTIONAL", alors que les valeurs "MANDATORY", "OPTIONAL" et "OBSOLETE" peuvent être abrégées en "MAND", "OPT" et "OBS", respectivement. Les détails de chaque valeur sont spécifiés en 5.11.24 de l'IEC 62656-1:2014.
Type de données: Chaîne
Obligation: Facultatif
Exemple: "requirement": "KEY"

7.5 Représentation de données de la section de données en JSON

7.5.1 Notation JSON verticale de la section de données

En notation JSON verticale, le nom JSON "data" contient une paire de noms JSON "values" et un objet JSON comme valeur, contenant un ensemble non ordonné de paires nom/valeur JSON, chacune indiquant un identificateur de propriété et ses valeurs. Le nom JSON "data" contient également les noms JSON facultatifs "defaultSupplier" et "defaultVersion" et leurs valeurs. Si le nom JSON "defaultSupplier" et sa valeur sont présents dans la section de données, le RAI de chaque identificateur de propriété utilisé comme nom JSON dans la valeur du nom JSON "values" peut être ignoré et doit être complété par un récepteur de données. De la même manière, si le nom JSON "defaultVersion" et sa valeur sont présents dans la section de données, le VI de chaque identificateur de propriété utilisé comme nom JSON dans la valeur du nom JSON "values" peut être ignoré et doit être complété par un récepteur de données.

Si une valeur est nulle, "null" en tant que valeur JSON doit être décrit dans un tableau JSON.

Un exemple de notation JSON verticale de la section de données est donné à l'Article C.1 et en C.2.1.

7.5.2 Notation JSON latérale de la section de données

En notation JSON latérale, le nom JSON "data" contient une paire de noms JSON "instances" et un tableau JSON comme valeur, dont chaque élément est un tableau JSON de propriété-valeurs d'une instance. L'ordre et le nombre d'éléments de chaque tableau JSON de propriété-valeurs doivent être les mêmes que ceux des propriétés dans la section en-tête de schéma.

Un exemple de notation JSON latérale de la section de données est donné à l'Article C.1 et en C.2.2.

7.6 Codage de caractère

L'ISO/IEC 21778 recommande l'utilisation d'un codage de caractère UTF-8, UTF-16 ou UTF-32. Conformément à cette spécification, il convient que toutes les données textuelles en JSON échangées dans le service Web de paquet soient codées en UTF-8, UTF-16 ou UTF-32.

8 Représentation de données en XML

8.1 Structure de base de la représentation de données en XML

Conformément à la spécification XML fournie par le World Wide Web Consortium (W3C), une valeur peut être décrite de deux façons: par un élément XML et par un attribut XML. La spécification ne formule aucune règle ni recommandation quant à celle qu'il convient d'utiliser. Toutefois, un élément XML permet de représenter les données structurées. Par conséquent, comme base de leur utilisation, le présent document spécifie l'utilisation d'un élément XML pour décrire les éléments de la structure de base d'un ensemble conjonctif de paquets de données décrit à la Figure 2. Un attribut XML permet de représenter une valeur supplémentaire d'un élément.

La structure de base de la représentation de données en XML est décrite à la Figure 12. Dans la structure de base d'un ensemble conjonctif de paquets de données décrit à la Figure 8, tous les éléments sont représentés à l'aide d'éléments XML. Une couche d'ontologie de paquet présentée comme une valeur de "ontoLayer" est une information supplémentaire pour un ensemble conjonctif de paquets et est donc représentée comme un attribut XML.

L'élément XML "classHeader" dans l'élément XML "header" est utilisé pour contenir les informations de la section en-tête de classe d'un paquet de données. L'élément XML de la section en-tête de classe est composé d'éléments XML, chacun indiquant une instruction et sa valeur. La spécification de la représentation de données de la section en-tête de classe en XML est présentée en 8.3.

L'élément XML "schemaHeader" dans l'élément XML "header" est utilisé pour contenir une liste des informations relatives aux propriétés d'un paquet de données, qui sont le schéma de la section de données. La spécification de la représentation de données de la section en-tête de schéma en XML est présentée en 8.3.9.

