



DPC: 23 / 30456076 DC

BSI Group Headquarters

389 Chiswick High Road London W4 4AL

Tel: +44 (0)20 8996 9000

Fax: +44 (0)20 8996 7400

www.bsigroup.com

Date: 08 February 2023

Origin: International

Latest date for receipt of comments: 29 March 2023

Project No. 2022/00975

Responsible committee: PEL/69 Electric vehicles

Interested committees:

Title: BS EN 63380-3 Local Charging station management systems and Local Energy Management Systems network connectivity and information exchange

Part 3: Communication Protocol and Cybersecurity Specific Aspects

Please notify the secretary if you are aware of any keywords that might assist in classifying or identifying the standard or if the content of this standard

- i) has any issues related to 3rd party IPR, patent or copyright
- ii) affects other national standard(s)
- iii) requires additional national guidance or information

**WARNING: THIS IS A DRAFT AND MUST NOT BE REGARDED OR USED AS A BRITISH STANDARD.
THIS DRAFT IS NOT CURRENT BEYOND 29 March 2023**

This draft is issued to allow comments from interested parties; all comments will be given consideration prior to publication. No acknowledgement will normally be sent. **See overleaf for information on the submission of comments.**

No copying is allowed, in any form, without prior written permission from BSI except as permitted under the Copyright, Designs and Patent Act 1988 or for circulation within a nominating organization for briefing purposes. Electronic circulation is limited to dissemination by e-mail within such an organization by committee members.

Further copies of this draft may be purchased from BSI Shop <http://shop.bsigroup.com> or from BSI Customer Services, Tel: +44(0) 20 8996 9001 or email cservices@bsigroup.com. British, International and foreign standards are also available from BSI Customer Services.

Information on the co-operating organizations represented on the committees referenced above may be obtained from <http://standardsdevelopment.bsigroup.com>

Responsible Committee Secretary: **Ms Andreea Vieru (BSI)**
Direct tel: **0208 996 7432**
E-mail: **andreea.vieru@bsigroup.com**

Introduction

This draft standard is based on national and international discussions. Your comments on this draft are invited and will assist in the preparation of the consequent standard.

For international standards, comments will be reviewed by the relevant UK national committee before sending the consensus UK vote and comments to the international committee, which will then decide appropriate action. If the international standard is approved, it is usual for the text to be published as a British Standard.

For national standards, comments will be reviewed by the relevant UK national committee and the resulting standards published as a British Standard.

UK Vote

Please indicate whether you consider the UK should submit a negative (with supporting technical reasons) or positive vote on this draft. Please indicate if you are aware of any reason why this draft standard should not be published as a British Standard.

Submission of Comments

- **Annotated drafts are not acceptable and will be rejected.**
- All comments should be submitted online at <https://standardsdevelopment.bsigroup.com/>. You will need to register in order to comment.

Template for comments and secretariat observations

Date: xx/xx/20xx	Document: ISO/DIS xxxx
------------------	------------------------

1	2	(3)	4	5	(6)	7
MB	Clause No./ Subclause No./Annex (e.g. 3.1)	Paragraph/Figure/ Table/Note	Type of comment	Commend (justification for change) by the MB	Proposed change by the MB	Secretariat observations on each comment submitted
	3.1	Definition 1	ed	Definition is ambiguous and needs clarifying.	Amend to read '...so that the mains connector to which no connection...'	
	6.4	Paragraph 2	te	The use of the UV photometer as an alternative cannot be supported as serious problems have been encountered in its use in the UK.	Delete reference to UV photometer.	

PROJECT NUMBER: IEC 63380-3 ED1	
DATE OF CIRCULATION: 2023-02-03	CLOSING DATE FOR COMMENTS: 2023-04-28
SUPERSEDES DOCUMENTS: 69/741A/NP, 69/751/RVN	

IEC TC 69 : ELECTRICAL POWER/ENERGY TRANSFER SYSTEMS FOR ELECTRICALLY PROPELLED ROAD VEHICLES AND INDUSTRIAL TRUCKS	
SECRETARIAT: Belgium	SECRETARY: Mr Peter Van den Bossche
OF INTEREST TO THE FOLLOWING COMMITTEES:	PROPOSED HORIZONTAL STANDARD: ☒ Other TC/SCs are requested to indicate their interest, if any, in this CD to the secretary.
FUNCTIONS CONCERNED: ☐ EMC ☐ ENVIRONMENT ☐ QUALITY ASSURANCE ☐ SAFETY	

This document is still under study and subject to change. It should not be used for reference purposes.

Recipients of this document are invited to submit, with their comments, notification of

- any relevant patent rights of which they are aware and to provide supporting documentation,
- any local regulations or technical reasons that may exist and should be considered should this proposal proceed, recognizing that failure to address such requirements could result in the need for new or existing "in some countries" clauses. See AC/22/2007.

TITLE:

<p>Local Charging station management systems and Local Energy Management Systems network connectivity and information exchange - Part 3 Communication Protocol and Cybersecurity Specific Aspects</p>

NOTE FROM TC/SC OFFICERS:

Copyright © 2023 International Electrotechnical Commission, IEC. All rights reserved. It is permitted to download this electronic file, to make a copy and to print out the content for the sole purpose of preparing National Committee positions. You may not copy or "mirror" the file or printed version of the document, or any part of it, for any other purpose without permission in writing from IEC.

CONTENTS

1		
2		
3	FOREWORD.....	7
4	INTRODUCTION.....	9
5	1 Scope.....	10
6	2 Normative references	10
7	3 Terms and definitions	11
8	4 Reader's Guide	14
9	4.1 General.....	14
10	5 SPINE Protocol	15
11	5.1 General.....	15
12	5.2 Architecture overview.....	15
13	5.2.1 General rules.....	15
14	5.2.2 Common data types.....	16
15	5.2.3 Address level details.....	20
16	5.3 SPINE Datagram	22
17	5.3.1 Introduction	22
18	5.3.2 Header	23
19	5.3.3 Payload	30
20	5.4 Communication modes.....	43
21	5.4.1 Simple communication mode	44
22	5.4.2 Enhanced communication mode	44
23	5.5 Functional commissioning	45
24	5.5.1 Detailed discovery	45
25	5.5.2 Destination list.....	59
26	5.5.3 Binding	62
27	5.5.4 Subscription	71
28	5.5.5 Use Case discovery.....	79
29	6 SHIP.....	82
30	6.1 Architecture overview.....	82
31	6.1.1 General	82
32	6.1.2 General considerations on Closing communication channels	83
33	6.1.3 SHIP node parameters	83
34	6.2 Registration	85
35	6.2.2 Successful registration	86
36	6.2.3 Registration details and recommendations (informative)	86
37	6.3 Reconnection.....	87
38	6.3.1 General	87
39	6.3.2 Reconnection details in case of changed key material (informative).....	88
40	6.4 Discovery.....	88
41	6.4.1 General	88
42	6.4.2 Service instance	88
43	6.4.3 Service name	89
44	6.4.4 Multicast DNS name	89
45	6.4.5 Recommendations for re-discovery.....	91
46	6.5 TCP	91

