



BSI Standards Publication

Standard interface for connecting charging stations to local energy management systems.

Part 3 Communication protocol and cybersecurity specific aspects

National foreword

This British Standard is the UK implementation of EN IEC 63380-3:2025. It is identical to IEC 63380-3:2025.

The UK participation in its preparation was entrusted to Technical Committee PEL/69, Electric vehicles.

A list of organizations represented on this committee can be obtained on request to its committee manager.

Contractual and legal considerations

This publication has been prepared in good faith, however no representation, warranty, assurance or undertaking (express or implied) is or will be made, and no responsibility or liability is or will be accepted by BSI in relation to the adequacy, accuracy, completeness or reasonableness of this publication. All and any such responsibility and liability is expressly disclaimed to the full extent permitted by the law.

This publication is provided as is, and is to be used at the recipient's own risk.

The recipient is advised to consider seeking professional guidance with respect to its use of this publication.

This publication is not intended to constitute a contract. Users are responsible for its correct application.

© The British Standards Institution 2025
Published by BSI Standards Limited 2025

ISBN 978 0 539 22341 5
ICS 29.240.99; 43.120

Compliance with a British Standard cannot confer immunity from legal obligations.

This British Standard was published under the authority of the Standards Policy and Strategy Committee on 30 September 2025.

Amendments/corrigenda issued since publication

Date	Text affected

English Version

Standard interface for connecting charging stations to local
energy management systems - Part 3 Communication protocol
and cybersecurity specific aspects
(IEC 63380-3:2025)

Interface normale pour la connexion de bornes de charge
aux systèmes locaux de gestion de l'énergie - Partie 3:
Protocole de communication et aspects spécifiques liés à la
cybersécurité
(IEC 63380-3:2025)

Standardschnittstelle zum Anschluss von Ladestationen an
lokale Energiemanagementsysteme - Teil 3:
Kommunikationsprotokoll und spezifische Aspekte der
Cybersecurity
(IEC 63380-3:2025)

This European Standard was approved by CENELEC on 2025-08-12. CENELEC members are bound to comply with the CEN/CENELEC Internal Regulations which stipulate the conditions for giving this European Standard the status of a national standard without any alteration.

Up-to-date lists and bibliographical references concerning such national standards may be obtained on application to the CEN-CENELEC Management Centre or to any CENELEC member.

This European Standard exists in three official versions (English, French, German). A version in any other language made by translation under the responsibility of a CENELEC member into its own language and notified to the CEN-CENELEC Management Centre has the same status as the official versions.

CENELEC members are the national electrotechnical committees of Austria, Belgium, Bulgaria, Croatia, Cyprus, the Czech Republic, Denmark, Estonia, Finland, France, Germany, Greece, Hungary, Iceland, Ireland, Italy, Latvia, Lithuania, Luxembourg, Malta, the Netherlands, Norway, Poland, Portugal, Republic of North Macedonia, Romania, Serbia, Slovakia, Slovenia, Spain, Sweden, Switzerland, Türkiye and the United Kingdom.



European Committee for Electrotechnical Standardization
Comité Européen de Normalisation Electrotechnique
Europäisches Komitee für Elektrotechnische Normung

CEN-CENELEC Management Centre: Rue de la Science 23, B-1040 Brussels

European foreword

The text of document 69/1051/FDIS, future edition 1 of IEC 63380-3, prepared by TC 69 "Electrical power/energy transfer systems for electrically propelled road vehicles and industrial trucks" was submitted to the IEC-CENELEC parallel vote and approved by CENELEC as EN IEC 63380-3:2025.

The following dates are fixed:

- latest date by which the document has to be implemented at national (dop) 2026-09-30 level by publication of an identical national standard or by endorsement
- latest date by which the national standards conflicting with the (dow) 2028-09-30 document have to be withdrawn

Attention is drawn to the possibility that some of the elements of this document may be the subject of patent rights. CENELEC shall not be held responsible for identifying any or all such patent rights.

This document has been prepared under a standardization request addressed to CENELEC by the European Commission. The Standing Committee of the EFTA States subsequently approves these requests for its Member States.

Any feedback and questions on this document should be directed to the users' national committee. A complete listing of these bodies can be found on the CENELEC website.

Endorsement notice

The text of the International Standard IEC 63380-3:2025 was approved by CENELEC as a European Standard without any modification.

Annex ZA (normative)

Normative references to international publications with their corresponding European publications

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.

NOTE 1 Where an International Publication has been modified by common modifications, indicated by (mod), the relevant EN/HD applies.

NOTE 2 Up-to-date information on the latest versions of the European Standards listed in this annex is available here: www.cenelec.eu.

<u>Publication</u>	<u>Year</u>	<u>Title</u>	<u>EN/HD</u>	<u>Year</u>
IEC 62394	-	Service diagnostic interface for consumer electronics products and networks - Implementation for echonet	-	-
IEC 63380-2	-	Standard interface for connecting charging stations to local energy management systems - Part 2: Specific data model mapping	IEC 63380-2	-
ISO/IEC 14543-4-3	2015	Information technology - Home Electronic Systems (HES) architecture - Part 4-3: Application layer interface to lower communications layers for network enhanced control devices of HES Class 1	-	-
IETF RFC 793	1981	Transmission Control Protocol - DARPA Internet Program Protocol Specification	-	-
IETF RFC 3280	2002	Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile	-	-
IETF RFC 6455	2011	The WebSocket Protocol	-	-
IETF RFC 6763	-	DNS-Based Service Discovery	-	-
IETF RFC 5246	-	The Transport Layer Security (TLS) Protocol Version 1.2	-	-
IETF RFC 5289	-	TLS Elliptic Curve Cipher Suites with SHA-256/384 and AES Galois Counter Mode (GCM)	-	-
IETF RFC 8422	-	Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS) Versions 1.2 and Earlier	-	-

CONTENTS

FOREWORD.....	6
INTRODUCTION.....	8
1 Scope.....	10
2 Normative references	10
3 Terms, definitions, and abbreviated terms	11
3.1 Terms and definitions	11
3.2 Abbreviated terms	14
4 Overview	15
5 SPINE protocol.....	15
5.1 General.....	15
5.2 Architecture overview	16
5.2.1 General rules	16
5.2.2 Common data types	16
5.2.3 Address level details	21
5.3 SPINE datagram	22
5.3.1 Overview	22
5.3.2 Header	23
5.3.3 Payload	31
5.4 Communication modes	45
5.4.1 General	45
5.4.2 Simple communication mode	46
5.4.3 Enhanced communication mode	46
5.5 Functional commissioning	47
5.5.1 General	47
5.5.2 Detailed discovery.....	47
5.5.3 Destination list	63
5.5.4 Binding	66
5.5.5 Subscription.....	75
5.5.6 Use case discovery	82
6 SHIP	85
6.1 Architecture overview	85
6.1.1 General	85
6.1.2 General considerations on closing communication channels.....	87
6.1.3 SHIP node parameters	87
6.2 Registration	88
6.2.1 General	88
6.2.2 Successful registration	89
6.2.3 Registration details and recommendations (informative).....	89
6.3 Reconnection	90
6.3.1 General	90
6.3.2 Reconnection details in case of changed key material (informative).....	90
6.4 Discovery.....	91
6.4.1 General	91
6.4.2 Service instance	91
6.4.3 Service name.....	91
6.4.4 Multicast DNS name.....	92

6.4.5	Recommendations for re-discovery	94
6.5	TCP	95
6.5.1	General	95
6.5.2	Limited connection capabilities	95
6.5.3	Online detection	95
6.5.4	TCP connection establishment	96
6.5.5	Retransmission timeout	96
6.6	TLS	96
6.6.1	General	96
6.6.2	Cipher suites	97
6.6.3	Maximum fragment length	98
6.6.4	TLS compression	98
6.6.5	Renegotiation	98
6.6.6	Session resumption	98
6.6.7	TLS extension for ECC	99
6.6.8	TLS probing	100
6.7	WebSocket	100
6.7.1	General	100
6.7.2	TLS dependencies	100
6.7.3	Opening handshake	101
6.7.4	Data framing	101
6.7.5	Keep-alive connection	101
6.8	Message representation using JSON text format	102
6.8.1	General	102
6.8.2	Definitions	102
6.8.3	Examples for each type	103
6.8.4	XML to JSON transformation	103
6.8.5	JSON to XML transformation	109
6.9	Key management	110
6.9.1	General	110
6.9.2	Certificates	110
6.9.3	SHIP node specific public key	115
6.9.4	Verification procedure	117
6.9.5	Symmetric key	123
6.9.6	SHIP node PIN	124
6.9.7	SHIP commissioning tool	125
6.9.8	QR code	127
6.10	SHIP data exchange	130
6.10.1	General	130
6.10.2	Terms in the context of SHIP data exchange	131
6.10.3	Protocol architecture/hierarchy	132
6.10.4	SHIP message exchange	133
6.11	Well-known protocolId	173
7	ECHONET Lite	173
Annex A (normative)	SHIP XSD	175
Bibliography		180

Figure 1 – Overview of communication protocols within IEC 63380-3	15
Figure 2 – PossibleOperationsType.....	19
Figure 3 – DeviceAddressType.....	20
Figure 4 – EntityAddressType	20
Figure 5 – FeatureAddressType	20
Figure 6 – SPINE datagram	23
Figure 7 – SPINE header	24
Figure 8 – SPINE payload.....	32
Figure 9 – Example of selectors part (extract) with entity address part	44
Figure 10 – Communication modes of SPINE devices A, B and C	45
Figure 11 – Discovery example	47
Figure 12 – Hierarchy types	48
Figure 13 – Function Discovery Example over Feature Description	49
Figure 14 – nodeManagementDetailedDiscoveryData function overview, part 1	52
Figure 15 – nodeManagementDetailedDiscoveryData function overview, part 2: deviceInformation.description.....	53
Figure 16 – nodeManagementDetailedDiscoveryData function overview, part 3: entityInformation.description	53
Figure 17 – nodeManagementDetailedDiscoveryData function overview, part 4: featureInformation.description	54
Figure 18 – nodeManagementDestinationListData function overview, part 1	65
Figure 19 – nodeManagementDestinationListData function overview, part 2.....	65
Figure 20 – Binding request	68
Figure 21 – nodeManagementBindingRequestCall function overview	68
Figure 22 – nodeManagementBindingData function overview	70
Figure 23 – nodeManagementBindingDeleteCall function overview	72
Figure 24 – Subscription request.....	76
Figure 25 – nodeManagementSubscriptionRequestCall function overview	76
Figure 26 – nodeManagementSubscriptionData function overview	78
Figure 27 – nodeManagementSubscriptionDeleteCall function overview	80
Figure 28 – nodeManagementUseCaseData function.....	83
Figure 29 – Physical connections in the overall system.....	86
Figure 30 – SHIP stack overview.....	86
Figure 31 – Full TLS 1.2 handshake with mutual authentication	97
Figure 32 – Quick TLS Handshake with Session Resumption.....	99
Figure 33 – Easy mutual authentication with QR codes and smart phone	124
Figure 34 – QR code model 2, "low" error correction code level, 0,33mm/module, with SKI and PIN.....	129
Figure 35 – QR code model 2, "low" error correction code level, 0,33 mm/module, with all values.....	130
Figure 36 – QR code model 2, "low" error correction code level, 0,33 mm/module, with brainpoolP256r1 SKI and brainpoolP384r1 SKI.....	130
Figure 37 – Protocol architecture and hierarchy.....	132
Figure 38 – CMI Message sequence example.....	136

Figure 39 – Connection state "hello" sequence example without prolongation request: "A" and "B" already trust each other; "B" is slower/delayed	143
Figure 40 – Connection state "hello" sequence example with prolongation request.....	144
Figure 41 – Connection State "Protocol Handshake" message sequence example.....	149
Figure 42 – Connection state "PIN verification" message sequence example (begin).....	158
Figure 43 – ECHONET Lite frame format.....	174
Table 1 – Structure of the SPINE datagram	23
Table 2 – cmdClassifier values and kind of messages for a message "M" and scope of related acknowledgement messages	27
Table 3 – Structure of the SPINE header	30
Table 4 – Elements of the SPINE payload	32
Table 5 – Example table (template).....	36
Table 6 – Considered cmdOptions combinations for classifier "write"	37
Table 7 – Considered cmdOptions combinations for classifier "notify"	38
Table 8 – Considered cmdOptions combinations for classifier "read".....	39
Table 9 – Considered cmdOptions combinations for classifier "reply"	39
Table 10 – Address path examples.....	43
Table 11 – Notify/response list of entities and their corresponding features with nodeManagementDetailedDiscoveryData	54
Table 12 – nodeManagementDetailedDiscoveryDataSelectors	61
Table 13 – Notify/response of DestinationList information with nodeManagementDestinationListData	66
Table 14 – Binding request with nodeManagementBindingRequestCall	68
Table 15 – nodeManagementBindingData holds list of binding entries.....	71
Table 16 – Remove binding with nodeManagementBindingDeleteCall	73
Table 17 – Subscription request with nodeManagementSubscriptionRequestCall	77
Table 18 – nodeManagementSubscriptionData holds list of subscription entries	79
Table 19 – Remove subscription with nodeManagementSubscriptionDeleteCall.....	81
Table 20 – nodeManagementUseCaseData	84
Table 21 – SHIP parameters default values	87
Table 22 – Mandatory parameters in the TXT record.....	93
Table 23 – Optional parameters in the TXT record.....	93
Table 24 – Mapping from the XSD types to JSON types.....	103
Table 25 – Transformation of a simple type	104
Table 26 – Mapping from the XSD compositors to JSON types.....	104
Table 27 – Examples for XML and JSON representations	106
Table 28 – Example transformation of several combined XSD item types	108
Table 29 – Example for JSON to XML transformation	110
Table 30 – Trust levels	123
Table 31 – MessageType values	134
Table 32 – Structure of SmeHelloValue of SME "hello" message.....	137
Table 33 – Structure of SmeProtocolHandshakeValue of SME "Protocol Handshake" message.....	145