L'élément XML "data" est utilisé pour contenir les données de la section de données en notation XML verticale ou notation XML latérale. Ces notations XML sont spécifiées en 8.5.

```
<conjunctiveParcels ontLayer="<AO, MO, DO ou DL>">
   <parcel>
      <header>
         <classHeader>
            <Représentation de données de la section en-tête de classe en XML>
         </classHeader>
         <schemaHeader>
            <Représentation de données de la section en-tête de schéma en XML>
         </schemaHeader>
      </header>
      <data>
         <Représentation de données de la section de données en XML>
      </data>
   </parcel>
</conjunctiveParcels>
```

IEC

Figure 12 – Structure de base de la représentation de données en XML

Le schéma XML de représentation des données en notation XML verticale est indiqué en A.2.1, celui utilisé pour représenter les données en notation XML latérale étant indiqué en A.2.2. Le schéma XML correspondant à la notation XML des exceptions est spécifié en A.2.3.

8.2 Mot-clé réservé indiquant le paquet de données

Mot-clé: parcel
Nom: paquet de données
Définition: élément XML réservé pour décrire un paquet de données
Occurrence: 1..n pour une occurrence du mot-clé "conjunctiveParcels"

8.3 Éléments XML pour la section en-tête de classe

8.3.1 Élément XML indiquant l'instruction "#CLASS_ID"

Mot-clé: classID
Définition: élément XML réservé pour décrire une valeur de l'instruction nommée "#CLASS_ID" dans un paquet de données
Type de données: Chaîne
Obligation: Obligatoire
Exemple: <classID>MDC_C002</classID>

8.3.2 Élément XML indiquant l'instruction "#PARCEL_MODE"

Mot-clé: parcelMode
Définition: élément XML réservé pour décrire une valeur de l'instruction nommée "#PARCEL_MODE" dans un paquet de données
Description: Si cet élément XML est présent, sa valeur peut être "FULL", "UPDATE" ou "PARTIAL". Si un paquet de données est utilisé dans un service d'enregistrement de paquet, cet élément doit être présent.
Type de données: Chaîne
Obligation: Facultatif
Exemple: <parcelMode>PARTIAL</parcelMode>

8.3.3 Élément XML indiquant l'instruction "#PARCEL_ID"

Mot-clé: parcelID
Définition: élément XML réservé pour décrire une valeur de l'instruction nommée "#PARCEL_ID" dans un paquet de données
Type de données: Chaîne
Obligation: Facultatif
Exemple: <parcelID>2006-06-25 08:19:49</parcelID>

8.3.4 Élément XML indiquant l'instruction "#DEFAULT_SUPPLIER"

Mot-clé: defaultSupplier
Définition: élément XML réservé pour décrire une valeur de l'instruction nommée "#DEFAULT_SUPPLIER" dans un paquet de données
Type de données: Chaîne
Obligation: Facultatif
Exemple: <defaultSupplier>0112/2///62656_1</defaultSupplier>

8.3.5 Élément XML indiquant l'instruction "#DEFAULT_VERSION"

Mot-clé: defaultVersion
Définition: élément XML réservé pour décrire une valeur de l'instruction nommée "#DEFAULT_VERSION" dans un paquet de données
Type de données: Chaîne
Obligation: Facultatif
Exemple: <defaultVersion>1</defaultVersion>

8.3.6 Élément XML indiquant l'instruction "#OBJECT_ID_NAME"

Mot-clé: objectIDName
Définition élément XML réservé pour décrire une valeur de l'instruction nommée "#OBJECT_ID_NAME" dans un paquet de données
Description: Si cet élément est présent, sa valeur peut être "GUID" ou "UUID".
Type de données: Chaîne
Obligation: Facultatif
Exemple: <objectIDName>GUID</objectIDName>

8.3.7 Élément XML indiquant l'instruction "#ID_ENCODE"

Mot-clé: idEncode
Définition: élément XML réservé pour décrire une valeur de l'instruction nommée "#ID_ENCODE" dans un paquet de données
Type de données: Chaîne
Obligation: Facultatif
Exemple: <idEncode>ICID</idEncode>

8.3.8 Élément XML indiquant l'instruction "#PWS_CODIFICATION_MODE"

Mot-clé: pwsCodificationMode
Définition élément XML réservé pour décrire une valeur de l'instruction nommée "#PWS_CODIFICATION_MODE" dans un paquet de données
Description: si cet élément est présent, sa valeur doit être "VERTICAL" ou "LATERAL".
Type de données: Chaîne
Obligation: Obligatoire
Exemple: <pwsCodificationMode>LATERAL</pwsCodificationMode>

8.3.9 Élément XML indiquant l'instruction "#INTENDED_LANGUAGE"

Mot-clé: intendedLanguage
Définition élément XML réservé pour décrire une valeur de l'instruction nommée "#INTENDED_LANGUAGE" dans un paquet de données
Description: Si plusieurs langues sont spécifiées, chacune d'elles est séparée par une virgule (",").
Type de données: Chaîne
Obligation: Facultatif
Exemple: <intendedLanguage>en,fr</intendedLanguage>