47	6.5.1	General	91
48	6.5.2	Limited connection capabilities	91
49	6.5.3	Online detection	92
50	6.5.4	TCP connection establishment.....	92
51	6.5.5	Retransmission timeout	93
52	6.6	TLS.....	93
53	6.6.1	General	93
54	6.6.2	Cipher suites	94
55	6.6.3	Maximum fragment length.....	94
56	6.6.4	TLS compression.....	95
57	6.6.5	Renegotiation	95
58	6.6.6	Session resumption	95
59	6.6.7	TLS extension for ECC	96
60	6.6.8	TLS probing.....	96
61	6.7	WebSocket	97
62	6.7.1	General	97
63	6.7.2	TLS dependencies	97
64	6.7.3	Opening handshake.....	97
65	6.7.4	Data framing.....	97
66	6.7.5	Connection keepalive	98
67	6.8	Message representation using JSON text format.....	98
68	6.8.1	Introduction	98
69	6.8.2	Definitions	99
70	6.8.3	Examples for each type	99
71	6.8.4	XML to JSON transformation	99
72	6.8.5	JSON to XML transformation	105
73	6.9	Key management.....	106
74	6.9.1	General	106
75	6.9.2	Certificates	107
76	6.9.3	SHIP node specific public key.....	111
77	6.9.4	Verification procedure.....	114
78	6.9.5	Symmetric key	120
79	6.9.6	SHIP node PIN	120
80	6.9.7	SHIP commissioning tool	121
81	6.9.8	QR Code	124
82	6.10	SHIP data exchange	127
83	6.10.1	Introduction	127
84	6.10.2	Terms in the context of SHIP Data Exchange	127
85	6.10.3	Protocol architecture / hierarchy	129
86	6.10.4	SHIP message exchange.....	131
87	6.11	Well-known protocolId.....	176
88	7	ECHONET Lite	176
89	7.1	General.....	176
90	Annex A (normative)	SHIP XSD	178
91	Bibliography		184
92	Figure 1 – Overview of transport protocols within IEC 63380-3		15
93	Figure 2 – PossibleOperationsType		19
94	Figure 3 – DeviceAddressType		19

95	Figure 4 – EntityAddressType	20
96	Figure 5 – FeatureAddressType	20
97	Figure 6 – SPINE datagram	22
98	Figure 7 – SPINE header	23
99	Figure 8 – SPINE payload	31
100	Figure 9 – Example of selectors part (extract) with entity address part	42
101	Figure 10 – Communication modes of SPINE devices A, B and C.	43
102	Figure 11 – Discovery example	45
103	Figure 12 – Hierarchy types	46
104	Figure 13 – Function Discovery Example over Feature Description	46
105	Figure 14 – nodeManagementDetailedDiscoveryData function overview, part 1	50
106	Figure 15 – nodeManagementDetailedDiscoveryData function overview, part 2:	
107	deviceInformation.description	51
108	Figure 16 – nodeManagementDetailedDiscoveryData function overview, part 3:	
109	entityInformation.description	51
110	Figure 17 – nodeManagementDetailedDiscoveryData function overview, part 4:	
111	featureInformation.description	52
112	Figure 18 – nodeManagementDestinationListData function overview, part 1	61
113	Figure 19 – nodeManagementDestinationListData function overview, part 2	61
114	Figure 20 – Binding request	64
115	Figure 21 – nodeManagementBindingRequestCall function overview	64
116	Figure 22 – nodeManagementBindingData function overview	66
117	Figure 23 – nodeManagementBindingDeleteCall function overview	68
118	Figure 24 – Subscription request	72
119	Figure 25 – nodeManagementSubscriptionRequestCall function overview	73
120	Figure 26 – nodeManagementSubscriptionData function overview	74
121	Figure 27 – nodeManagementSubscriptionDeleteCall function overview	77
122	Figure 28 – nodeManagementUseCaseData function	80
123	Figure 29 – Physical Connections in the Overall System	82
124	Figure 30 – SHIP Stack Overview	83
125	Figure 31 – Full TLS 1.2 Handshake with mutual authentication	94
126	Figure 32 – Quick TLS Handshake with Session Resumption	96
127	Figure 33 – Easy Mutual Authentication with QR-codes and Smart Phone	121
128	Figure 34 – QR Code Model 2, "low" ECC level, 0.33mm/Module, with SKI and PIN	126
129	Figure 35 – QR Code Model 2, "low" ECC level, 0.33mm/module, with all values	126
130	Figure 36 – QR Code Model 2, "low" ECC level, 0.33mm/module, with brainpoolP256r1	
131	SKI and brainpoolP384r1 SKI	127
132	Figure 37 – Protocol Architecture and Hierarchy	130
133	Figure 38 – CMI Message Sequence Example	134
134	Figure 39 – Connection State "Hello" Sequence Example Without Prolongation	
135	Request: "A" and "B" already trust each other; "B" is slower/delayed.	142
136	Figure 40 – Connection State "Hello" Sequence Example With Prolongation Request	143
137	Figure 41 – Connection State "Protocol Handshake" Message Sequence Example	149
138	Figure 42 – Connection State "PIN verification" Message Sequence Example (Begin)	159
139	Figure 43: ECHONET Lite frame format	177

140		
141	Table 1 – Structure of the SPINE datagram	22
142	Table 2 – cmdClassifier values and kind of messages for a message "M" and the scope	
143	of related acknowledgement messages.....	25
144	Table 3 – Structure of the SPINE header	29
145	Table 4 – Elements of the SPINE payload.....	31
146	Table 5 – Example table (template): This template is used in the subsequent sections	
147	for specific cmdOptions combinations. In this template, each "..." is just a placeholder.	34
148	Table 6 – Considered cmdOptions combinations for classifier "write"	35
149	Table 7 – Considered cmdOptions combinations for classifier "notify"	36
150	Table 8 – Considered cmdOptions combinations for classifier "read"	37
151	Table 9 – Considered cmdOptions combinations for classifier "reply"	38
152	Table 10 – Address path examples	41
153	Table 11 – Notify/response list of entities and their corresponding features with	
154	nodeManagementDetailedDiscoveryData	52
155	Table 12 – nodeManagementDetailedDiscoveryDataSelectors	57
156	Table 13 – Notify/response of DestinationList information with	
157	nodeManagementDestinationListData	61
158	Table 14 – Binding request with nodeManagementBindingRequestCall.....	64
159	Table 15 – nodeManagementBindingData holds list of binding entries	66
160	Table 16 – Remove Binding with nodeManagementBindingDeleteCall	69
161	Table 17 – Subscription request with nodeManagementSubscriptionRequestCall.....	73
162	Table 18 – nodeManagementSubscriptionData holds list of subscription entries	75
163	Table 19 – Remove subscription with nodeManagementSubscriptionDeleteCall	77
164	Table 20 – nodeManagementUseCaseData	80
165	Table 21 – SHIP Parameters Default Values	83
166	Table 22 – Mandatory Parameters in the TXT Record	89
167	Table 23 – Optional Parameters in the TXT Record	90
168	Table 24 – Mapping from the XSD Types to JSON Types.....	100
169	Table 25 – Transformation of a simple type	100
170	Table 26 – Mapping from the XSD compositors to JSON Types	101
171	Table 27 – Examples for XML and JSON representations.	102
172	Table 28 – Example transformation of several combined XSD item types.....	104
173	Table 29 – Example for JSON to XML transformation.....	106
174	Table 30 – Trust Levels	119
175	Table 31 – MessageType Values	132
176	Table 32 – Structure of SmeHelloValue of SME "hello" Message.	135
177	Table 33 – Structure of SmeProtocolHandshakeValue of SME "Protocol Handshake"	
178	Message.....	144
179	Table 34 – Structure of SmeProtocolHandshakeErrorValue of SME "Protocol	
180	Handshake Error" Message	145
181	Table 35 – Values of Sub-element "error" of messageProtocolHandshakeError	148
182	Table 36 – Structure of SmeConnectionPinStateValue of SME "Pin state" message.	150
183	Table 37 – Structure of SmeConnectionPinInputValue of SME "Pin input" message.....	151
184	Table 38 – Structure of SmeConnectionPinErrorValue of SME "Pin error" message.....	151