Table 34 – Structure of SmeProtocolHandshakeErrorValue of SME "Protocol Handshake Error" message	146
Table 35 – Values of Sub-element "error" of messageProtocolHandshakeError	148
Table 36 – Structure of SmeConnectionPinStateValue of SME "PIN state" message	150
Table 37 – Structure of SmeConnectionPinInputValue of SME "pin input" message	151
Table 38 – Structure of SmeConnectionPinErrorValue of SME "Pin error" message	151
Table 39 – Values of Sub-element "error" of connectionPinError	157
Table 40 – Structure of MessageValue of "data" message	159
Table 41 – Structure of SmeConnectionAccessMethodsRequestValue of SME "Access methods request" message	162
Table 42 – Structure of SmeConnectionAccessMethodsValue of SME "Access methods" message	162
Table 43 – Structure of SmeConnectionCommissioningRequestValue of SME "commissioning request" message	164
Table 44 – Structure of SmeConnectionCommissioningResponseValue of SME "commissioning response" message	165
Table 45 – Structure of SmeConnectionKeyMaterialRequestValue of SME "key material request" message	165
Table 46 – Structure of SmeConnectionKeyMaterialValue of SME "key material" message	166
Table 47 – Structure of SmeConnectionKeyMaterialResponseValue of SME "key material response" message	167
Table 48 – Structure of SmeConnectionKeyMaterialDeleteValue of SME "key material delete" message	168
Table 49 – Structure of SmeConnectionKeyMaterialDeleteResponseValue of SME "key material delete response" message	169
Table 50 – Structure of SmeConnectionKeyMaterialStateValue of SME "key material state" message	170
Table 51 – Structure of SmeConnectionKeyMaterialStateResponseValue of SME "key material state response" message	170
Table 52 – Structure of SmeConnectionKeyMaterialStateRequestValue of SME "key material state request" message	171
Table 53 – Structure of SmeCloseValue of SME "close" message	172
Table 54 – Well-known values for the element "protocolId"	173

INTERNATIONAL ELECTROTECHNICAL COMMISSION

**Standard interface for connecting charging
stations to local energy management systems -
Part 3: Communication protocol and cybersecurity specific aspects**

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) IEC draws attention to the possibility that the implementation of this document may involve the use of (a) patent(s). IEC takes no position concerning the evidence, validity or applicability of any claimed patent rights in respect thereof. As of the date of publication of this document, IEC had not received notice of (a) patent(s), which may be required to implement this document. However, implementers are cautioned that this may not represent the latest information, which may be obtained from the patent database available at <https://patents.iec.ch>. IEC shall not be held responsible for identifying any or all such patent rights.

IEC 63380-3 has been prepared by IEC technical committee 69: Electrical power/energy transfer systems for electrically propelled road vehicles and industrial trucks. It is an International Standard.

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

Draft	Report on voting
69/1051/FDIS	69/1060/RVD

Full information on the voting for its approval can be found in the report on voting indicated in the above table.

The language used for the development of this International Standard is English.

This document was drafted in accordance with ISO/IEC Directives, Part 2, and developed in accordance with ISO/IEC Directives, Part 1 and ISO/IEC Directives, IEC Supplement, available at www.iec.ch/members_experts/refdocs. The main document types developed by IEC are described in greater detail at www.iec.ch/publications.

In this document, all text record fields are written in lowercase Courier font, since they belong to protocol information/binary data exchange.

A list of all parts in the IEC 63380 series, published under the general title *Standard interface for connecting charging stations to local energy management systems*, can be found on the IEC website.

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

- reconfirmed,
- withdrawn, or
- revised.

INTRODUCTION

The expansion of renewable energy and the simultaneous reduction in conventional generation of electricity result in new power flows and loads on the equipment in the grid and at the house connection point. At the same time, electrical consumers with high power consumption are increasingly being installed in low-voltage systems in private customer systems. These include charging systems for electric vehicles and heat pumps. These two developments can temporarily lead to peak loads and bottlenecks in the network. An expansion of the distribution grids for the comparatively few hours of high simultaneous power consumption is not considered economically sensible. The legislator of energy efficiency has therefore introduced the concept of "network-friendly control of controllable consumer devices".

It is crucial to define a standardized interface for the connected consumers and generating facilities, which also includes the charging infrastructure for electric vehicles. When developing a local, standardized interface, it is important to make a fundamental distinction between the terms "power management" and "energy management".

In order to avoid an overload and the associated emergency shutdown due to specified power limits in the property while all consumers are drawing electricity at the same time – especially heating and air conditioning technology as well as charging infrastructure –, power management is of great urgency. The maximum load at the grid connection point can therefore be reduced. Accordingly, it is important to give priority to local power management over, for example, optimization of operations and tariffs or desired charging plans.

Furthermore, the tariff-optimized operation can be pursued within the limits specified by the grid infrastructure – controlled by the energy management system. As a consequence, a charging infrastructure will be able to transmit information about procurement and tariff-optimized operation from the local energy management of the property to the electric vehicle so that it can coordinate its charging plan according to local demands. Effective coordination becomes essential if generating systems are used within the property in order to achieve the highest possible self-consumption of electricity.

The long-term goal is to buffer power and energy bottlenecks within a property using the energy stored in the vehicle, which also brings the topic of energy recovery into focus; this aspect needs to be considered during the development of a standardized interface for local power and energy management.

The aim of the IEC 63380 series is to define a standard interface for connecting charging stations to local energy management systems and the information exchange.

The IEC 63380 series specifies use cases, the sequences of information exchange, the data models as well as the communication protocols to be used and includes all aspects of local energy management of charging stations.

The IEC 63380 series covers scenarios where the charging infrastructure is managed by the entity that operates the private electrical network, and local energy management systems are used for local load management.

The IEC 63380 series addresses the energy management in installations with forward and bidirectional charging whereby the overall energy management is ensured by the customer energy manager.

The IEC 63380 series does not cover the secure information exchange between the charging station and the IT backend system(s), such as the management of energy transfer of the charge session, contractual and billing data, provided by the IT backend.

The IEC 63380 series consists of the following structure, describing the interface between charging stations and local energy management systems;

- IEC 63380-1¹: General requirements, use cases and abstract messages;
- IEC 63380-2: Specific data model mapping;
- IEC 63380-3: Communication protocol and cybersecurity specific aspects;
- IEC 63380-4²: Test specifications.

1 Under preparation: Stage at the time of publication: IEC/CFDIS 63380-1:2025.

2 Under preparation. Stage at the time of publication: IEC/ACD 63380-4:2021.

1 Scope

This part of IEC 63380 defines the secure information exchange between local energy management systems and electric vehicle charging stations. The local energy management systems communicate to the charging station controllers via the resource manager.

This document specifies the application of relevant transport protocols; in this case, SPINE (smart premises interoperable neutral-message exchange), SHIP (smart home IP), and ECHONET Lite. Other communication protocols can be defined in future editions.

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 62394, *Service diagnostic interface for consumer electronics products and networks – Implementation for ECHONET*

IEC 63380-2, *Standard interface for connecting charging stations to local energy management systems – Part 2: Specific data model mapping*

ISO/IEC 14543-4-3:2015, *Information technology, Home Electronic Systems (HES) architecture – Part 4-3: Application layer interface to lower communications layers for network enhanced control devices of HES Class 1*

IETF RFC 793:1981, *Transmission Control Protocol*

IETF RFC 3280:2002, *Internet X.509 Public Key Infrastructure Certificate and Certificate Revocation List (CRL) Profile*

IETF RFC 6455:2011, *The WebSocket Protocol*

IETF RFC 6763, *DNS-Based Service Discovery*

IETF RFC 5246, *The Transport Layer Security (TLS) Protocol Version 1.2*

IETF RFC 5289, *TLS Elliptic Curve Cipher Suites with SHA-256/384 and AES Galois Counter Mode (GCM)*

IETF RFC 8422, *Elliptic Curve Cryptography (ECC) Cipher Suites for Transport Layer Security (TLS) Versions 1.2 and earlier*

3 Terms, definitions, and abbreviated terms

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

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

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

3.1 Terms and definitions

3.1.1

CA

certificate authority

certification authority

entity which can provide a digital signature for certificates

Note 1 to entry: Other SHIP nodes can check this digital signature with the certificate from the CA itself, the "CA-certificate".

3.1.2

commissioning tool

<SHIP> instrument to establish the trust between different devices in the smart home installation, for example distribute trustworthy credentials from some SHIP nodes to other SHIP nodes

Note 1 to entry: For example, a smart phone, a Web server or a dedicated device can embody the role of a commissioning tool. So far, the SHIP specification does not specify a commissioning tool; an interoperable protocol for commissioning can be used on the layer above SHIP.

Note 2 to entry: Manufacturer can also use their own solutions.

3.1.3

DNS host name

fully qualified domain name used within DNS as host name to get the IP address of the corresponding internet host

3.1.4

factory default

setting that allows the user to reset the SHIP node to the as-new condition

Note 1 to entry: All data that has been provided and stored by the SHIP node during its operation time shall be deleted.

3.1.5

mDNS

multicast DNS host name

fully qualified domain name used within mDNS as host name to get the IP address of the corresponding local SHIP node

3.1.6

M/O/NV/C

abbreviations which refer to

- 1) M = mandatory,
- 2) O = optional,
- 3) NV = not valid, and
- 4) C = choice, i.e., a presence or support depends also on the selection from multiple possibilities,

and which are primarily used within specific definition tables describing certain specialized data model definitions

3.1.7

PIN

personal identification number

secret number used for SHIP specific verification procedures

3.1.8

push button

switching mechanism to control some aspect of a machine or a process

Note 1 to entry: A push button event does not necessarily mean that a real physical button is used to trigger this event. A push button event can also be generated by other means, for example, via a smart phone application or a Web-interface (secure connection to SHIP node required). A push button shall provide a simple mechanism for a user to bring the device to a certain state or start a certain process.

3.1.9

QR code™

quick response code

code used for efficient encoding of data into a small graphic

Note 1 to entry: Among other standards, ISO/IEC 18004:2024 specifies the encoding of QR code symbols.

Note 2 to entry: QR code is the trademark of a product supplied by Denso Wave incorporated. This information is given for the convenience of users of this document and does not constitute an endorsement by IEC of the product named. Equivalent products can be used if they can be shown to lead to the same results.

3.1.10

SHIP ID

identification which is used to uniquely identify a SHIP node, for example, in its service discovery, and which is present in the mDNS or DNS-SD local service discovery

Note 1 to entry: Each SHIP node has a globally unique SHIP ID.

Note 2 to entry: See 6.4.

3.1.11

SHIP client

role which is assigned to the SHIP node that also embodies the TCP client role for a specific peer-to-peer connection

3.1.12

SHIP commissioning

SHIP term which denotes the distribution of trustworthy SKIs from certain SHIP nodes to other SHIP nodes

Note 1 to entry: The distribution of the trustworthy SKIs is handled by a so-called SHIP commissioning tool.

3.1.13**SHIP commissioning tool**

instrument which is used to distribute trustworthy SKIs from certain SHIP nodes to other SHIP nodes, and which allows a user to handle the trust relationships in the whole SHIP installation (if each node in the installation supports commissioning) over one simple user interface

3.1.14**SHIP node**

logical device which communicates via the described SHIP protocol and can be integrated into a Web server or physical device

Note 1 to entry: One physical device can have more than one logical SHIP node. In this case, each SHIP node shall use distinct ports (e.g. a physical device provides 3 open ports with 3 different SHIP services).

3.1.15**SHIP server**

role which shall be assigned to the SHIP node that also embodies the TCP server role for a specific peer-to-peer connection

3.1.16**SKI****subject key identifier**

identifying certificate that contains a specific public key and is used as a cryptographically backed identification and authentication criterion

Note 1 to entry: Each SHIP node has a specific public key.

3.1.17**trusted SHIP node**

node which has a trusting relationship to another ship node

Note 1 to entry: If SHIP node A has a communication partner and a trusted relationship to SHIP node B, SHIP node B is called a trusted SHIP node from SHIP node A's point of view; a trusted relationship can be established in different ways.