8.4 Éléments XML pour la section en-tête de schéma

8.4.1 Structure de base de la représentation de données d'une section en-tête de schéma en XML

L'élément XML "schemaHeader" comporte des éléments XML "property", chacun donnant des informations relatives à une propriété utilisée comme schéma d'une colonne pour les cellules. La structure de la représentation de données d'une section en-tête de schéma en XML est décrite à la Figure 13. Les informations d'une propriété contenue dans chaque élément XML "property" sont spécifiées en 8.4.2.

```
<schemaHeader>
   <property>
     <informations relatives à une propriété en tant que schéma d'une colonne
pour les cellules>
   <property>
   <property>
     <informations relatives à une propriété en tant que schéma d'une colonne
pour les cellules>
   <property>
   …
</schemaHeader>
```

IEC

Figure 13 – Structure de base de la représentation de données d'une section en-tête de schéma en XML

8.4.2 Éléments XML de la section en-tête de schéma

8.4.2.1 Élément XML indiquant l'instruction "#PROPERTY_ID"

Mot-clé: propertyID
Définition: élément XML réservé pour décrire une valeur de l'instruction nommée "#PROPERTY_ID" dans un paquet de données
Type de données: Chaîne
Obligation: Obligatoire
Exemple: <propertyID>MDC_P001_5</propertyID>

8.4.2.2 Élément XML indiquant l'instruction "#DEFAULT_DATA_SUPPLIER"

Mot-clé: defaultDataSupplier
Définition: élément XML réservé pour décrire une valeur de l'instruction nommée "#DEFAULT_DATA_SUPPLIER" dans un paquet de données
Type de données: Chaîne
Obligation: Facultatif
Exemple: <defaultDataSupplier>MDC_P001_5</defaultDataSupplier>

8.4.2.3 Élément XML indiquant l'instruction "#DEFAULT_DATA_VERSION"

Mot-clé: defaultDataVersion
Définition: élément XML réservé pour décrire une valeur de l'instruction nommée "#DEFAULT_DATA_VERSION" dans un paquet de données
Type de données: Chaîne
Obligation: Facultatif
Exemple: <defaultDataVersion>1</defaultDataVersion>

8.4.2.4 Élément XML indiquant l'instruction "#DEFAULT_VALUE"

Mot-clé: defaultValue
Définition: élément XML réservé pour décrire une valeur de l'instruction nommée "#DEFAULT_VALUE" dans un paquet de données
Type de données: Chaîne
Obligation: Facultatif
Exemple: <defaultValue>New York</defaultValue>

8.4.2.5 Élément XML indiquant l'instruction "#REQUIREMENT"

Mot-clé: requirement
Définition: élément XML réservé pour décrire une valeur de l'instruction nommée "#REQUIREMENT" dans un paquet de données
Description: Si ce mot-clé est présent, sa valeur doit être vide ou l'une des suivantes: "CONST", "KEY", "NOT_NULL", "MANDATORY", "OPTIONAL" ou "OBSOLETE. Si sa valeur est vierge, cela équivaut à la désignation "OPTIONAL", alors que les valeurs "MANDATORY", "OPTIONAL" et "OBSOLETE" peuvent être abrégées en "MAND", "OPT" et "OBS", respectivement. Les détails de chaque valeur sont spécifiés en 5.11.24 de l'IEC 62656-1:2014.
Type de données: Chaîne
Obligation: Facultatif
Exemple: <requirement>KEY</requirement>

8.5 Éléments et attributs XML de la section de données

8.5.1 Notation XML verticale de la section de données

8.5.1.1 Structure de base de la représentation de données d'une section de données en notation XML verticale

En notation XML verticale, l'élément XML "data" contient des éléments XML "values", chacun étant identifié par son attribut "propertyID", dans lequel l'identificateur de propriété est spécifié. Chaque élément XML "values" contient également un ensemble ordonné d'éléments XML "value". Chaque élément XML "value" contient une propriété-valeur.

L'élément XML "data" contient les attributs XML "defaultSupplier" et "defaultVersion". Si l'attribut XML "defaultSupplier" est présent dans la section de données, le RAI de chaque identificateur de propriété spécifié dans l'attribut XML "propertyID" de l'élément XML "values" peut être ignoré et doit être complété par un récepteur de données. De la même manière, si l'attribut XML "defaultVersion" est présent dans la section de données, le VI de chaque identificateur de propriété spécifié dans l'attribut XML "propertyID" de l'élément XML "values" peut être ignoré et doit être complété par un récepteur de données.