185	Table 39 – Values of Sub-element "error" of connectionPinError.....	157
186	Table 40 – Structure of MessageValue of "data" Message	160
187	Table 41 – Structure of SmeConnectionAccessMethodsRequestValue of SME "Access	
188	methods request" message.....	163
189	Table 42 – Structure of SmeConnectionAccessMethodsValue of SME "Access	
190	methods" message.	163
191	Table 43 – Structure of SmeConnectionCommissioningRequestValue of SME	
192	"commissioning request" message	166
193	Table 44 – Structure of SmeConnectionCommissioningResponseValue of SME	
194	"commissioning response" message	166
195	Table 45 – Structure of SmeConnectionKeyMaterialRequestValue of SME "key material	
196	request" message	167
197	Table 46 – Structure of SmeConnectionKeyMaterialValue of SME "key material"	
198	message	168
199	Table 47 – Structure of SmeConnectionKeyMaterialResponseValue of SME "key	
200	material response" message	169
201	Table 48 – Structure of SmeConnectionKeyMaterialDeleteValue of SME "key material	
202	delete" message	170
203	Table 49 – Structure of SmeConnectionKeyMaterialDeleteResponseValue of SME "key	
204	material delete response" message	171
205	Table 50 – Structure of SmeConnectionKeyMaterialStateValue of SME "key material	
206	state" message	172
207	Table 51 – Structure of SmeConnectionKeyMaterialStateResponseValue of SME "key	
208	material state response" message	173
209	Table 52 – Structure of SmeConnectionKeyMaterialStateRequestValue of SME "key	
210	material state request" message	173
211	Table 53 – Structure of SmeCloseValue of SME "close" Message.....	174
212		
213		

INTERNATIONAL ELECTROTECHNICAL COMMISSION

STANDARD INTERFACE FOR CONNECTING CHARGING POINTS AND/OR CHARGING STATIONS TO LOCAL ENERGY MANAGEMENT SYSTEMS

Part 3: Communication Protocol and Cybersecurity Specific Aspects

For rules on the drafting of the title, refer to the ISO/IEC Directives, Part 2:2021, [Clause 11](#).

The foreword is a mandatory element of the text.

For rules on the drafting of the foreword, refer to the ISO/IEC Directives, Part 2:2021, [Clause 12](#).

FOREWORD

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

IEC 63380 has been prepared by subcommittee PT63380: Local Charging station management systems, of 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

XX/XX/CD	XX/XX/RVD
----------	-----------

266
267 Full information on the voting for its approval can be found in the report on voting indicated in
268 the above table.

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

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

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

- 277 • reconfirmed,
- 278 • withdrawn,
- 279 • replaced by a revised edition, or
- 280 • amended.

281

INTRODUCTION

The expansion of renewable energy and the simultaneous reduction in conventional generation 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 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, a fundamental distinction must be made between the terms power 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. This allows the maximum load at the grid connection point to be reduced. Accordingly, priority must be given to local power management over, for example, optimization of operations and tariffs or desired charging plans.

In addition to the above-described goal of power management, the further goal of procurement- or tariff-optimized operation can be pursued within the performance limits specified by the infrastructure – controlled by the energy management system. Accordingly, a charging infrastructure should 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 requirements. Such coordination is particularly necessary if generating systems (e.g. solar system, combined heat and power plant) 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 and should be taken into account when developing a standardized interface for local power and energy management.

The aim of this document is to define a standard interface for connecting charging points and/or charging stations to local energy management system.

STANDARD INTERFACE FOR CONNECTING CHARGING POINTS AND/OR CHARGING STATIONS TO LOCAL ENERGY MANAGEMENT SYSTEMS

Part 3: Communication Protocol and Cybersecurity Specific Aspects

1 Scope

This document defines the secure information exchange between Local Charging Station Management Systems (LCSMS), which may be part of charging stations or implemented as separate local (or cloud-based) systems, and local Energy Management Systems/Customer Energy Managers (EMS/CEM) in homes and buildings.

It will specify use cases, the sequences of information exchange, the data models and the communication protocols to use and shall cover all aspects of local energy management of charging stations. It will be based on the needs of energy transfer aspects, information exchange and data models defined in the IEC 63110 series, ensuring interoperability of in-building appliances as well as charging stations on the one hand, and local EMS/CEM on the other.

This document 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. Neither does it cover the information exchange over any other interfaces defined in IEC 63110-1.

While IEC 63110-1 covers scenarios where there is a charging infrastructure backend, this document (IEC 63380) will cover cases where such a backend is not involved.

The series will consist of the following structure, describing the interface between LCSMS and local energy management systems from the LCSMS perspective.

- Part -1 General Requirements, Use Cases and abstract Messages
- Part -2 Specific Data Model Mapping
- Part -3 Communication Protocol and Cybersecurity Specific Aspects
- Part -4 Test Specifications

This part of IEC 63380 specifies the application of relevant transport protocols; in this case, SPINE (Smart Premises Interoperable Neutral-Message Exchange), SHIP (Smart Home IP), and ECHONET Lite.

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.

IETF RFC 793, Transmission Control Protocol

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

IETF RFC 6455, The WebSocket Protocol

IETF RFC 6763, DNS-Based Service Discovery

359 ISO/IEC 14543-4-3:2015 Information technology, Home Electronic Systems (HES) architecture — Part
360 4-3: Application layer interface to lower communications layers for network enhanced control devices of
361 HES Class 1

362 IEC 62394:2022, Service diagnostic interface for consumer electronics products and networks -
363 Implementation for ECHONET

364 IEC 63380-2: Specific Data Model Mapping, Local Charging station management systems and Local
365 Energy Management Systems network connectivity and information exchange

366 **3 Terms and definitions**

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

- 368 • IEC Electropedia: available at <http://www.electropedia.org/>

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

370 **3.1**

371 **CA**

372 **Certificate Authority**

373 **Certification Authority**

374 entity which can provide a digital signature for certificates

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

377 **3.2**

378 **Commissioning Tool**

379 <SHIP> instrument to establish the trust between different devices in the smart home installation, e.g.
380 distribute trustworthy credentials from some SHIP nodes to other SHIP nodes

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

384 Note 2 to entry: A manufacturer may also use their own solutions
385

386 **3.3**

387 **DNS**

388 Domain Name System,

389 [Source: IETF RFC 1035]

390 **3.4**

391 **DNS host name**

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

394 **3.5**

395 **DNS-SD**

396 Domain Name System – Service discovery

397 [Source: IETF RFC 6763]

398 **3.6**

399 **Factory Default** 400

setting that allows the user to reset the SHIP node to the as-new condition; this means that all data that has been provided and stored by the SHIP node during its operation time SHALL be deleted

3.7

IANA

Internet Assigned Numbers Authority

3.8

IP

Internet Protocol

3.9

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

3.10

M/O/NV/C

abbreviations which refer to:

1. M = mandatory

2. O = optional

3. NV = not valid

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

3.11

Numerical representation

written system for expressing numbers. For example, 0xab represents a decimal value of 171

3.12

PIN

Personal Identification Number

specification which makes use of a PIN as secret for SHIP specific verification procedures

3.13

PKI

Public Key Infrastructure

3.14

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 has to be used to trigger this event. A push button event may also be generated by other means, e.g. 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.