3.1.18**UTF****UCS transformation format**

computing industry standard for the consistent encoding, representation, and handling of text expressed in most of the world's writing systems

3.1.19**Web server-based SHIP node**

SHIP node that is hosted by a Web server

3.1.20**wireless computed network**

IP network based on the IEEE 802.11TM set of standards, used for wireless IP communication

3.2 Abbreviated terms

ABNF	augmented Backus-Naur form
CA	certificate authority
CMI	connection mode initialisation
DEV	device
DNS	domain name system
DNS-SD	domain name system-service discovery
ECC	elliptic curve cryptography
ECDH	elliptic-curve Diffie-Hellman
ECDHE	elliptic curve Diffie-Hellman ephemeral
ECDSA	elliptic curve digital signature algorithm
IANA	Internet assigned numbers authority
ICMP	Internet control message protocol
IETF	Internet engineering task force
IP	Internet protocol
JSON	Java script object notation
mDNS	multicast DNS
MTU	maximum transmission unit
NAT	network address translation
NECD	network enhanced control device
NFC	near field communication
PIN	personal identification number
PKI	public key infrastructure
RFC	request for comments
RFE	restricted function exchange
RFU	reserved for future use
RST	reset
RTO	retransmission timeout
RTT	round-trip times
SHIP	smart home Internet protocol
SKI	subject key identifier
SLAAC	stateless address autoconfiguration
SRV	service record
SME	SHIP message exchange
SPINE	smart premises interoperable neutral message exchange
TLS	transport layer security
TTL	time-to-live
UCS	universal character set
UTF	unicode transformation format
XML	extensible markup language
XSD	XML schema definition

4 Overview

This document specifies the application of relevant communication protocols for the high-level use cases in IEC 63380-1³ and the data models in IEC 63380-2; in this case, SPINE (smart premises interoperable neutral-message exchange), SHIP (smart home IP), and ECHONET Lite. Other communication protocols can be defined in future editions. Figure 1 shows the use of the communication protocols within this document.

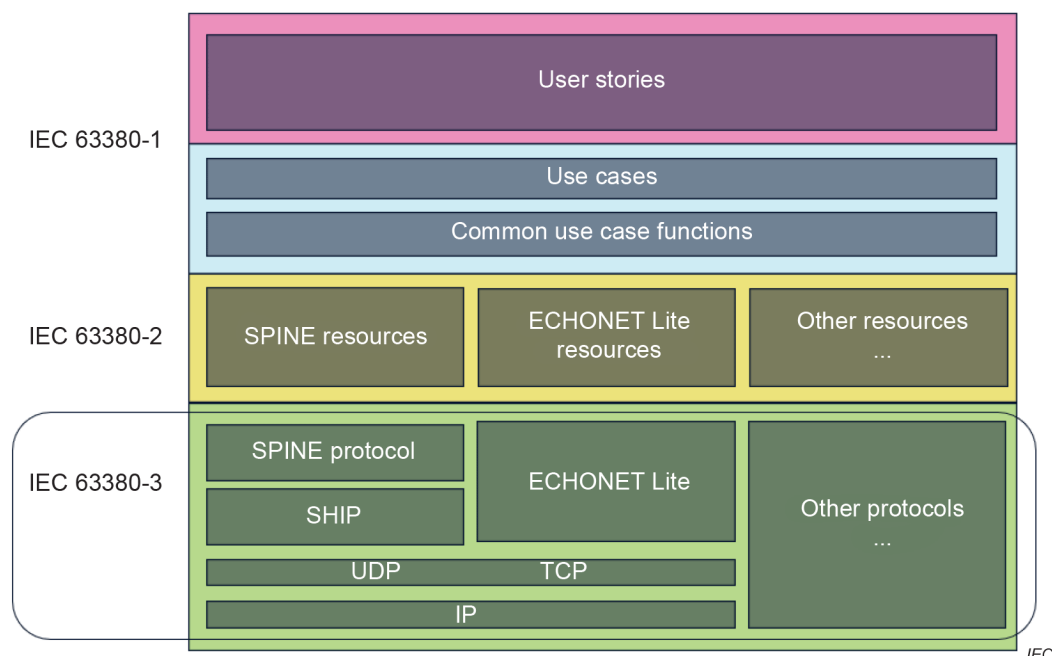


Figure 1 – Overview of communication protocols within IEC 63380-3

5 SPINE protocol

5.1 General

The SPINE protocol serves as neutral message framework to model some typical mechanisms of common communication protocols. This permits to model interactions between SPINE devices and local energy management system.

Clause 5 discusses a general SPINE device architecture including the concept of addressing, a general neutral message format ("SPINE datagram"), and so-called "communication modes". Afterwards, some further mechanisms (like detailed discovery, binding, subscription, etc.) are discussed. This can be seen as "functional commissioning" between SPINE devices, i.e., the management of useful relations between functionalities of SPINE devices.

³ Under preparation: Stage at the time of publication: IEC/CFDIS 63380-1:2025.

5.2 Architecture overview

5.2.1 General rules

In theory, the SPINE data model permits arbitrary combinations of SPINE entities, SPINE features, SPINE classes, etc. However, the definition of interoperable binding (see 5.5.4) and subscription (see 5.5.5) mechanisms requires imposing some restrictions, i.e. the definition of an architecture. Thus, the following rules in 5.2.1 apply.

- 1) A device consists of entities. An entity consists of (sub-)entities or features. For each address level (device level, entity level, feature level), the SPINE protocol defines an addressing scheme.

NOTE 1 This is explained in more detail in 5.2.3.

- 2) Each device shall implement a so-called primary NodeManagement instance with information on itself. The device shall offer this information on its entity 0 (entity of the device with the entity address = 0) at feature 0 (feature of the entity 0 with the feature address = 0). This information should contain information about all entities of the device and all features of these entities. The so-called "entityType" (see 5.5.2) of entity 0 shall be "DeviceInformation". The so-called "featureType" (see 5.5.2) of feature 0 of entity 0 shall be "NodeManagement".
 - 3) This document does not consider any non-primary NodeManagement instances (i.e. NodeManagement instances on different addresses than entity 0 at feature 0 are not considered).
 - 4) Each NodeManagement instance is a feature in general (see rule 2)). Thus, NodeManagement instances include proper information on itself or other NodeManagement instances as well, according to rules 1) to 3).
 - 5) On each feature, there shall be at maximum one class implemented with regard to the features primary functionality, i.e., it is not permitted to offer more than one class on a single feature (see also 5.3.3).
 - 6) A feature on a device is assigned a "functional role". This role is either "server" or "client", or "special". A "functional role" is independent from any "connection role" (i.e. a role typically used in communications technologies like TCP).
 - 7) The functional role "server" is used if the device is the "owner" of data of the corresponding feature (see also 5.3.3.3), i.e., the device can notify changes or send this data as reply upon request. It can also accept "write" operations to perform a data change. In most cases, the feature role as "server" also includes the capability of a device to operate autonomously, i.e., to execute its server feature tasks even if no feature with role "client" sends data to control the server's feature tasks.
 - 8) The functional role "client" is used to receive information provided by a "server" and to use the server's features accordingly
- EXAMPLE Configure the server features using "write" operations, provided this is offered by the server feature).
- 9) The functional role "special" is reserved for specific features. It shall only be used for features explicitly mentioned in official SPINE specifications. In this document, the primary NodeManagement instance shall have the role "special".
 - 10) The concepts for binding and subscription respect the role assignment. Details are given in the corresponding sections.

NOTE 2 The rules only describe the architecture. Whether all or just reduced information of a device's NodeManagement instances is shared with another device and whether the communication between the features of two devices is restricted (or even blocked) or not is discussed separately (especially using the "trust level" concept).

5.2.2 Common data types

5.2.2.1 General

Most data types use data types defined by W3C. In this document, these data types are identified by the namespace prefix "xs:" (e.g. "xs:boolean").

This document defines also some own data types which are based upon W3C data types. In this document, these data types are mentioned without namespace prefix. Details on their definition can be found in 5.2.2.2 to 5.2.2.8.

5.2.2.2 ElementTagType

Used for elements that have no content (empty tag). A complexType, containing no elements.

If used as type for an element, the element shall have no further content in this document.

5.2.2.3 LabelType

A default type used for all "label" elements. A simpleType (restricted string).

The string-length should not be longer than 256 characters. If it is longer, the sender shall consider the possibility that the receiver will shorten the string to 256 characters.

5.2.2.4 DescriptionType

A default type used for all "description" elements. A simpleType (restricted string).

The string-length should not be longer than 4 096 characters. If it is longer, the sender shall consider the possibility that the receiver will shorten the string to 4 096 characters.

5.2.2.5 SpecificationVersionType

The SPINE specification version.

A simpleType (restricted string), with the pattern

```
[1-9][0-9]*\.[0-9]+\.[0-9]+
```

allowing version strings like "1.0.0", "2.5.3", etc. The maximum string length is limited to 128 characters.

5.2.2.6 EnumExtendType

Used for extending enumeration with vendor specific ones. Some enumerations can be extended by the vendors with own values. A simpleType (restricted string), with the pattern

```
_(i:[1-9][0-9]*|n:[a-zA-Z0-9-]+)_[^p{Cc}\p{Cf}\p{Z}]+
```

The maximum string length is limited to 256 characters.

5.2.2.7 RoleType

If a role shall be stated, this type is used. A simpleType (restricted string), containing the enumeration values client, server or special.

5.2.2.8 FeatureGroupType

Used for connecting related features. A simpleType (restricted string), with the pattern

```
(#[1-9][0-9]*)*
```

allowing feature group strings like "#1", "#1#5", etc. The maximum string length is limited to 128 characters.

5.2.2.9 DeviceTypeType

Union of

- DeviceTypeEnumType (see 5.2.2.10), and
- EnumExtendType (see 5.2.2.6).

5.2.2.10 DeviceTypeEnumType

The type of the device.

5.2.2.11 EntityTypeType

Union of

- EntityTypeEnumType (see 5.2.2.12), and
- EnumExtendType (see 5.2.2.6).

5.2.2.12 EntityTypeEnumType

The type of the entity.

5.2.2.13 FeatureTypeType

Union of

- FeatureTypeEnumType (see 5.2.2.14), and
- EnumExtendType (see 5.2.2.6).

5.2.2.14 FeatureTypeEnumType

The type of the feature.

5.2.2.15 FunctionType

Union of

- FunctionEnumType (see 5.2.2.16), and
- EnumExtendType (see 5.2.2.6).

5.2.2.16 FunctionEnumType

A list of all SPINE functions.

5.2.2.17 PossibleOperationsClassifierType

If the element partial is used, filtering the further content is permitted. Otherwise, only full operations (no filtering) are allowed. A complexType, containing the element partial (type: ElementTagType, see 5.2.2.2).

5.2.2.18 PossibleOperationsReadType

If read operations are permitted, this element is set. Elements from the base type can be used. A complexType with the base type PossibleOperationsClassifierType (see 5.2.2.17).

5.2.2.19 PossibleOperationsWriteType

If write operations are permitted, this element is set. Elements from the base type can be used. A complexType with the base type PossibleOperationsClassifierType (see 5.2.2.17).

5.2.2.20 PossibleOperationsType

Defines which operations on a function are allowed as shown in Figure 2. A complexType, containing the elements read (type: PossibleOperationsReadType, see 5.2.2.18) and write (type: PossibleOperationsWriteType, see 5.2.2.19).

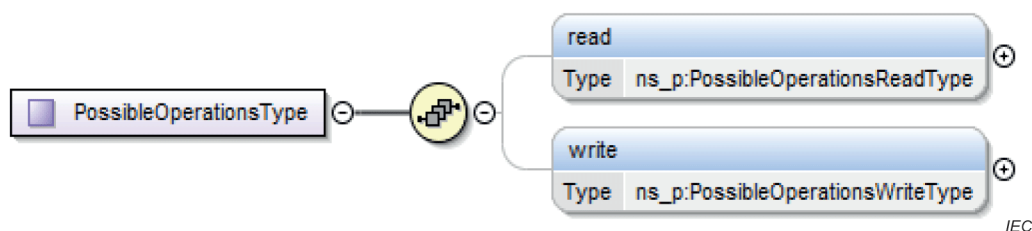


Figure 2 – PossibleOperationsType

5.2.2.21 MaxResponseDelayType

Used to represent the maxResponseDelay. A simpleType (xs:duration).

5.2.2.22 AbsoluteOrRelativeTimeType

The commonly used type for time information in the SPINE data model. Allows the modelling of absolute (xs:dateTime) or relative (xs:duration) times. A union with the two members (allowed types) xs:duration and xs:dateTime.

xs:dateTime is used to model absolute times.

xs:duration is used to model relative times. If not stated otherwise, a duration of 0 s (e.g. "PT0S", or "PT0H", etc.) equals "now", while negative durations denote times in the past and positive values denote times in the future relative to "now".

5.2.2.23 AddressDeviceType

Used to address a (physical) device. A simpleType (restricted string), containing the device address part, with the pattern

```
d:_ (i:[1-9][0-9]*|n:[a-zA-Z0-9-]+)_ [^\p{Cc}\p{Cf}\p{Z}]+
```

5.2.2.24 AddressEntityType

Used to address a logical device. A simpleType (unsignedInt), containing an entity address part.