Par conséquent, la représentation de données de la section de données en notation XML verticale est structurée comme cela est décrit à la Figure 14.

```
<data defaultSupplier="<RAI>" defaultVersion="<VI>">
  <values propertyID="<property ID>">
    <valeur de la propriété spécifiée en notation XML verticale>
    <valeur de la propriété spécifiée en notation XML verticale>
    …
  </values>
  <values propertyID="<property ID>">
    <valeur de la propriété spécifiée en notation XML verticale>
    <valeur de la propriété spécifiée en notation XML verticale>
    …
  </values>
  …
</data>
```

IEC

Figure 14 – Structure de la représentation de données d'une section de données en notation XML verticale

Un exemple de représentation de données en notation XML verticale est donné à l'Article C.1 et en C.3.1.

8.5.1.2 Élément XML indiquant les valeurs d'une propriété en codification verticale

Mot-clé: values
Définition: élément XML réservé en mode de codification verticale pour décrire une liste des valeurs d'une propriété
Description: L'identificateur d'une propriété est spécifié dans l'attribut "propertyID" de cet élément.
Obligation: Obligatoire
Exemple:

```
<values propertyID="MDC_P001_5">
  <value>AAA001</value>
  <value>AAA002</value>
  <value>AAA003</value>
  <value>AAA013</value>
</values>
```

8.5.1.3 Attribut XML indiquant un identificateur de propriété pour une valeur en codification verticale

Mot-clé: propertyID
Définition: attribut XML réservé de l'élément XML "values" en mode de codification verticale, décrivant l'identificateur d'une propriété dont les valeurs figurent dans l'élément
Description: Si le RAI de l'identificateur est ignoré, il doit être complété par la valeur de l'attribut "defaultSupplier" de l'élément "data". Si le VI de l'identificateur est ignoré, il doit être complété par la valeur de l'attribut "defaultVersion" de l'élément "data".
Type de données: Chaîne
Obligation: Obligatoire
Exemple:

```
<values propertyID="MDC_P001_5">
  <value>AAA001</value>
  <value>AAA002</value>
  <value>AAA003</value>
  <value>AAA013</value>
</values>
```

8.5.1.4 Élément XML indiquant une valeur de propriété en codification verticale

Mot-clé: value
Définition: élément XML réservé en mode de codification verticale pour décrire une valeur de la propriété spécifiée dans l'attribut "propertyID" de l'élément parent "values" de cet élément
Description: Si une valeur est vierge ou nulle, il convient de décrire l'élément comme "<value></value>" ou "<value/>".
Type de données: Toutes
Obligation: Obligatoire
Exemple: <value>composant</value>

8.5.2 Notation XML latérale de la section de données

8.5.2.1 Structure de base de la représentation de données d'une section de données en notation XML latérale

En notation XML latérale, l'élément XML "data" contient des éléments XML "instance", chacun contenant des éléments XML "value". Chaque élément XML "value" contient une propriété-valeur et comporte un attribut XML "propertyID" qui spécifie l'identificateur de propriété de la valeur.

Comme pour la notation XML verticale de la section de données, l'élément XML "data" contient les attributs XML "defaultSupplier" et "defaultVersion". Si l'attribut XML "defaultSupplier" est présent dans la section de données, le RAI de l'identificateur de propriété spécifié dans l'attribut XML "propertyID" de chaque élément XML "value" peut être ignoré et doit être complété par un récepteur de données. De la même manière, si l'attribut XML "defaultVersion" est présent dans la section de données, le VI de l'identificateur de propriété spécifié dans l'attribut XML "propertyID" de chaque élément XML "value" peut être ignoré et doit être complété par un récepteur de données.

Par conséquent, la représentation de données de la section de données en notation XML latérale est structurée comme cela est décrit à la Figure 15.

```
<data defaultSupplier="<RAI>" defaultVersion="<VI>">
  </instance>
      <valeur d'une instance en notation XML latérale>
      <valeur d'une instance en notation XML latérale>
      …
  </instance>
  <instance>
      <valeur d'une instance en notation XML latérale>
      <valeur d'une instance en notation XML latérale>
      …
  </instance>
    …
</data>
```

IEC

Figure 15 – Structure de la représentation de données d'une section de données en notation XML latérale

Un exemple de représentation de données en notation XML latérale est donné à l'Article C.1 et en C.3.2.