3.15

QR Code

the term "QR Code" is a registered trademark of DENSO WAVE INCORPORATED; "QR Code" is the short form for "Quick Response Code" and used for efficient encoding of data into a small graphic

Note to entry: Among others, the international standard ISO/IEC 18004:2015 specifies the encoding of QR code symbols.

3.16**RFC**

request for comments

3.17**SHIP**

Smart Home Internet Protocol

Note 1 to entry: This term is used throughout this document to refer to the described communication protocol

3.18**SHIP ID**

identification, which is used to uniquely identify a SHIP node, e.g. in its service discovery, and which is present in the mDNS/DNS-SD local service discovery

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

Note to entry 2: See section 6.4.

3.19**SHIP Client**

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

3.20**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.

3.21**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

3.22**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 may have more than one logical SHIP node. In this case, each SHIP node MUST use distinct ports (e.g. a physical device provides 3 open ports with 3 different SHIP services).

3.23**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

3.24**SKI**

Subject Key Identifier

identifying certificates that contain 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.

3.25

SPINE

Smart Premises Interoperable Neutral message Exchange

Technical Specification of EEBus Initiative e.V

3.26

Trusted SHIP Node

term only applicable from a specific SHIP node point of view

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

3.27

UCS

Universal Character Set

3.28

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

3.29

Web server-based SHIP node

SHIP node that is hosted by a web server

3.30

WiFi

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

4 Reader's Guide

4.1 General

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. Figure 1 shows the use of the transport protocols within this document.

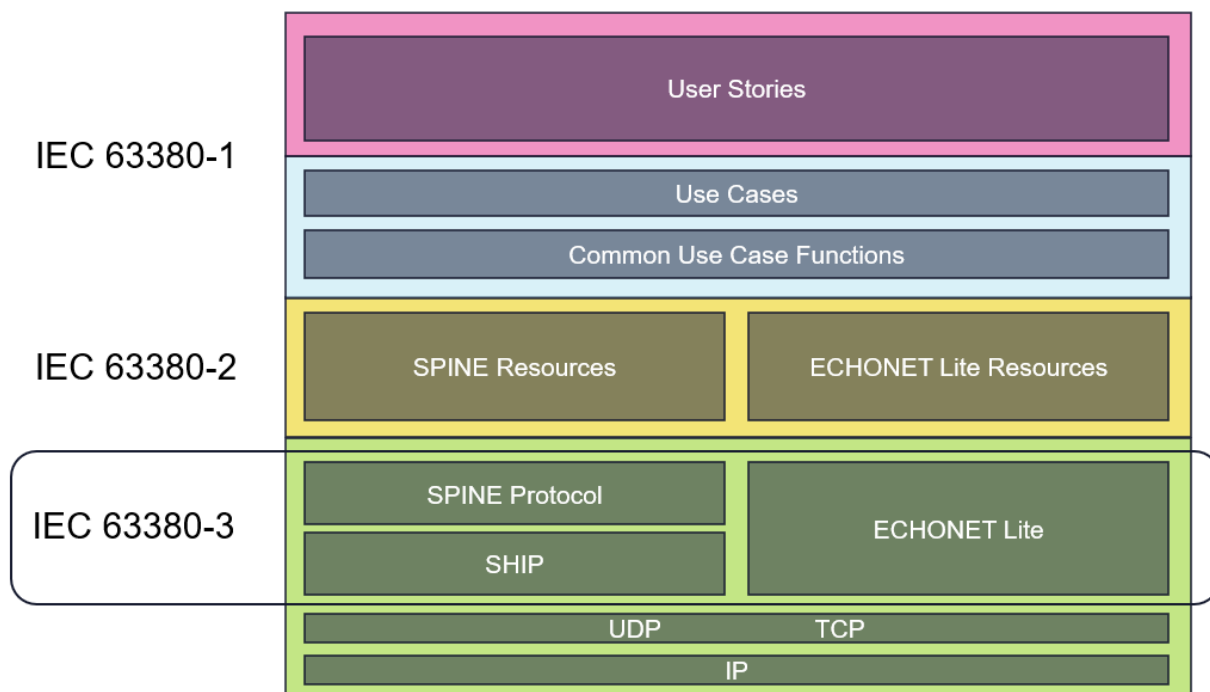


Figure 1 – Overview of transport protocols within IEC 63380-3

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.

The next sub-clauses discuss 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.

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 section 5.5.3) and subscription (see section 5.5.4) mechanisms requires imposing some restrictions, i.e. the definition of an architecture. Thus, the following rules 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.
Remark: This is explained in more detail in section 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 section 5.5.1) of entity 0 SHALL be

- "DeviceInformation". The so-called "featureType" (see section 5.5.1) of feature 0 of entity 0 SHALL be "NodeManagement".
3. This document does not consider any non-primary NodeManagement instance (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 above). Thus, NodeManagement instances include proper information on itself or other NodeManagement instances as well, according to the rules given above.
 5. On each feature there SHALL be at maximum one class implemented with regard to the features primary functionality. I.e. it is NOT permissive to offer more than one class on a single feature (see also section 5.3.3).
 6. A feature on a device is assigned a "functional role". This role is EITHER "server" OR "client" OR "special". Please note 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 section 5.3.3.3). I.e. the device can notify changes or send this data as reply upon request. It may 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 The rules above 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 (esp. using the "trust level" concept).

5.2.2 Common data types

The majority of data types use data types defined by W3C [5]. In this document these data types are identified by the namespace prefix "xs:" ("xs:boolean", e.g.).

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 the following.

5.2.2.1 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 SPINE edition.

5.2.2.2 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.

5.2.2.3 DescriptionType

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

The string-length SHOULD NOT be longer than 4096 characters. If it is longer, the sender SHALL consider the possibility that the receiver will shorten the string to 4096 characters.

5.2.2.4 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.

5.2.2.5 EnumExtendType

Used for extending enumeration with vendor specific ones. Some enumerations may 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.

5.2.2.6 RoleType

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

5.2.2.7 FeatureGroupType

Use 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.

5.2.2.8 DeviceTypeType

Union of:

- DeviceTypeEnumType (see section 5.2.2.9)
- EnumExtendType (see section 5.2.2.5)

5.2.2.9 DeviceTypeEnumType

The type of the device.

5.2.2.10 EntityTypeType

Union of:

629 - EntityTypeEnumType (see section 5.2.2.11)

630 - EnumExtendType (see section 5.2.2.5)

631 **5.2.2.11 EntityTypeEnumType**

632 The type of the entity.

633

634 **5.2.2.12 FeatureTypeType**

635 Union of:

636 - FeatureTypeEnumType (see section 5.2.2.13)

637 - EnumExtendType (see section 5.2.2.5)

638 **5.2.2.13 FeatureTypeEnumType**

639 The type of the feature.

640

641 **5.2.2.14 FunctionType**

642 Union of:

643 - FunctionEnumType (see section 5.2.2.15)

644 - EnumExtendType (see section 5.2.2.5)

645 **5.2.2.15 FunctionEnumType**

646 A list of all SPINE functions.

647

648 **5.2.2.16 PossibleOperationsClassifierType**

649 If the element partial is used, filtering the further content is permitted. Otherwise only full operations
650 (no filtering) are allowed. A complexType, containing the element partial (type: ElementTagType, see
651 section 5.2.2.1).

652

653 **5.2.2.17 PossibleOperationsReadType**

654 If read operations are permitted, this element is set. Elements from the base type can be used. A
655 complexType with the base type PossibleOperationsClassifierType (see section 5.2.2.16).