5.2.2.25 AddressFeatureType

Used to address a specific feature. A simpleType (unsignedInt), containing the feature address part.

5.2.2.26 DeviceAddressType

Address type including only the device part, as shown in Figure 3. A complexType with the element device (type: AddressDeviceType, see 5.2.2.23).

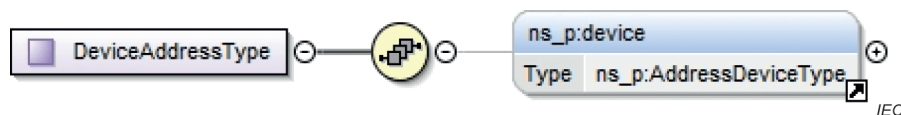


Figure 3 – DeviceAddressType

5.2.2.27 EntityAddressType

Address type including the device part and a list of entities, as shown in Figure 4. A complexType with the base type DeviceAddressType (see 5.2.2.26) and a further list of element entity (type: AddressEntityType, see 5.2.2.24).

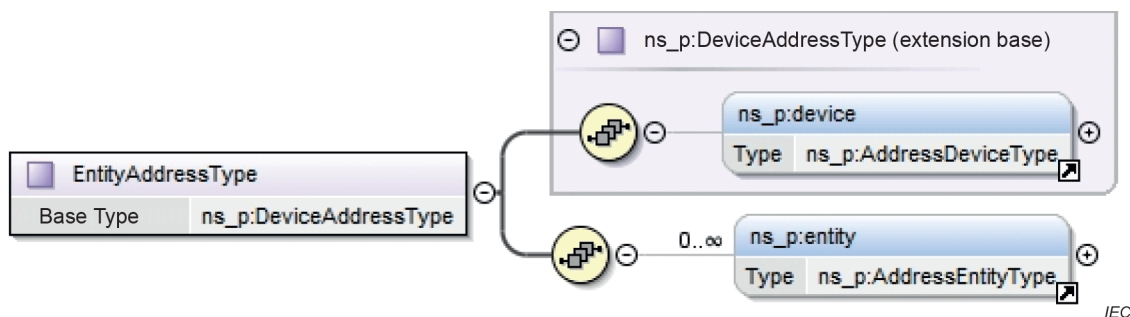


Figure 4 – EntityAddressType

5.2.2.28 FeatureAddressType

Address type including the device part, a list of entities and the feature part, as shown in Figure 5. A complexType with the base type EntityAddressType (see 5.2.2.27) and a further element feature (type: AddressFeatureType, see 5.2.2.25).

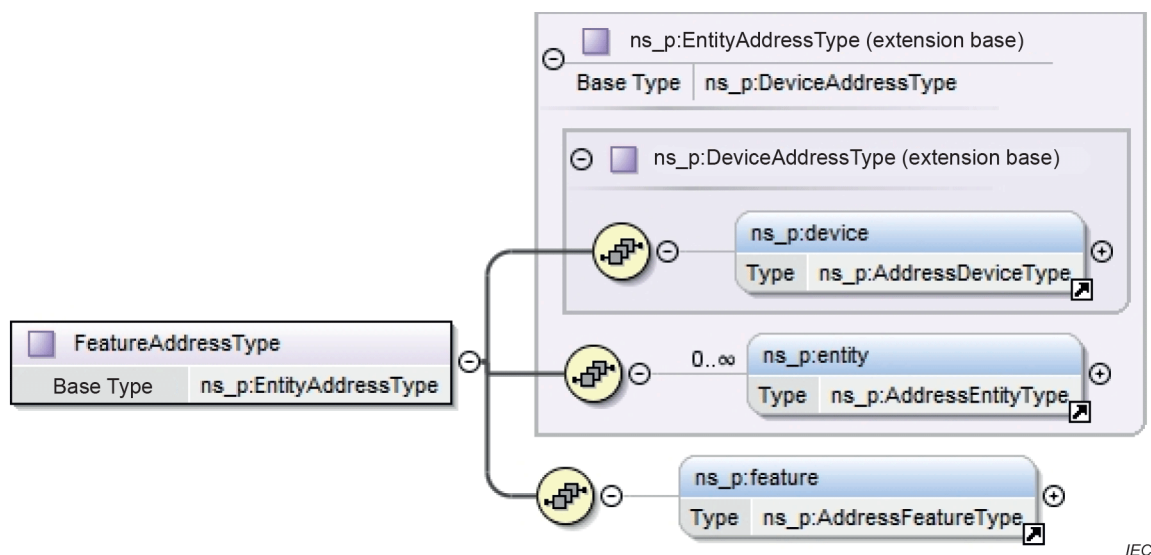


Figure 5 – FeatureAddressType

5.2.3 Address level details

The concept of the miscellaneous address levels as described in 5.2.1.

The following example defines a device with name "someDevice" with three top-level entities (entities directly below the device "someDevice") with entity addresses 0, 1, 4 ("entity 0" describes a SPINE entity with the entity address = 0, feature 0 describes a SPINE feature with the feature address = 0):

```
"someDevice"
  +--- entity 0          (child of "someDevice")
  |      +--- feature 0
  +--- entity 1
  |      +--- entity 4 (child of "someDevice"/entity 1)
  |      |      +--- feature 7 (*1)
  |      +--- entity 5 (child of "someDevice"/entity 1)
  |      |      +--- feature 1 (*2)
  |      |      +--- feature 7 (*2)
  |      +--- feature 1 (child of "someDevice"/entity 1)
  |      +--- feature 4 (child of "someDevice"/entity 1)
  +--- entity 4          (child of "someDevice")
  |      +--- feature 1 (child of "someDevice"/entity 4)
(*1): child of "someDevice"/entity 1/entity 4
(*2): child of "someDevice"/entity 1/entity 5
```

In this example, the top-level entity 1 contains two features (1, 4), but also two other (sub-)entities. This shows that entities can be nested.

Starting at the top, the full address paths are as follows:

NOTE 1 This is an example to describe the SPINE address level concept and that the following lines do not state valid SPINE addresses; the correct usage of SPINE addresses is described in 5.3.

- 1) device "someDevice";
- 2) device "someDevice" / entity 0;
- 3) device "someDevice" / entity 0 / feature 0;
- 4) device "someDevice" / entity 1;
- 5) device "someDevice" / entity 1 / entity 4;
- 6) device "someDevice" / entity 1 / entity 4 / feature 7;
- 7) device "someDevice" / entity 1 / entity 5;
- 8) device "someDevice" / entity 1 / entity 5 / feature 1;
- 9) device "someDevice" / entity 1 / entity 5 / feature 7;
- 10) device "someDevice" / entity 1 / feature 1;
- 11) device "someDevice" / entity 1 / feature 4;
- 12) device "someDevice" / entity 4;
- 13) device "someDevice" / entity 4 / feature 1.

Only the full address path is unique, i.e. feature 7 of { device "someDevice" / entity 1 / entity 4 } differs to feature 7 of { device "someDevice" / entity 1 / entity 5 }. Similarly, entity 4 of { device "someDevice" / entity 1 } differs to entity 4 of { device "someDevice" }.

The following explains how this is modelled in the SPINE XSDs and XMLs. In the XSDs, an element "entity" is often defined with attributes

```
minOccurs="0" maxOccurs="unbounded"
```

The upper bound of this cardinality means an "entity" tag can occur arbitrarily often at the given position in an XML (representing arbitrary depth of nested entities). However, a device that complies with this and previous versions of the SPINE protocol only can silently discard messages where an entity list comprises more than 15 "entity" items. This means an implementation should avoid implementing any of its entities with more than 15 "entity" items as it is likely that a communication partner will ignore such entities. The lower bound "0" of this cardinality means there can be no "entity" element at all. The architecture itself always requires the presence of at least one entity. But certain message definitions that contain address elements with this generic address type can well permit the absence of "entity" elements for specific purposes (among others, the deletion of all so-called "bindings" between two devices omits all "entity" elements in a specific address element).

The i-th occurrence of "entity" within an address instance corresponds to the i-th entity level (depth). The entity level "i+1" is a child of entity level "i". In other terms, an XML part with full address path for { device "someDevice" / entity 1 / entity 4 / feature 7 } is represented as follows:

```
<device>someDevice</device>
<entity>1</entity>
<entity>4</entity>
<feature>7</feature>
```

Within the message description tables, the possibility of multiple "entity" tags (representing nested entities) is specified as follows:

... entity (list)

As explained, multiple entity tags in an address instance always describe "nested entities", i.e. entities as child elements of other entities. Such multiple entity tags do not describe "parallel" entities (entities on the same hierarchical level). One address instance can only describe one single address path and is not used to describe several parallel address paths of a device.

With regard to the so-called "restricted function exchange" (see 5.3.3.4), a SPINE address as information unit shall always be seen as an indivisible information unit that is transmitted completely (with the exception that the "device" part can be omitted in some cases). This means that a SPINE address with a list of entity tags does neither match the definition of "identifiable list entries" nor the definition of "non-identifiable list entries" of 5.3.3.4.1.

NOTE 2 A SPINE address can also be used for other purposes than an information unit – see 5.3.3.4.7.3 for its use as so-called selectors.

5.3 SPINE datagram

5.3.1 Overview

5.3.1.1 General

The ISO-OSI layer model defines an application layer, among others. Interaction with another application is considered an end-to-end connection. With regard to the SPINE data model, the "functions" can be exchanged on this level.

In order to model the exchange between two end points dynamically, the SPINE data model defines an element "datagram" consisting of "header" and "payload", as shown in Figure 6.

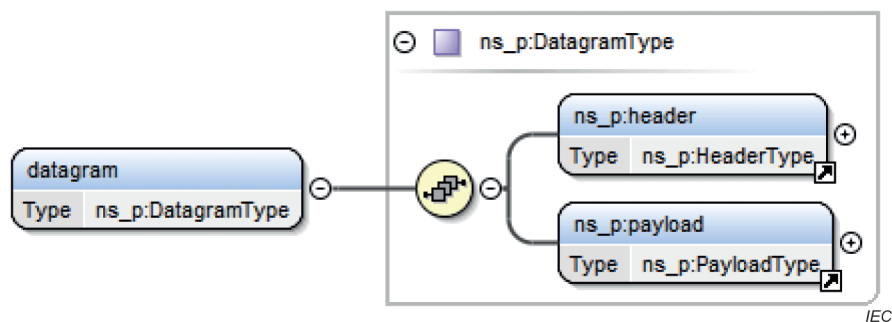


Figure 6 – SPINE datagram

5.3.1.2 Structure

The structure of the SPINE datagram shall be set as shown in Table 1.

Table 1 – Structure of the SPINE datagram

Element name	M/O/NV/C	Brief explanation
datagram	M	The root element of the SPINE datagram shall always be present.
datagram. header	M	The header element shall be present. For sub-elements of the header, see 5.3.2.7.
datagram. payload	M	The payload element shall be present. For sub-elements of the payload, see 5.3.3.2.

The notation "a.b" (for example "datagram.header") is used here to show the hierarchical structure of a SPINE datagram. It is used to define that "b" is a child element of "a".

5.3.2 Header

5.3.2.1 General

The SPINE header contains information on the version of the applied data model, addresses of end points, etc., as described in Figure 7. This technical solution defines the SPINE protocol version 1.3.0.

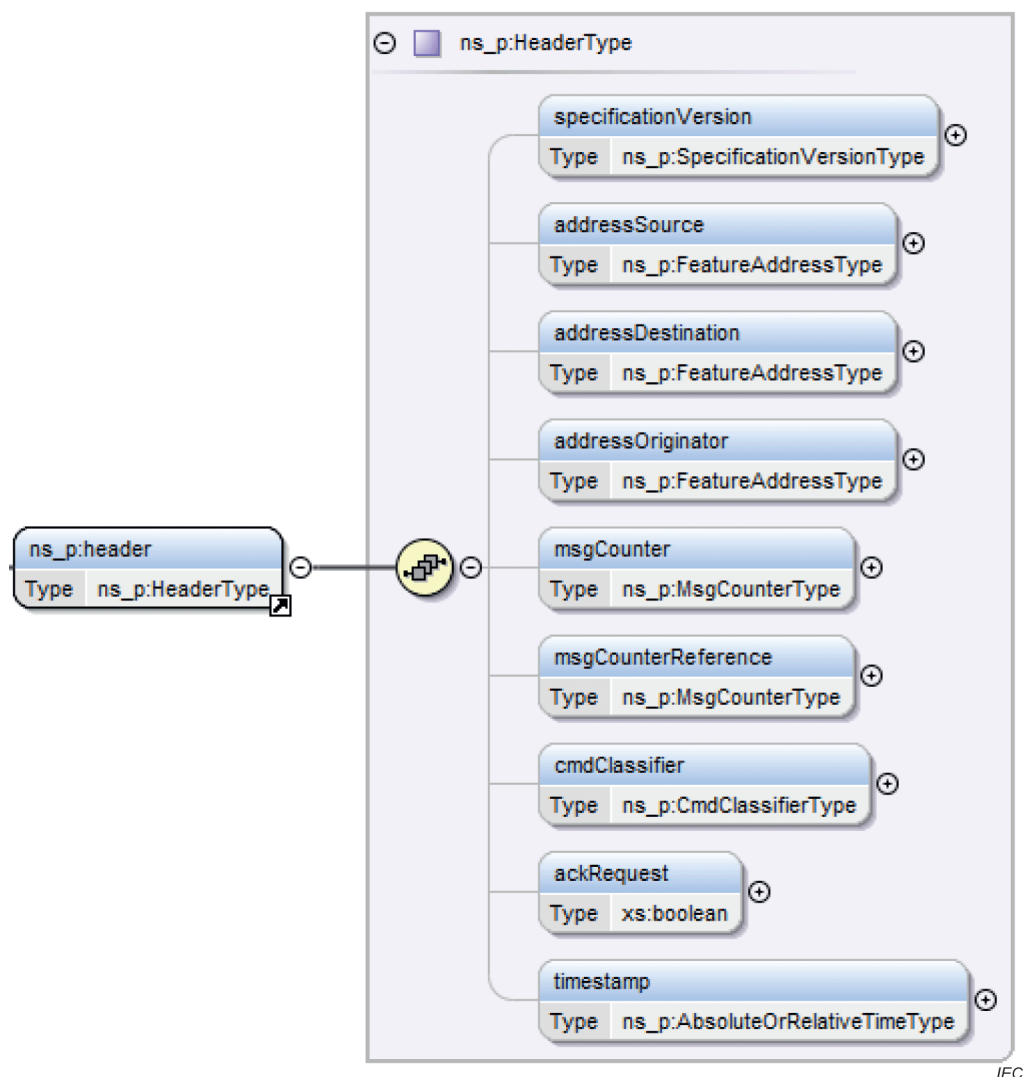


Figure 7 – SPINE header

5.3.2.2 Address information

5.3.2.2.1 addressSource and addressDestination

5.3.2.2.1.1 General

The elements `addressSource` and `addressDestination` are defined with their child elements and purpose in Table 3. `addressSource` corresponds to the unique address path (see 5.2.3) of a feature where a SPINE message was created. `addressDestination` corresponds to the unique address path of the receiving feature. Subsequently, some rules on the child elements of `addressSource` and `addressDestination` are given.

If a device creates a message, it shall set its entity and feature address parts into `addressSource`. The recipient's entity and feature address parts shall be set in `addressDestination`.

The "device" address parts of addressSource and addressDestination can be omitted in some cases. When to set the "device" address parts is defined subsequently. For this, we consider a communication between the SPINE devices "A" and "B". The "common rules" (see 5.3.2.2.1.2) apply in any case. Furthermore, additional rules apply dependent on the communication mode.

5.3.2.2.1.2 Common rules on the use of "device" address elements

In general, the use of "addressSource. device" and "addressDestination. device" requires ensuring the uniqueness of all "device" values, i.e. device "A"'s value of "device" shall be different to device "B"'s value of "device". This is also in accordance with 5.5.2.2.5.2.

We assume a message is transmitted from device "A" to device "B".

- 1) If "addressSource. device" is present and not empty, it shall be identical to device "A"'s own value of "device". The message shall be considered illegal if the value differs.
- 2) If "addressDestination. device" is present and not empty, it shall be identical to device "B"'s own value of "device". The message shall be considered illegal if the value differs.

5.3.2.2.1.3 Additional rules in case of "simple communication mode"

The rules in 5.3.2.2.1.3 apply in case of a "simple communication mode" (see 5.4.2). In this mode, two devices are considered being directly connected to each other. We assume a message is transmitted from device "A" to device "B".

Within the header, the "device" elements of addressSource and addressDestination should be set to the correct value. The absence of "device" in "addressSource" shall be treated as if "device" in "addressSource" was set to device "A"'s own value of "device" (i.e. the "device" address part of device "A"). The absence of "device" in "addressDestination" shall be treated as if "device" in "addressDestination" was set to device "B"'s own value of "device" (i.e. the "device" address part of device "B").

5.3.2.2.1.4 Additional rules in case of "enhanced communication mode"

The rules in 5.3.2.2.1.4 apply in case of an "enhanced communication mode" (see 5.4.3). In this mode, both "addressSource. device" and "addressDestination. device" shall always be set in the header.

5.3.2.2.2 addressOriginator

As explained in Table 3, the element "addressOriginator" can be used to model information of a "forwarded" XML and permits preserving the address of the original submitter. The use of addressOriginator is optional. However, a brief example shall demonstrate a typical use of this element.

We assume device "A" created a message "X" and submitted it to device "B". We also assume device "B" serves as some kind of "data warehouse" for all messages it received from miscellaneous devices. With its "data warehouse" service, it can provide an overview of messages from all its connected devices. We also assume these messages are available on one or more features of device "B". If a device "C" reads on such a feature of device "B", device "B" shall set its own feature address into "addressSource" of its response to device "C". In order to keep the information where the functional content of the message originally stemmed from (device "A"), the element "addressOriginator" should be set properly.

If "addressOriginator" is used as described, the elements "addressSource. device", "addressDestination. device" and "addressOriginator. device" shall be set.

5.3.2.3 Message counter

5.3.2.3.1 General

A message counter (`msgCounter`, `msgCounterReference`) serves for the identification of a message. This is especially important if replies are delivered in a different order than the corresponding requests.

5.3.2.3.2 `msgCounter`

If a device creates a message, it shall assign `msgCounter` a value that does not conflict with any other of its recently created messages (i.e. it shall be a virtually unique value). In general, the value shall not collide with any of the device's previously created messages where the device still expects proper responses. The `msgCounter` value shall be ascending and restart from 0 once the largest possible number ($2^{64} - 1$) was used. The `msgCounter` values may skip some numbers (e.g. after a message "X" with `msgCounter` "10", a message "Y" with `msgCounter` "15" is sent).

If a SPINE device "A" receives a message "X" from SPINE device "B" with a `msgCounter` less or equal than the last `msgCounter` received from device "B", "A" shall process the message "X" as usual. Afterwards, device "A" shall use the unexpectedly low `msgCounter` value as the last `msgCounter` received from device "B". If device "A" receives a message with unexpectedly low `msgCounter` value from device "B", it may report this to the user in order to report that the communication partner can have a problem or unexpected condition (e.g. factory reset); however, such a report is only useful if the underlying communications technology preserves the order of messages.

A best practice to keep `msgCounter` values unique even in case of power failures is as follows: a device keeps a "stored `msgCounter` value" in a non-volatile memory. When a device is turned on, it first increases the "stored `msgCounter` value" by 1 000 and uses the result for its next message. From then on, every 1 000th created message, the device copies the `msgCounter` value into the "stored `msgCounter` value".

A device may assign its messages in the `msgCounter` field values that are unique across all its communication partners. But this kind of uniqueness is not required by this document. Rather, this kind of uniqueness can be easier for implementation in embedded devices whereas Web servers can scale better with values that are communication partner specific.

5.3.2.3.3 `msgCounterReference`

If the created message serves as reply or comparable reaction to a previously received message, the device shall take the value of `msgCounter` from the received message and set it into `msgCounterReference` of its own message.

The element `msgCounterReference` shall not be set if the sent-out message does not relate to a received message.

5.3.2.4 Message classifiers

The element `cmdClassifier` conveys information on the kind of operation associated with the given message "M".

5.3.2.4 uses also the terms "dedicated reply", "dedicated data", or "dedicated response". The SPINE protocol defines a number of (sub-)classes with (SPINE) functions. A device "A" can be the "owner" of a specific function instance (i.e. it has the role "server" at the function's feature address). Another device "B" can request to get a (full or specifically curtailed) copy of this (SPINE) function. A response is called "dedicated" if this request can be fulfilled, i.e. the reply conveys indeed a proper copy of this (SPINE) function.

Table 2 shows permitted values for cmdClassifier and the relation to the kind of message "M". The receipt of a message "M" can lead to the return of a related "acknowledgement message" (see 5.3.2.5), denoted as "N". The scope of "N" is also shown in Table 2.

Table 2 – cmdClassifier values and kind of messages for a message "M" and scope of related acknowledgement messages

cmdClassifier of message "M"	Kind of message "M"	Scope of related acknowledgement message "N" (see 5.3.2.5.2)
Read	Initial	Application error
Write	Initial	Application success or error
Call	Initial	Application success or error
Reply	Response	Transmission success or error
Notify	Response	Transmission success or error
Result	Acknowledgement	Not applicable

We assume device "A" sends an "initial message" to device "B" (address level details like "feature" are just left out to simplify the explanation). Dependent on the received "initial message", device "B" creates a "response message" for device "A", i.e., a "response message" is always related to an "initial message". Furthermore, "response messages" are only used in case the received "initial message" was processed successfully by the recipient.

An "acknowledgement message" can be used to indicate whether an "initial message" or "response message" was processed or transmitted successfully or not (see 5.3.2.5.2).

The different values of cmdClassifier are used for the following operations.

- **Read:** This denotes a "read operation" from the sender (addressSource) to the recipient (addressDestination). The recipient is considered the owner of the proper information (i.e. function). The recipient shall respond with a dedicated "reply" if the read operation is valid and can be processed regularly, i.e., without any failures. Otherwise, the recipient shall respond with an "acknowledgement message" (see 5.3.2.5.2) where "errorNumber" is set to a different value than "0". The obligation to create a "reply" or "error indication" is independent from the value of the element "ackRequest" of the header. More precisely, the acknowledgement request (see 5.3.2.5.1) of a received "read operation" shall not be evaluated.
- **Reply:** This denotes a "reply operation" from the owner of the (assumed) proper information (addressSource) to the recipient (addressDestination). A reply shall be created according to the rules given for cmdClassifier value "read". It shall not be used in any other cases. The recipient of a "reply operation" shall evaluate the acknowledgement request of the message according to 5.3.2.5.1.

NOTE 1 A proper acknowledgement message just denotes a transmission success or error of the "reply operation".

- **Notify:** This denotes a notification ("notify operation") from the owner of the proper information (addressSource) to the recipient (addressDestination). In contrast to a dedicated "reply" upon a "read" message, a "notify" message is created autonomously by the owner of the proper information, i.e. without the need of a received "read" message. A typical example for the creation of a "notify" message is the notification of a value change in a feature that is subscribed by the recipient. The recipient of a "notify operation" shall evaluate the acknowledgement request of the message according to 5.3.2.5.1.

NOTE 2 A proper acknowledgement message just denotes a transmission success or error of the "notify operation".

- **Write:** This denotes a replacement or modification instruction ("write operation", "full" or "restricted") from the sender (addressSource) to the recipient (addressDestination). The recipient is considered the owner of the information (SPINE class function) that shall be replaced or modified according to the instruction. The recipient of a "write operation" shall evaluate the acknowledgement request of the message according to 5.3.2.5.1.
- **Call:** This denotes an instruction ("call operation") from the sender (addressSource) to the recipient (addressDestination) to trigger a specific action at the recipient. A "call operation" can be used to exchange information where the "ownership concept" of the other classifiers usually does not apply. The recipient of a "call operation" shall evaluate the acknowledgement request of the message according to 5.3.2.5.1.
- **Result:** This message classifier is used for so-called "application acknowledgement messages" (also called "result messages"; see 5.3.2.5.1) to indicate the general success or error with regard to a transmitted message. The recipient of a "result message" shall not evaluate the acknowledgement request of the received "result message". A "result message" shall not be created as any kind of response to a received "result message".

5.3.2.5 Acknowledgement concept

5.3.2.5.1 Acknowledgement request

A message "M" denotes the kind of its "acknowledgement request" with the element "datagram. header. ackRequest" (see Table 3): the value "true" indicates "acknowledgement message is required" whereas the value "false" indicates "acknowledgement message is not required".

The recipient of the message "M" shall submit an acknowledgement message (also referred to as "result message" as described below) (see 5.3.2.5.2) if all the conditions in 5.3.2.5.1 are fulfilled.

- 1) The received message "M" belongs to an operation that requires the evaluation of the acknowledgement request according to 5.3.2.4.
- 2) The received message "M" does not belong to an operation that forbids the evaluation of the acknowledgement request according to 5.3.2.4.
- 3) The received message "M" does not belong to an operation that forbids the creation of an acknowledgement message according to 5.3.2.4.
- 4) The received message "M" indicates "acknowledgement message is required", i.e., the value of "ackRequest" is set to "true".

5.3.2.5.2 Acknowledgement message

An acknowledgement message "N" is related to a received message "M" and indicates whether the received message "M" could be processed or received successfully (positive acknowledgement) or not (negative acknowledgement, i.e., error indication). More precisely, the following scopes of "N" can be assigned.

- 1) **Application success**
Positive acknowledgement: The recipient of "M" received and evaluated and processed "M" successfully.
- 2) **Application error**
Negative acknowledgement: The recipient of "M" encountered a problem with "M" or an error occurred with the transmission of "M" to the recipient of "M".
- 3) **Transmission success**
Positive acknowledgement: The recipient of "M" confirms it received "M".
- 4) **Transmission error**
Negative acknowledgement: An error occurred with the transmission of "M" to the recipient of "M".

Which scope applies is defined in 5.3.2.4.