656

657 **5.2.2.18 PossibleOperationsWriteType**

658 If write operations are permitted, this element is set. Elements from the base type can be used. A
659 complexType with the base type PossibleOperationsClassifierType (see section 5.2.2.16).

660

661 **5.2.2.19 PossibleOperationsType**

662 Defines which operations on a function are allowed as shown in Figure 2. A complexType, containing
663 the elements read (type: PossibleOperationsReadType, see section 5.2.2.17) and write (type:
664 PossibleOperationsWriteType, see section 5.2.2.18).

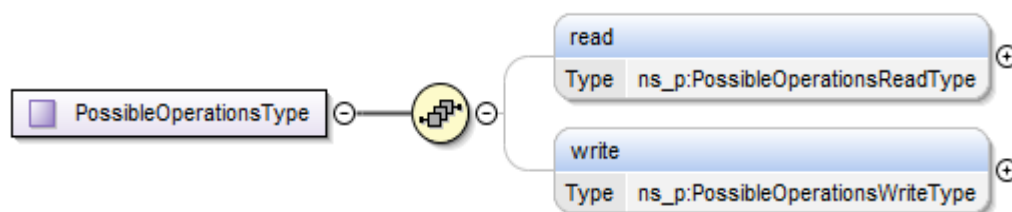


Figure 2 – PossibleOperationsType

5.2.2.20 MaxResponseDelayType

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

5.2.2.21 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 seconds (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".

5.2.2.22 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}]+`

5.2.2.23 AddressEntityType

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

5.2.2.24 AddressFeatureType

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

5.2.2.25 DeviceAddressType

Address type including only the device part, as shown in Figure 3. A complexType with the element device (type: AddressDeviceType, see section 5.2.2.22).

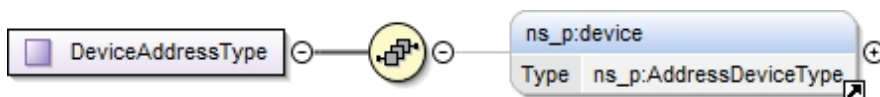


Figure 3 – DeviceAddressType

5.2.2.26 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 section 5.2.2.25) and a further list of element entity (type: AddressEntityType, see section 5.2.2.23).

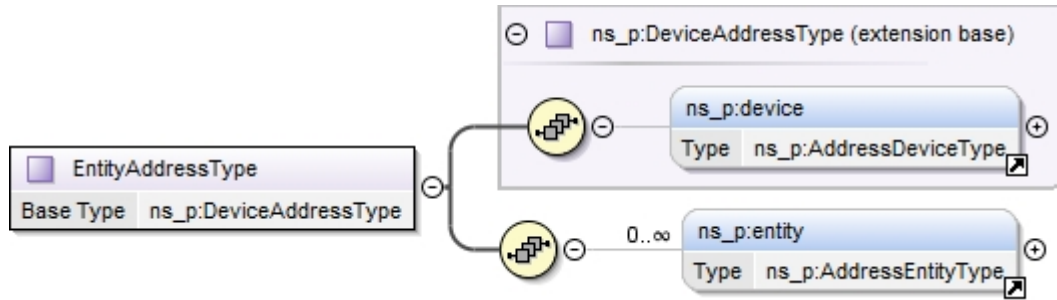


Figure 4 – EntityAddressType

5.2.2.27 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 section 5.2.2.26) and a further element feature (type: AddressFeatureType, see section 5.2.2.24).

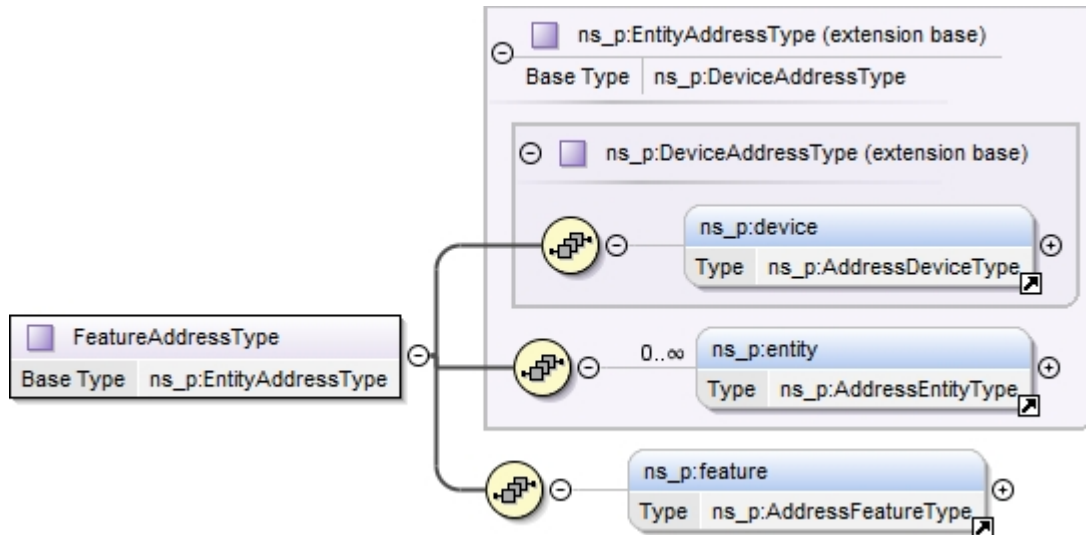


Figure 5 – FeatureAddressType

5.2.3 Address level details

The concept of the miscellaneous address levels as described in section 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 (remark: "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)
```

```

722         |          +--- feature 1 (child of "someDevice"/entity 1)
723         |          +--- feature 4 (child of "someDevice"/entity 1)
724         +--- entity 4          (child of "someDevice")
725         |          +--- feature 1 (child of "someDevice"/entity 4)
726 (*1): child of "someDevice"/entity 1/entity 4
727 (*2): child of "someDevice"/entity 1/entity 5

```

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

730 Starting at the top, the full address paths are (note that this is an example to describe the SPINE
 731 address level concept and that the following lines DO NOT state valid SPINE addresses; the correct
 732 usage of SPINE addresses is described in section 5.3):

- 733 1. device "someDevice"
- 734 2. device "someDevice" / entity 0
- 735 3. device "someDevice" / entity 0 / feature 0
- 736 4. device "someDevice" / entity 1
- 737 5. device "someDevice" / entity 1 / entity 4
- 738 6. device "someDevice" / entity 1 / entity 4 / feature 7
- 739 7. device "someDevice" / entity 1 / entity 5
- 740 8. device "someDevice" / entity 1 / entity 5 / feature 1
- 741 9. device "someDevice" / entity 1 / entity 5 / feature 7
- 742 10. device "someDevice" / entity 1 / feature 1
- 743 11. device "someDevice" / entity 1 / feature 4
- 744 12. device "someDevice" / entity 4
- 745 13. device "someDevice" / entity 4 / feature 1

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

749 It shall now be explained how this is modelled in the SPINE XSDs and XMLs. In the XSDs an element
 750 "entity" is often defined with attributes

```

751         minOccurs="0" maxOccurs="unbounded"

```

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

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

```

765         <device>someDevice</device>
766         <entity>1</entity>
767         <entity>4</entity>
768         <feature>7</feature>

```

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