As the result classifier is used for acknowledgement messages, such messages are also called "result messages".

A "result message" is composed as follows.

Element "cmdClassifier" shall be set to "result". The remaining header elements are set as if the message is a regular reply to the related received message. Furthermore, the element "payload" shall contain a "resultData" function. The element "errorNumber" of "resultData" expresses the kind of the message and shall be present and set as follows.

- 1) A value of "0" shall be used for "positive acknowledgement".
- 2) Every other value denotes a "negative acknowledgement".

In addition, the element "description" of "resultData" may be present and filled with a readable information.

In case a "result message" cannot be sent within time according to "defaultMaxResponseDelay" or "maxResponseDelay" described in 5.3.2.5.3, refer to 5.3.2.5.3.

NOTE 5.3.2.5.2 does not specify the circumstances when an acknowledgement message is to be sent or not. For this, see 5.3.2.4 and 5.3.2.5.1.

5.3.2.5.3 Delayed application response

For each feature server, a device needs some time to generate a proper response (which could be a message with "reply" classifier or "result" classifier) upon a received request. To have an interoperable concept for devices to know how long to wait for a response, a timeout mechanism with "maximum response delay" is used.

The duration specified by "maximum response delay" only relates to the immediate access to the communication interface of a device. This means that latencies from communication channels are not covered by "maximum response delay".

NOTE Each communication channel has some latency. This latency depends on the kind of communications technology itself as well as on the environment (especially wireless technologies often suffer from poor coverage or too many participants sharing the available bandwidth). As specified, this kind of latency is independent from the "maximum response delay".

There are two timeout levels to derive the "maximum response delay" for a given server feature.

- 1) By default, the "maximum response delay" is equal to defaultMaxResponseDelay. The defaultMaxResponseDelay shall be 10 s.
- 2) If the maxResponseDelay element within the feature description of the detailed discovery (see 5.5.2.3) is set AND the value of the maxResponseDelay element is larger than zero seconds, the value of the maxResponseDelay element shall be applied for this feature as "maximum response delay" (i.e. instead of defaultMaxResponseDelay).

The timeout levels number 3) and 4) are recommended on the value of maxResponseDelay.

- 3) If a feature server will usually or frequently need more time than defaultMaxResponseDelay to generate proper replies, it should set the maxResponseDelay element in the feature description of the detailed discovery (see 5.5.2.3) to a duration that the feature server can almost always fall short of for the generation of proper replies.
- 4) If a feature server will almost always need less time than defaultMaxResponseDelay to generate proper replies, it may set the maxResponseDelay element in the feature description of the detailed discovery (see 5.5.2.3) to a duration that the feature server can almost always fall short of for the generation of proper replies.

The rules number 5) to 7) are the rules on the use of "maximum response delay".

- 5) An implementation that complies with this or a subsequent version shall be able to handle response delays of at least defaultMaxResponseDelay.
- 6) A feature client may use "maximum response delay" for the detection of a response-timeout even if "maximum response delay" is shorter than defaultMaxResponseDelay.
- 7) An implementation should consider the communications technology specific latency carefully before it begins with the detection of a timeout to an expected response.

5.3.2.6 Time information in "timestamp"

The header's (optional) element "timestamp" uses the type AbsoluteOrRelativeTimeType. In case of absolute times, the UTC zone shall be applied. A valid example is "2016-04-28T19:43:14.3Z" (in contrast to "2016-04-28T17:43:14.3-02:00" or even the bare local time "2016-04-28T17:43:14.3").

The application of relative times (i.e. according to the type "xs:duration") is of limited use. It is usually only useful for devices that "collect" their data/messages over a certain period and send them only at specific times (e.g., battery powered devices; these often also have no real time clock/UTC setup). In this case, the (negative) relative time is relative to "now" and indicates when the message was created in the past.

5.3.2.7 Structure

The structure of the SPINE header shall be set as shown in Table 3.

Table 3 – Structure of the SPINE header

Element name	Type	M/O/NV/C	Brief explanation
datagram. header		M	The header element shall be present.
datagram. header. specificationVersion	SpecificationVersionType (see 5.2.2)	M	The version of the SPINE data model applicable to the XML. Shall be present.
datagram. header. addressSource	FeatureAddressType (see 5.2.2)	M	The address of the submitter of this XML. Shall be present.
datagram. header. addressSource. device	AddressDeviceType; see "Device address" in 5.5.2.2.2	O	Device part of the source address. Should be present in simple communication mode, shall be present in all other communication modes. If the element is present, it shall not be empty. See 5.3.2.2.1 for details.
datagram. header. addressSource. entity (list)	AddressEntityType (see 5.2.2)	M	Entity part(s) of the source address. Shall be present. See also 5.2.3, especially concerning restrictions on the supported depth.
datagram. header. addressSource. feature	AddressFeatureType (see 5.2.2)	M	Feature part of the source address. Shall be present.
datagram. header. addressDestination	FeatureAddressType	M	The address of the receiver of this XML. Shall be present.
datagram. header. addressDestination. device	AddressDeviceType; see "Device address" in 5.5.2.2.2	O	Device part of the destination address. Should be present in simple communication mode, shall be present in all other communication modes. If the element is present, it shall not be empty. See 5.3.2.2.1 for details.
datagram. header. addressDestination. entity (list)	AddressEntityType (see 5.2.2)	M	Entity part(s) of the destination address. Shall be present. See also 5.2.3, especially concerning restrictions on the supported depth.

Element name	Type	M/O/NV/C	Brief explanation
datagram. header. addressDestination. feature	AddressFeatureType (see 5.2.2)	M	Feature part of the destination address. Shall be present.
datagram. header. addressOriginator	FeatureAddressType (see 5.2.2)	O	Can be used to model information of a "forwarded" SPINE message. The element would contain the address of the original submitter. May be present. See also 5.3.2.2.2.
datagram. header. addressOriginator. device	AddressDeviceType; see "Device address" in 5.5.2.2.2	M	Device part of the originator address. Shall be present.
datagram. header. addressOriginator. entity (list)	AddressEntityType (see 5.2.2)	M	Entity part(s) of the originator address. Shall be present. See also 5.2.3, especially concerning restrictions on the supported depth.
datagram. header. addressOriginator. feature	AddressFeatureType (see 5.2.2)	M	Feature part of the originator address. Shall be present.
datagram. header. msgCounter	xs:unsignedLong	M	The message number of the submitter of this SPINE message. Shall be present.
datagram. header. msgCounterReference	xs:unsignedLong	O	The message number of the related SPINE message. Shall not be set if the message does not relate to another message. Otherwise it shall be present and set to the message number of the related message.
datagram. header. cmdClassifier	CmdClassifierType (see 5.3.2.4)	M	The so-called "classifier" associated to the given SPINE function. This denotes for which kind of operation a function is used (read, write, notify, etc.). Shall be present. See 5.3.2.4.
datagram. header. ackRequest	xs:boolean	O	Indicates the kind of "acknowledgement request" of the message. The value "true" indicates that an explicit acknowledgement message for this message is requested. Can be present. If absent, the default value "false" applies. See 5.3.2.5.1 for details.
datagram. header. timestamp	AbsoluteOrRelativeTimeType (see 5.2.2)	O	The timestamp of the creation of this SPINE message. Can be present. See 5.3.2.6.

5.3.3 Payload

5.3.3.1 General

Within the element "payload", a single "command" can be placed (broadly speaking; the data model is also prepared for future extensions, but this is not considered further in detail in this document). It permits or even requires the presence of additional elements to express/identify a functionality, as shown in Figure 8. Details are discussed in 5.3.3.2 to 5.3.3.4.

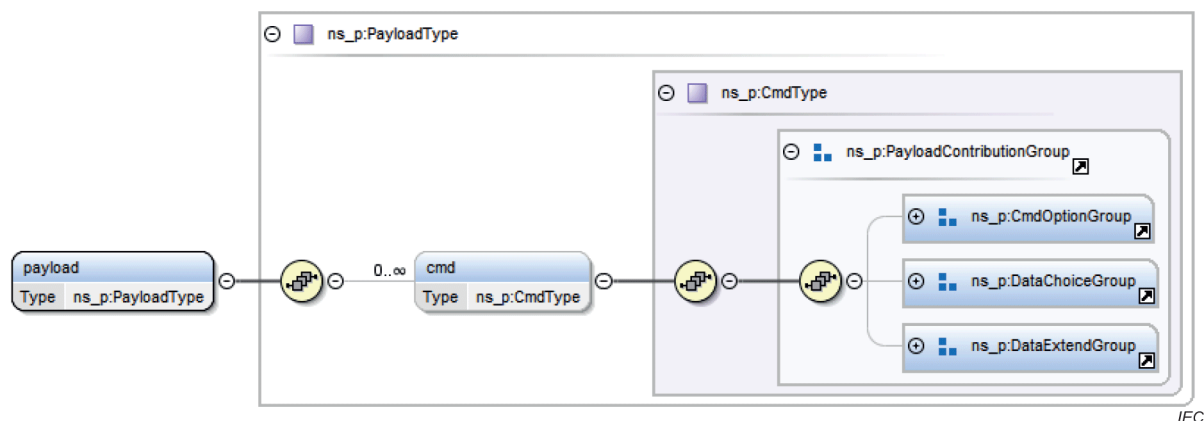


Figure 8 – SPINE payload

5.3.3.2 Elements and usage

Within the protocol, the elements of the SPINE payload shall be set as shown in Table 4.

Table 4 – Elements of the SPINE payload

Element name	Type	M/O/NV/C	Brief explanation
datagram. payload		M	The payload element shall be present.
datagram. payload. cmd		1..unbounded	Each "cmd" instance can take information for one function. In this document, just one (the first) occurrence of "cmd" within "payload" shall be used (i.e. although the data model permits the occurrence of multiple "cmd" instances in theory, just one is considered).
datagram. payload. cmd. function	FunctionType (see 5.2.2)	O	Contains the function name of element "datagram.payload.cmd.<_function_>". Shall be present if datagram.payload.cmd.filter is present. Shall be absent otherwise. See 5.3.3.4.
datagram. payload. cmd. filter		0..unbounded	Identifies the content that shall be restricted (see 5.3.3.4). Although the model permits multiple occurrences, this document permits two occurrences at maximum. Shall only be present if at least one child element is present.
datagram. payload. cmd. filter. filterId	xs:unsignedInt	O	Reserved for future use.
datagram. payload. cmd. filter. cmdControl		O	Specifies the kind of function/data restriction. Shall be present for specified function/data restrictions only. Shall be absent otherwise. See 5.3.3.4. In this document, exactly one of the possible child elements shall be present if datagram.payload.cmd.filter.cmdControl is present.
datagram. payload. cmd. filter. cmdControl. delete		O	Denotes the restriction is a "delete" operation.

Element name	Type	M/O/NV/C	Brief explanation
datagram. payload. cmd. filter. cmdControl. partial		O	Denotes the restriction is a "partial" operation.
datagram. payload. cmd. filter. <_selectors_>		0..unbounded	"<_selectors_>" is a placeholder for SPINE class specific "Selectors" definitions. E.g. for a SPINE function "X" there can be a specific "XSelectors" defined. See the SPINE data model for a list of all possible entries. If present, it shall be an instance of the specific "selectors" of the function "datagram. payload. cmd. <_function_>". I.e. with the function name of "...cmd.<_function_>" abbreviated with "X", the "Selectors" name in "<_selectors_>" shall be "XSelectors". Can occur multiple times. All occurrences shall be interpreted as logical "OR" operation.
datagram. payload. cmd. filter. <_elements_>		O	"<_elements_>" is a placeholder for SPINE class specific "Elements" definitions. E.g. for a SPINE function "X" there can be a specific "XElements" defined. Please look at the SPINE data model for a list of all possible entries. If present, it shall be an instance of the specific "Elements" of the function "datagram. payload. cmd. <_function_>". I.e. with the function name of "...cmd.<_function_>" abbreviated with "X", the "Elements" name in "<_elements_>" shall be "XElements".
datagram. payload. cmd. <_function_>		M	"<_function_>" is a placeholder for exactly one SPINE function. Shall be present.
datagram. payload. cmd. manufacturerSpecificExtension	xs:hexBinary	O	Can be used to extend a given function with proprietary data. It shall only be used together with a function. May be present.
datagram. payload. cmd. lastUpdateAt	AbsoluteOrRelativeTime Type (see 5.2.2)	O	Can be used in case an implementation caches data of the actual node. May be present.

On each feature there shall be at maximum one class implemented with regard to the feature's primary functionality.

The general XSD-based SPINE payload definition permits the presence of multiple "cmd" instances and therefore multiple functions in theory. However, within this version of the protocol, each instance of a "payload" element shall contain exactly one "cmd" instance.

5.3.3.3 Ownership

The combination of functions and classifiers/operations of the SPINE data model should be applied considering the concept of "ownership". The concept of "ownership" is directly linked to the SPINE role concept. The "ownership" of data is always defined on SPINE feature level.

The "owner" of data (functions) can

- notify its own data,
- reply a request (i.e. received read operation) with its own data, and
- update its own data upon request (i.e., upon received write operation).

This, of course, only provided that these actions are supported by the owner.