```

771         ... entity (list)

```

As already explained above, 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 section 5.3.3.4) it needs to be emphasized that a SPINE address as information unit shall always be seen as an indivisible information unit that needs to be transmitted completely (with the exception that the "device" part might 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 section 5.3.3.4.1.

NOTE A SPINE address might also be used for other purposes than an information unit – see esp. section 5.3.3.4.7.2 for its use as so-called selectors.

5.3 SPINE Datagram

5.3.1 Introduction

5.3.1.1 General information

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" could 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.

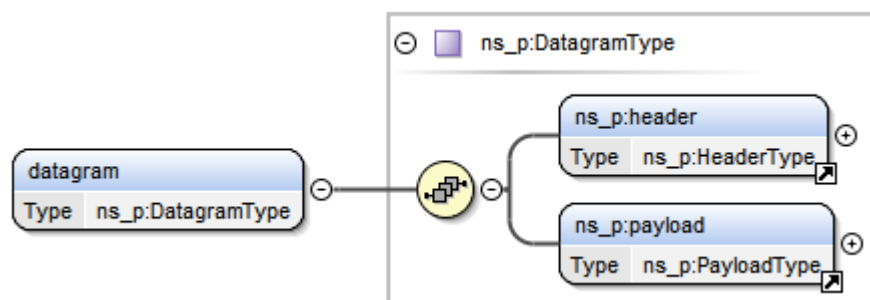


Figure 6 – SPINE datagram

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 section 5.3.2.7.
datagram. payload	M	The payload element SHALL be present. For sub-elements of the payload see section 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 information

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.

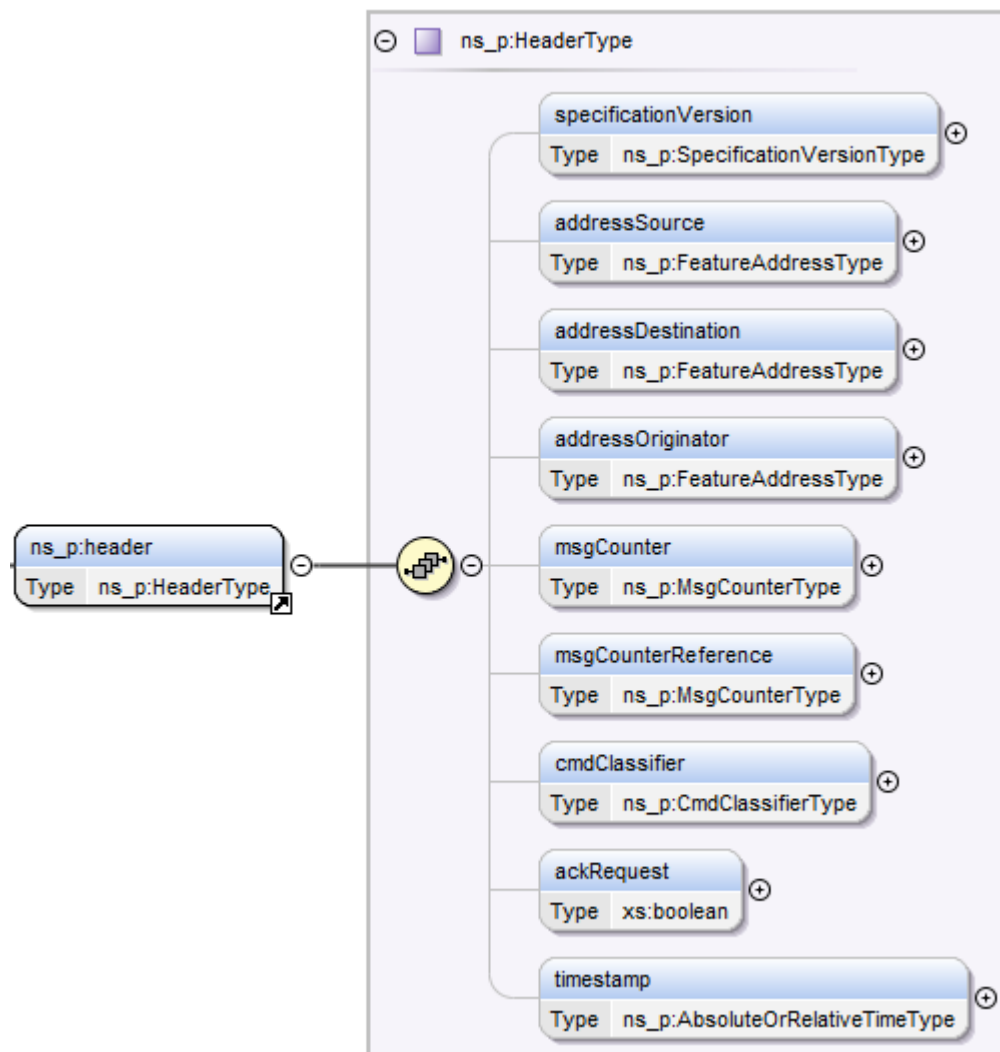


Figure 7 – SPINE header

5.3.2.2 Address information

5.3.2.2.1 addressSource and addressDestination

The elements addressSource and addressDestination are defined with their child elements and purpose in Table 3. addressSource corresponds to the unique address path (see section 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 may 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 below) apply in any case. Furthermore, additional rules apply dependent on the communication mode:

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

In general, the use of "addressSource. device" and "addressDestination. device" REQUIRES to ensure 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 section 5.5.1.1.4.1.

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.

Additional rules in case of "simple communication mode":

The following rules apply in case of a "simple communication mode" (see section 5.4.1). 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").

Additional rules in case of "enhanced communication mode":

The following rules apply in case of an "enhanced communication mode" (see section 5.4.2). 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" needs to be set properly.

NOTE If "addressOriginator" is used as described above, the elements "addressSource. device", "addressDestination. device" and "addressOriginator. device" SHALL be set!

5.3.2.3 Message counter

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.

5.3.2.3.1 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 may 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!

Implementation advice: 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 1000 and uses the result for its next message. From then on, every 1000th created message the device copies the msgCounter value into the "stored msgCounter value".

Please note: 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 may be easier for implementation in embedded devices whereas web servers may scale better with values that are communication partner specific.

5.3.2.3.2 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".

This subclause also uses the terms "dedicated reply", "dedicated data", or "dedicated response". The SPINE protocol defines a number of (sub-)classes with (SPINE) functions. A device "A" may be the "owner" of a specific function instance (i.e. it has the role "server" at the function's feature address). Another device "B" may 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" may lead to the return of a related "acknowledgement message" (see section 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 the scope of related acknowledgement messages

cmdClassifier of message "M"	Kind of message "M"	Scope of related acknowledgement message "N" (see section 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

912

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

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

920 The different values of cmdClassifier are used for the following operations:

921 **read**

922 This denotes a "read operation" from the sender (addressSource) to the recipient
923 (addressDestination). The recipient is considered the owner of the proper information (i.e. function).
924 The recipient SHALL respond with a dedicated "reply" if the read operation is valid and can be
925 processed regularly, i.e. without any failure. Otherwise the recipient SHALL respond with an
926 "acknowledgement message" (see section 5.3.2.5.2) where "errorNumber" is set to a different value
927 than "0". Please note that the obligation to create a "reply" or "error indication" is independent from the
928 value of the element "ackRequest" of the header. More precisely, the acknowledgement request (see
929 section 5.3.2.5.1) of a received "read operation" SHALL NOT be evaluated.

930 **reply**

931 This denotes a "reply operation" from the owner of the (assumed) proper information (addressSource)
932 to the recipient (addressDestination). A reply SHALL be created according to the rules given for
933 cmdClassifier value "read". It SHALL NOT be used in any other case. The recipient of a "reply
934 operation" SHALL evaluate the acknowledgement request of the message according to section
935 5.3.2.5.1. Please note that a proper acknowledgement message just denotes a transmission success
936 or error of the "reply operation".

937 **notify**

938 This denotes a notification ("notify operation") from the owner of the proper information
939 (addressSource) to the recipient (addressDestination). In contrast to a dedicated "reply" upon a "read"
940 message, a "notify" message is created autonomously by the owner of the proper information, i.e.
941 without the need of a received "read" message. A typical example for the creation of a "notify"
942 message is the notification of a value change in a feature that is subscribed by the recipient. The
943 recipient of a "notify operation" SHALL evaluate the acknowledgement request of the message
944 according to section 5.3.2.5.1. Please note that a proper acknowledgement message just denotes a
945 transmission success or error of the "notify operation".

946 **write**

947 This denotes a replacement or modification instruction ("write operation", "full" or "restricted") from the
948 sender (addressSource) to the recipient (addressDestination). The recipient is considered the owner
949 of the information (SPINE class function) that shall be replaced or modified according to the
950 instruction. The recipient of a "write operation" SHALL evaluate the acknowledgement request of the
951 message according to section 5.3.2.5.1.

952 **call**

953 This denotes an instruction ("call operation") from the sender (addressSource) to the recipient
954 (addressDestination) to trigger a specific action at the recipient. A "call operation" can be used to
955 exchange information where the "ownership concept" of the other classifiers usually does not apply.
956 The recipient of a "call operation" SHALL evaluate the acknowledgement request of the message
957 according to section 5.3.2.5.1.

result

This message classifier is used for so-called "application acknowledgement messages" (also called "result messages"; see section 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 section 5.3.2.5.2) if all of the following conditions are fulfilled:

1. The received message "M" belongs to an operation that REQUIRES the evaluation of the acknowledgement request according to section 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 section 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 section 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 section 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:

- 1005 1. A value of "0" SHALL be used for "positive acknowledgement".
- 1006 2. Every other value denotes a "negative acknowledgement".

1007 In addition, the element "description" of "resultData" MAY be present and filled with a readable
1008 information.

1009 In case a "result message" cannot be sent within time according to "defaultMaxResponseDelay" or
1010 "maxResponseDelay" described in section 5.3.2.5.3, please refer to section 5.3.2.5.3.

1011 Please note: This subclause does not specify the circumstances when an acknowledgement message
1012 is to be sent or not. For this, please see sections 5.3.2.4 and 5.3.2.5.1.

1013

1014 **5.3.2.5.3 Delayed application response**

1015 For each feature server a device needs some time to generate a proper response (which could be a
1016 message with "reply" classifier or "result" classifier) upon a received request. To have an interoperable
1017 concept for devices to know how long to wait for a response, a timeout mechanism with "maximum
1018 response delay" is used.

1019 The duration specified by "maximum response delay" only relates to the immediate access to the
1020 communication interface of a device. This means that latencies from communication channels are not
1021 covered by "maximum response delay".

1022 NOTE Each communication channel has some latency. This latency depends on the kind of the
1023 communications technology itself as well as on the environment (especially wireless technologies often suffer
1024 from poor coverage or too many participants sharing the available bandwidth). As specified above, this kind of
1025 latency is independent from the "maximum response delay".

1026 There are two timeout levels to derive the "maximum response delay" for a given server feature:

- 1027 1. By default, the "maximum response delay" is equal to defaultMaxResponseDelay. The
1028 defaultMaxResponseDelay SHALL be 10 seconds.
- 1029 2. If the maxResponseDelay element within the feature description of the detailed discovery (see
1030 section 5.5.1.2) is set AND the value of the maxResponseDelay element is larger than zero
1031 seconds, the value of the maxResponseDelay element SHALL be applied for this feature as
1032 "maximum response delay" (i.e. instead of defaultMaxResponseDelay).

1033 Recommendations on the value of maxResponseDelay:

- 1034 1. If a feature server will usually or frequently need more time than defaultMaxResponseDelay to
1035 generate proper replies, it SHOULD set the maxResponseDelay element in the feature description
1036 of the detailed discovery (see section 5.5.1.2) to a duration that the feature server can almost
1037 always fall short of for the generation of proper replies.
- 1038 2. If a feature server will almost always need less time than defaultMaxResponseDelay to generate
1039 proper replies, it MAY set the maxResponseDelay element in the feature description of the
1040 detailed discovery (see section 5.5.1.2) to a duration that the feature server can almost always fall
1041 short of for the generation of proper replies.

1042 Rules on the use of "maximum response delay":

- 1043 1. An implementation that complies with this or a subsequent version SHALL be able to handle
1044 response delays of at least defaultMaxResponseDelay.
- 1045 2. A feature client MAY use "maximum response delay" for the detection of a response-timeout even
1046 if "maximum response delay" is shorter than defaultMaxResponseDelay.
- 1047 3. An implementation SHOULD consider the communications technology specific latency carefully
1048 before it begins with the detection of a timeout to an expected response.

1049

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 below.

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 section 5.2.2)	M	The version of the SPINE data model applicable to the XML. SHALL be present.
datagram. header. addressSource	FeatureAddressType (see section 5.2.2)	M	The address of the submitter of this XML. SHALL be present.
datagram. header. addressSource. device	AddressDeviceType; see "Device address" in section 5.5.1.1.1	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 section 5.3.2.2.1 for details.
datagram. header. addressSource. entity (list)	AddressEntityType (see section 5.2.2)	M	Entity part(s) of the source address. SHALL be present. See also section 5.2.3, esp. concerning restrictions on the supported depth.
datagram. header. addressSource. feature	AddressFeatureType (see section 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 section 5.5.1.1.1	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 section 5.3.2.2.1 for details.
datagram. header. addressDestination. entity (list)	AddressEntityType (see section 5.2.2)	M	Entity part(s) of the destination address. SHALL be present. See also section 5.2.3, esp. concerning restrictions on the supported depth.
datagram. header. addressDestination. feature	AddressFeatureType (see section 5.2.2)	M	Feature part of the destination address. SHALL be present.
datagram. header. addressOriginator	FeatureAddressType (see section 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 section 5.3.2.2.2.
datagram. header. addressOriginator. device	AddressDeviceType; see "Device address" in section 5.5.1.1.1	M	Device part of the originator address. SHALL be present.
datagram. header. addressOriginator. entity (list)	AddressEntityType (see section 5.2.2)	M	Entity part(s) of the originator address. SHALL be present. See also section 5.2.3, esp. concerning restrictions on the supported depth.
datagram. header. addressOriginator. feature	AddressFeatureType (see section 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 section 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 section 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. May be present. If absent, the default value "false" applies. See section 5.3.2.5.1 for details.
datagram. header. timestamp	AbsoluteOrRelativeTimeType (see section 5.2.2)	O	The timestamp of the creation of this SPINE message. May be present. See section 5.3.2.6.

1062

1063 5.3.3 Payload

1064 5.3.3.1 General information

1065 Within the element "payload", a single "command" can be placed (broadly speaking; the data model is
 1066 also prepared for future extensions, but this is not considered further in detail in this document). It permits
 1067 or even requires the presence of additional elements to express/identify a functionality, as shown in
 1068 Figure 8. Details are discussed in the subsequent sections.

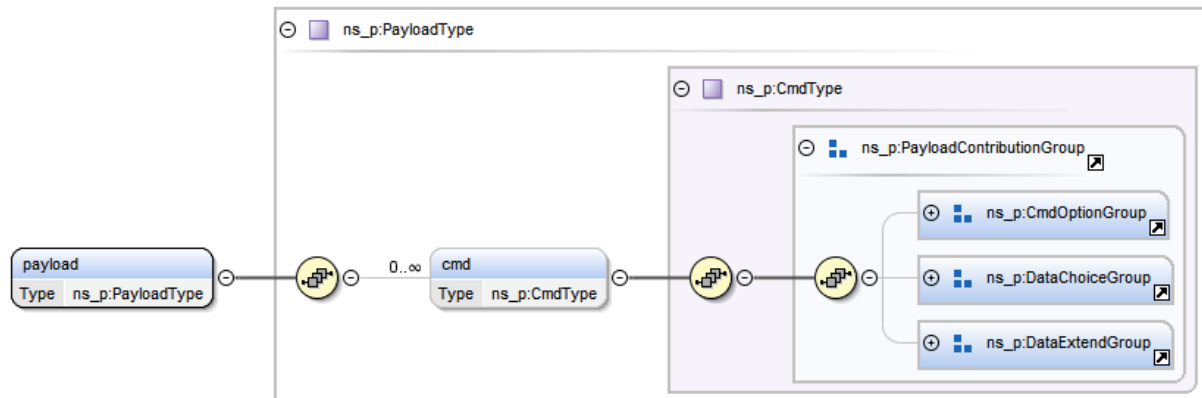


Figure 8 – SPINE payload

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. NOTE 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 section 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 section 5.3.3.4.
datagram. payload. cmd. filter		0..unbounded	Identifies the content that shall be restricted (see section 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 section 5.3.3.4. NOTE 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.
datagram. payload. cmd. filter. cmdControl. partial		O	Denotes the restriction is a "partial" operation.