Especially the write operation should be considered. As example we assume device "A" with a client feature "AA" submits a write operation with data "X" to device "B" with server feature "BB". Then, data "X" shall be considered to be owned and "known" by feature "BB" of device "B", i.e. device "A" should not expect device "B" considers "X" as data owned by feature "AA" of device "A".

Call operations and acknowledgement messages (with classifier "result") have a different scope, hence do not belong to the concept of ownership.

5.3.3.4 Restricted function exchange with cmdOptions

5.3.3.4.1 Overview

The restricted function exchange (RFE) concept is used to exchange just a certain restricted part of a function instead of the full function. The following elements of "datagram.payload.cmd" are subsequently called "cmdOptions":

- 1) datagram.payload.cmd.function;
- 2) datagram.payload.cmd.filter;
- 3) datagram.payload.cmd.filter.cmdControl.

The cmdOption "datagram.payload.cmd.function" shall be used and include the correct function name if and only if any of the other cmdOptions is used.

If full functions are requested or exchanged ("full function exchange"), cmdOptions are not used.

NOTE 1 The cmdOption "datagram.payload.cmd.function" just serves as some kind of preface in order to introduce which function to operate on in some cases. I.e., it does not contain function data.

The basic difference between restricted and full functions is determined by the absence or presence of cmdOptions.

NOTE 2 5.3.3.4.2 shows cmdOptions combinations that also contain the case of full function exchange for comparison.

In general, the support of restricted function exchange is optional unless a featureType or use case requires the support explicitly. Even then, the featureType or use case can support only specific kinds of restricted function exchange. Please also consider 5.3.3.4.9.

Let's assume "T" is a data model definition (including proper presence indications for each element) for a full function exchange. An example for "T" is the SPINE function "smartEnergyManagementPsData" of the feature type "SmartEnergyManagementPs". Subclause 5.3.3.4 defines general requirements of "restricted function exchange" and it defines for specific cases which elements are required at least. For example, a "notify" with "cmdControl" set to "partial" can be used to convey a <_function_> that contains added or modified function parts. In this case, added and modified elements are required, whereas unchanged elements (even if the definition of "T" designates them as "mandatory") are not required (unless rules of 5.3.3.4 impose specific requirements as in case of, e.g., identifiers). Of course, for each required (child) element, its respective parent elements are always mandatory.

NOTE 3 A featureType can specify one or more specifically restricted variants of "T". For example, the feature type "SmartEnergyManagementPs" defines the "primary use" of the SPINE function "smartEnergyManagementPsData" to express energy management related information of a device (conveyed in a "read" or "notify" operation). Additionally, the featureType also defines variants of this SPINE function to shift a functionality of the device or select a different option (these variants use "write" operations). These variants can be seen as extracts or simplifications of the "primary use" of the function. They are especially useful to describe dedicated changes of the device's SPINE function.

On the cardinality of "filter", the following is important.

As shown in Table 4, the element "datagram. payload. cmd. filter" can occur more than one time. This cardinality is used to permit a restricted function exchange with two "filter" parts within one message: one "filter" part with sub-element "cmdControl" set to "delete" and another "filter" part with sub-element "cmdControl" set to "partial". As brief example, we take one of the permitted "write" cmdOptions combinations of 5.3.3.4.2 where a "delete" as well as a "partial" part are used (functional details on these parts are explained in 5.3.3.4.2). We assume some "xyzData" list entries of the function "xyzListData" are deleted and some list entries are partially modified for this example. The "payload" part could then look like this (with some content replaced by "..." to improve readability):

```
...
<payload>
  <cmd>
    <function>xyzListData</function>
    <filter>
      <cmdControl><delete/></cmdControl>
      <xyzListDataSelectors>...</xyzListDataSelectors>
      <xyzDataElements>...</xyzListDataElements>
    </filter>
    <filter>
      <cmdControl><partial/></cmdControl>
      <xyzListDataSelectors>...</xyzListDataSelectors>
    </filter>
    <xyzListData>
      <xyzData>
        ...
      </xyzData>
    </xyzListData>
  </cmd>
</payload>
```

In 5.3.3.4, the combinations are explained in tables where each "filter" part is grouped together with the respective child elements in three columns. In Table 5 the first three columns with "cmdControl", "<_elements_>", and "<_selectors_>" belong to a "filter" element where "cmdControl" is set to "delete". Likewise, the second group (columns 4 to 6) belongs to a "filter" element where "cmdControl" is set to "partial". The "payload" example corresponds to the penultimate row of Table 5.

Table 5 – Example table (template)

Filter with "delete"			Filter with "partial"				Explanation and rules
cmdControl	^_elements_	^_selectors_	cmdControl	^_elements_	^_selectors_	^_function_	
-	-	-	partial	-	-	X	...
...	...	...	...	...	...	...	...
delete	-	X		-	-	dc	...
...	...	...	...	...	...	...	...
delete	(X)	(X)	partial	-	(X)	X	...
-	-	-	-	-	-	X	...

NOTE The template in Table 5 is used in 5.3.3.4.2 to 5.3.3.4.9 for specific cmdOptions combinations. In this template, each "..." is just a placeholder.

On the application of RFE to lists, the following applies.

Within the RFE concept, two types of lists should be distinguished. In general, a list comprises list entries. At a server, a list contains either only identifiable list entries or only non-identifiable list entries.

- **Identifiable list entries:** a list entry that can be identified at a server via primary identifier or sub-identifier.

NOTE 4 This definition applies also for new list entries to be added via write command, provided that the write command will be accepted by the server.

- **Non-identifiable list entries:** a list entry that is not an identifiable list entry.

For identifiable list entries, the RFE concept permits to transport only some entries of the list, i.e., without the need to transport all entries of the list.

For non-identifiable list entries, on the other hand, it is not possible to transport only some entries of the list. This means that a list with non-identifiable list entries can only be transmitted as a whole. Consequently, requesting changes (via "write" command) or reporting changes (via "notify" message) of only parts in such a list can only be expressed by the complete exchange of the list.

The <_selectors_> permit selection of identifiable list entries using identifiers or other criteria. The result should not be interpreted as "identification" of list entries, as <_selectors_> is rather intended as filtering mechanism. See 5.3.3.4.7 for details.

The RFE concept is not limited to exchange only specific list entries. In general, it permits also to exchange only such elements that got or shall get a value change (eventually accompanied with other elements that are required unconditionally in the corresponding message). In combination, this enables to exchange specific list entries that contain only elements with changed value. Detailed rules are given in 5.3.3.4.2 to 5.3.3.4.9.

5.3.3.4.2 "write" cmdOptions combinations

Table 6 shows which kind of cmdOptions combinations can be considered for the use with classifier "write".

Table 6 – Considered cmdOptions combinations for classifier "write"

Filter with "delete"			Filter with "partial"				
cmdControl	<_elements_>	<_selectors_>	cmdControl	<_elements_>	<_selectors_>	<_function_>	Explanation and rules
-	-	-	partial	-	-	X	<_function_> contains function parts to add or modify. Additionally, <_function_> can restrict the locations (specific identifiable list items) by using identifiers, as described in 5.3.3.4.6. Applies only to lists where the server has identifiable list items: List items in <_function_> that have NO identifier shall be applied to all corresponding list entries of the data owner ^a .
-	-	-	partial	-	X	X	<_selectors_> specify locations (specific identifiable list items), as described in 5.3.3.4.7, where <_function_> data are added or modified. <_function_> shall not use identifiers. <_selectors_> shall match with already existing locations. Therefore, it is not possible to add new list entries to a list with identifiable list items with this combination.
delete	-	X		-	-	dc	Locations (specific identifiable list items) specified by <_selectors_> shall be deleted.
delete	X	-		-	-	dc	Elements specified by <_elements_> shall be deleted.
delete	X	X		-	-	dc	Locations (specific identifiable list items) and elements that shall be deleted are determined by <_selectors_> and <_elements_>, according to 5.3.3.4.7.
delete	(X)	(X)	partial	-	(X)	X	At first, the filter with cmdControl "delete" shall be applied according to the "delete" combinations. Afterwards, <_function_> and filter with cmdControl "partial" shall be applied according to the "partial" combinations.
-	-	-	-	-	-	X	Full write.
"X" means a proper instance is present and not empty in the message. "(X)" means a proper instance can be present. If present, it is not empty. "-" means no such item is in the message. "dc" means that the corresponding instance shall be present but can be ignored (don't care; i.e. in case of pure "delete" commands (no additional "partial" part) a <_function_> instance shall be present but shall be empty). ^a Typically, identifiable list entries that are to be changed are selected individually by their identifiers. But the marked rule describes instead a possibility where a single list item of <_function_> is applied to all list items (of the same name and hierarchy) at the server. This means that the present elements of <_function_> are "copied" to all corresponding list items' elements.							

In this document, at maximum one "delete" filter and at maximum one "partial" filter shall be used in one command. If both filters are present, the "delete" filter shall be present before the "partial" filter in the command.

In general, a write operation with restricted function exchange shall only be executed by a server if it can execute the received operation completely.

5.3.3.4.3 "notify" cmdOptions combinations

A "notify" command is very similar constructed like a "write" command, because a notify is mostly used to communicate what has changed after a write process. Be it an external write or internal change, both can be viewed as write processes. Therefore, the "notify" cmdClassifier permits also the same combinations as the "write" cmdClassifier. Consider the details provided in 5.3.3.4.2.

Table 7 shows which kind of cmdOptions combinations can be considered for the use with classifier "notify".

Table 7 – Considered cmdOptions combinations for classifier "notify"

Filter with "delete"			Filter with "partial"				
cmdControl	<_elements_>	<_selectors_>	cmdControl	<_elements_>	<_selectors_>	<_function_>	Explanation and rules
-	-	-	partial	-	-	X	<_function_> contains added or modified function parts. Additionally, <_function_> can restrict the locations (specific identifiable list items) by using identifiers, as described in 5.3.3.4.6. Applies only to lists where the server has identifiable list items ^a : list items in <_function_> that have NO identifier shall be applied to all corresponding list entries of the data owner.
-	-	-	partial	-	X	X	<_selectors_> specify locations (specific identifiable list items), as described in 5.3.3.4.7, where <_function_> data was added or modified. <_function_> shall not use identifiers.
delete	-	X		-	-	dc	Locations (specific identifiable list items) specified by <_selectors_> were deleted.
delete	X	-		-	-	dc	Elements specified by <_elements_> were deleted.
delete	X	X		-	-	dc	Locations (specific identifiable list items) and elements specified by <_selectors_> and <_elements_> were deleted, according to 5.3.3.4.7.
delete	(X)	(X)	partial	-	(X)	X	At first, elements specified by filter with cmdControl "delete" were deleted according to the "delete" combinations. Afterwards, <_function_> and filter with cmdControl "partial" were applied according to the "partial" combinations.
-	-	-	-	-	-	X	Full notify.
"X" means a proper instance is present and not empty in the message. "(X)" means a proper instance can be present. If present, it is not empty. "-" means no such item is in the message. "dc" means that the corresponding instance shall be present but can be ignored (don't care; i.e. in case of pure "delete" commands (no additional "partial" part) a <_function_> instance shall be present but shall be empty). ^a Typically, identifiable list entries that changed are listed explicitly with their identifiers. But the marked rule describes instead a possibility where a single list item of <_function_> contains changes of all list items (of the same name and hierarchy) at the server. This means that the present elements of <_function_> expresses the same changes of all corresponding list items' elements.							

In this document, at maximum one "delete" filter and at maximum one "partial" filter shall be used in one command. If both filters are present, the "delete" filter shall be specified before the "partial" filter in the command order.

5.3.3.4.4 "read" cmdOptions combinations

Table 8 shows which kind of cmdOptions combinations can be considered for the use with classifier "read".

Table 8 – Considered cmdOptions combinations for classifier "read"

Filter with "partial"				
"cmdControl"	<_selectors_>	<_elements_>	<_function_>	Explanation and rules
partial	X	-	present but EMPTY	Locations (specific identifiable list items) specified by <_selectors_>, as described in 5.3.3.4.7, shall be returned.
partial	-	X	present but EMPTY	Elements specified by <_elements_> shall be returned.
partial	X	X	present but EMPTY	Locations (specific identifiable list items) and elements that shall be returned are determined by <_selectors_> and <_elements_> according to 5.3.3.4.7.
-	-	-	-	Full read.
"X" means a proper instance is present and not empty in the message. "-" means no such item is in the message.				

5.3.3.4.5 "reply" cmdOptions combinations

Table 9 shows which kind of cmdOptions combinations can be considered for the use with classifier "reply".

Table 9 – Considered cmdOptions combinations for classifier "reply"

Filter with "partial"				
"cmdControl"	<_selectors_>	<_elements_>	<_function_>	Explanation and rules
partial	-	-	X	<_function_> contains requested function parts. <_function_> can contain identifiers, as described in 5.3.3.4.6. In case of lists with identifiable list items, identifiers of each list item in the reply shall be full, even if the corresponding "read operation" made use of elements selection with "<_elements_>" but did not specify the elements of the identifier.
-	-	-	-	Full reply.
"X" means a proper instance is present and not empty in the message. "-" means no such item is in the message.				

A server can ignore unsupported cmdOption combinations and then replies with more than the requested parts instead. If the server does not support cmdOptions with "read" at all, it shall respond with a full reply.

5.3.3.4.6 <_function_> identifiers – Implicit list item selection

5.3.3.4.6.1 General

Identifiers are used to select specific list entries directly in the <_function_> (element "datagram.payload.cmd.<_function_>" of Table 4). The approach requires that list items can be identified in a unique way.

If a featureType specifies identifiers and their use, then the values of identifiers in a (full or restricted) instance of "<_function_>" serve implicitly for the identification of the proper list item.

The read request belonging to this example can be found in 5.3.3.4.7. As requested, device "A" responds with this XML:

```
<datagram>
  <header>
    ...
    <cmdClassifier>reply</cmdClassifier>
  </header>
  <payload>
    <cmd>
      <function>measurementListData</function>
      <filter>
        <cmdControl>
          <partial/>
        </cmdControl>
      </filter>
      <measurementListData>
        <measurementData>
          <measurementId>5</measurementId>
          <valueType>minValue</valueType>
          <timestamp>2015-07-14T15:00:00.0Z</timestamp>
          <value>
            <number>-173</number>
            <scale>-1</scale>
          </value>
          <evaluationPeriod>
            <startTime>2015-07-14T10:00:00.0Z</startTime>
            <endTime>2015-07-14T15:00:00.0Z</endTime>
          </evaluationPeriod>
          <valueSource>measuredValue</valueSource>
          <valueTendency>rising</valueTendency>
          <valueState>normal</valueState>
        </measurementData>
      </measurementListData>
    </cmd>
  </payload>
</datagram>
```

In this example, it is assumed the device keeps no further history of "minValue" values, i.e., it keeps just the latest "minValue".

The following rules apply if cmdOptions are used.

- 1) <_function_> identifiers shall only be used for "partial" write, or notify, or reply commands.
- 2) If <_function_> identifiers are used, it is not allowed to use <_selectors_> in a filter with cmdContol "partial".
- 3) If <_function_> contains at least one identifier, ALL list items included in <_function_> shall have complete identifiers, i.e., it is not valid to have a list item without identifier and another list item with identifier within a <_function_> instance.

5.3.3.4.6.2 Identifier hierarchy and completeness of list identifiers

SPINE class functions with multiple identifiers usually assign the identifiers a "natural hierarchy" (e.g., an identifier "xld" serves as some kind of "parent identifier" for an identifier "yld"; a common example could be a postal address where the identifier "city" is a "parent identifier" of the identifier "street"). A featureType can explicitly define which identifiers are used and which hierarchy applies. If the featureType does not specify identifiers, the default identifiers of the underlying class apply and the hierarchy follows the order stated in the identifier section of the class description.

The identifiers of a list item are complete if no identifier is added to identify the target list item uniquely.

Among others, identifiers permit "finding" or choosing specific list entries of a function containing a large list. An example is shown in 5.3.3.4.7.

5.3.3.4.7 <_selectors_> – Explicit list item selection

5.3.3.4.7.1 General

5.3.3.4.7 describes the use of element "datagram.payload.cmd.filter.<_selectors_>" of Table 4.

5.3.3.4.7.2 Common rules and description

<_selectors_> are used to select specific identifiable list entries by referencing a value or value range of particular child elements from the corresponding list entry. One possibility is to select identifiable list entries by one or more identifiers. Some <_selectors_> also provide other search criteria.

NOTE 1 The use of search criteria that are not identifiers cannot be interpreted as "identification" of a list entry; rather, it just permits to select specific identifiable list entries.

All identifiable list entries, where every referenced child element has a corresponding value, are selected. This means, the identifiable list entry with all of its child elements (not just the referenced elements) of the identifiable list entry are selected. It also means that an identifiable list entry is not selected if no or just some of the referenced child elements match the search criteria. Usage of <_elements_> allows restricting a selection further.

NOTE 2 The requirement that all referenced child elements can match the corresponding search criteria can be interpreted as logical "AND" operation. Table 4 describes how a logical "OR" operation can be achieved using multiple <_selectors_>.

As explained below, an identifiable list entry can even contain a "nested list", and a proper <_selectors_> can support search criteria for this nested list. If this nested list is not an identifiable list and a given <_selectors_> matches a proper value of this nested list, then the surrounding identifiable list entry is selected.

The featureType specifies which values can be used for <_selectors_> and what elements are exactly selected when a match is found.

In this document, list structures can occur in the following functions.

- 1) In list-based SPINE "standard class" functions ("measurementListData", e.g.). All SPINE "standard class" functions of the pattern "xListData" are list-based SPINE standard class functions containing a list of "xData" entries. Some "xData" also contain an nested list.
- 2) In SPINE complex class functions with nested list structures. Some functions of, for example, the complex class "SmartEnergyManagementPs" have nested list structures. As complex functions are composed of basic functions (or at least their types), the selectors of complex functions are composed of (one or more) selectors (or at least their types) of list-based standard class functions.

<_selectors_> are mostly needed for "filter" with cmdControl "delete" and for "partial" read commands. However, <_selectors_> also allow "partial" writing or notifying the same values in multiple identifiable list entries at the same time.

For many resources, default values are defined for certain elements. If such an element is omitted within its parent element (i.e., the parent element is present), the default value applies. If a client explicitly sets an element with its default value in a selector but the server has previously omitted the corresponding element within its parent element (i.e., the parent element is present), the server should still successfully match the corresponding selector element value against the default value.

The following is an example of a "read operation". We assume device "A" is the owner of a feature with a "measurementListData" instance. Device "B" wants to get a copy of this instance where only such "measurementData" list items are embedded that have the element "measurementId" set to "5" AND "valueType" set to "minValue". Thus, device "B" sends the following read operation:

```
<datagram>
  <header>
    ...
    <cmdClassifier>read</cmdClassifier>
  </header>
  <payload>
    <cmd>
      <function>measurementListData</function>
      <filter>
        <cmdControl>
          <partial/>
        </cmdControl>
        <measurementListDataSelectors>
          <measurementId>5</measurementId>
          <valueType>minValue</valueType>
        </measurementListDataSelectors>
      </filter>
    </cmd>
  </payload>
</datagram>
```

It can be seen that the "read" request with the empty function "measurementListData" is accompanied with the dedicated selectors "measurementListDataSelectors" and proper values.

5.3.3.4.7.3 Selectors with address elements

Some selectors can take address elements to select a specific "device" address part or "entity" address part or "feature" address part. In fact, for such selectors parts, the same rules apply. However, especially due to the intrinsic list structure of "entity" address parts, only exact matches of an "entity" address part of a selectors with an "entity" address part within a function are considered as valid matches.

We consider an extract of the address tree example from 5.2.3. The tree begins with the device address part "someDevice":

```
"someDevice"
...
+--- entity 1
|       +--- entity 4 (child of "someDevice"/entity 1)
|       |       +--- feature 7 (*1)
...
+--- entity 4          (child of "someDevice")
|       +--- feature 1 (child of "someDevice"/entity 4)

(*1): child of "someDevice"/entity 1/entity 4
```

From this tree, some address paths are shown with their XML representation in Table 10. The "someDevice" device address part is simplified for brevity and – according to the permitted pattern – shown in full length in these XML examples.

Table 10 – Address path examples

No.	Address path	XML representation
A1	device "someDevice"	<device>d:_i:46 925_someDevice</device>
A2	device "someDevice" / entity 1	<device>d:_i:46 925_someDevice</device> <entity>1</entity>
A3	device "someDevice" / entity 1 / entity 4	<device>d:_i:46 925_someDevice</device> <entity>1</entity> <entity>4</entity>
A4	device "someDevice" / entity 1 / entity 4 / feature 7	<device>d:_i:46 925_someDevice</device> <entity>1</entity> <entity>4</entity> <feature>7</feature>
A5	device "someDevice" / entity 4	<device>d:_i:46 925_someDevice</device> <entity>4</entity>
A6	device "someDevice" / entity 4 / feature 1	<device>d:_i:46 925_someDevice</device> <entity>4</entity> <feature>1</feature>

Now, we consider an extract of a selectors (in this case of "nodeManagementDetailedDiscoveryDataSelectors") with an "entityAddress" part comprising of the optional child element "device" and an optional list of "entity" items (see Figure 9).

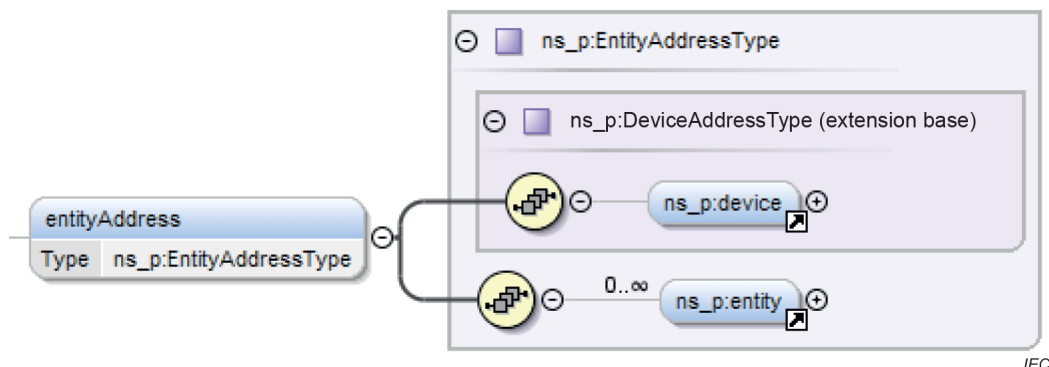


Figure 9 – Example of selectors part (extract) with entity address part

A selectors instance like

```
<entityAddress>
  <entity>1</entity>
  <entity>4</entity>
</entityAddress>
```

would match address parts A3 and A4 of Table 10.

A selectors instance like

```
<entityAddress>
  <entity>4</entity>
</entityAddress>
```

would match address parts A5 and A6 of Table 10, but neither A3 nor A4.

5.3.3.4.8 <_elements_> - Selection of "elements"

Each function (element "datagram.payload.cmd.<_function_>" of Table 4) is defined with specific (usually optional) child elements. The <_elements_> definition (element "datagram.payload.cmd.filter.<_elements_>" of Table 4) includes all elements from <_function_> but without type and value. Within the "<_elements_>" definition, it is possible to tell explicitly which subset of a function's elements is to be considered.

NOTE <_elements_> does not include identifiers in most cases.

<_elements_> shall only be used in the following two cases:

- data deletion (write or notify);
- "partial" read.

An "<_elements_>" instance contains no values that can be used for identification of certain list items. However, the application should consider as well if certain list items are selected explicitly via <_selectors_> (5.3.3.4.7).

Subsequently, we iterate through all elements of a given "<_elements_>" instance and call the iterated element "<_elements_>". The following rules determine whether <_elements_> shall be applied to the corresponding element of the target function and (in case of list items) to which list item of the target function it shall be applied.

- 1) If <_selectors_> is not used in the command:
 <_elements_> is applied.
- 2) If <_selectors_> is used in the command:
 <_elements_> is applied only for list items selected via <_selectors_>.

Among others, this means that the restricted function exchange can denote specific list items and the "<_elements_>" content then shall only be applied to those list items and not to other list items of the data owner.

5.3.3.4.9 Minimum restricted function exchange support

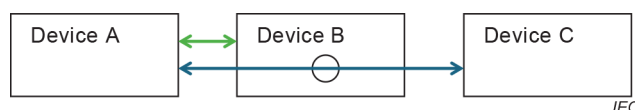
Especially for "basic functions" (standard featureTypes with list-based standard class functions, e.g.) it is useful to define whether and how a certain kind of minimum support for restricted function exchange can be achieved. Therefore, a feature that supports such kind of minimum restricted function exchange shall indicate this by stating the "partial" flag in the "possibleOperations" for "read" or "write" (or both) for a respective function (see 5.5.2.2.5.6).

5.4 Communication modes

5.4.1 General

In Figure 10, different communication modes are briefly depicted.

- 1) A green arrow shows the simple communication mode which assumes the direct communication between two SPINE devices.
- 2) A blue arrow shows the enhanced communication mode, which provides the possibility to send messages via intermediate devices, as needed by a (SPINE) technology gateway. In Figure 10 messages between devices "A" and "C" are exchanged via device "B".



NOTE The circle in device B symbolizes the "message forwarding" task of device B.

Figure 10 – Communication modes of SPINE devices A, B and C

The modes are only used to distinguish the kind of the connection or message flow according to Figure 10. For each mode, different requirements apply. This permits some simplification with regard to the use of address fields if SPINE devices are "directly connected" and only exchange data belonging to the respective communication partner.

The simple communication mode shall be supported by every SPINE device.

The enhanced communication mode shall be supported by all SPINE devices that have their networkFeatureSet element (see 5.5.2.2.5.4) set to a value different than "simple" (e.g. "smart" or "gateway" or "router").

If two devices exchange messages directly (without any device "between" them), the simple communication mode applies. If two devices need (at least) a third one to communicate with each other, the enhanced communication mode applies.