datagram. payload. cmd. filter. <SELECTORS>		0..unbounde d	"<SELECTORS>" is a placeholder for SPINE class specific "Selectors" definitions. E.g. for a SPINE function "X" there might be a specific "XSelectors" 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 "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 might 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. manufacturerSpecific Extension	xs:hexBinary	O	Can be used to extend a given function with proprietary data. Please note it shall only be used together with a function. MAY be present.
datagram. payload. cmd. lastUpdateAt	AbsoluteOrRel ativeTimeTyp e (see section 5.2.2)	O	Can be used in case an implementation caches data of the actual node. MAY be present.

1075

1076 On each feature there SHALL be at maximum one class implemented with regard to the feature's
1077 primary functionality.

1078 The general XSD-based SPINE payload definition permits the presence of multiple "cmd" instances
1079 and therefore multiple functions in theory. However, within this version of the protocol each instance of
1080 a "payload" element SHALL contain exactly one "cmd" instance.

1081

1082 5.3.3.3 Ownership

1083 The combination of functions and classifiers/operations of the SPINE data model should be applied
1084 considering the concept of "ownership". The concept of "ownership" is directly linked to the SPINE role
1085 concept. The "ownership" of data is always defined on SPINE feature level.

1086 The "owner" of data (functions) may

- 1087 • notify its own data,
- 1088 • reply a request (i.e. received read operation) with its own data,

- 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

NOTE 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.

Please note 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.

Please note: The following subsections show 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 may support only specific kinds of restricted function exchange. Please also consider section 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". Section 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 section 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.

Remark: A featureType may 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 esp. useful to describe dedicated changes of the device's SPINE function.

On the cardinality of "filter":

As shown in Table 4, the element "datagram. payload. cmd. filter" may 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 section 5.3.3.4.2 where a "delete" as well as a "partial" part are used (functional details on these parts are explained in section 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 the following subsections 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) belong to a "filter" element where "cmdControl" is set to "partial". The "payload" example above corresponds to the penultimate row of Table 5.

Table 5 – Example table (template): This template is used in the subsequent sections for specific cmdOptions combinations. In this template, each "..." is just a placeholder.

filter with "delete"			filter with "partial"				
cmdControl	<ELEMENTS>	<SELECTORS>	cmdControl	<ELEMENTS>	<SELECTORS>	<FUNCTION>	Explanation and rules
-	-	-	partial	-	-	X	...
...	...	...	...	...	...	...	...
delete	-	X		-	-	dc	...
...	...	...	...	...	...	...	...
delete	(X)	(X)	partial	-	(X)	X	...
-	-	-	-	-	-	X	...

On the application of RFE to lists:

Within the RFE concept, two types of lists need to 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 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. This is detailed in the following sections.

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.

NOTE 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 section 5.3.3.4.7 for details.

NOTE 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 the following sections.

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"				Explanation and rules
cmdControl	<ELEMENTS>	<SELECTORS>	cmdControl	<ELEMENTS>	<SELECTORS>	<FUNCTION>	
-	-	-	partial	-	-	X	<FUNCTION> contains function parts to add or modify. Additionally, <FUNCTION> may 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 described above. Afterwards <FUNCTION> and filter with cmdControl "partial" SHALL be applied according to the "partial" combinations described above.
-	-	-	-	-	-	X	Full write.
"X" means a proper instance is present and not empty in the message. "(X)" means a proper instance may 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.							

1199

1200 In this document, at maximum one "delete" filter and at maximum one "partial" filter SHALL be used in
1201 one command. If both filters are present, the "delete" filter SHALL be present before the "partial" filter
1202 in the command.

1203 In general, a write operation with restricted function exchange SHALL ONLY be executed by a server
1204 if it can execute the received operation completely.

1205 5.3.3.4.3 "notify" cmdOptions combinations

1206 A "notify" command is very similar constructed like a "write" command, because a notify is mostly used
1207 to communicate what has changed after a write process. Be it an external write or internal change,
1208 both can be viewed as write processes. Therefore, the "notify" cmdClassifier permits also the same
1209 combinations as the "write" cmdClassifier. Please consider the details provided in section 5.3.3.4.2.

1210 Table 7 shows which kind of cmdOptions combinations can be considered for the use with classifier
1211 "notify".

1212 **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> may 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 described above. Afterwards <FUNCTION> and filter with cmdControl “partial” were applied according to the “partial” combinations described above.
-	-	-	-	-	-	X	Full notify.
“X” means a proper instance is present and not empty in the message. “(X)” means a proper instance may 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.							

1213

1214 In this document, at maximum one “delete” filter and at maximum one “partial” filter SHALL be used in
 1215 one command. If both filters are present, the “delete” filter SHALL be specified before the “partial” filter
 1216 in the command order.

1217

1218 5.3.3.4.4 “read” cmdOptions combinations

1219 Table 8 shows which kind of cmdOptions combinations can be considered for the use with classifier
 1220 “read”.

1221 **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.				

1222

1223 5.3.3.4.5 "reply" cmdOptions combinations

1224 Table 9 shows which kind of cmdOptions combinations can be considered for the use with classifier
1225 "reply".

1226 **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> may 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.				

1227

1228 A server MAY ignore unsupported cmdOption combinations and then replies with more than the
1229 requested parts instead. If the server does not support cmdOptions with "read" at all, it SHALL
1230 respond with a full reply.

1231 5.3.3.4.6 <FUNCTION> identifiers - Implicit list item selection

1232 Identifiers are used to select specific list entries directly in the <FUNCTION> (element
1233 "datagram.payload.cmd.<FUNCTION>" of Table 4). The approach requires that list items can be
1234 identified in a unique way.

1235 If a featureType specifies identifiers and their use, then the values of identifiers in a (full or restricted)
1236 instance of "<FUNCTION>" serve implicitly for the identification of the proper list item.

1237 The read request belonging to this example can be found in section 5.3.3.4.7. As requested, device
1238 "A" responds with this XML:

1239 <datagram>
1240 <header>
1241 ...

```

1242         <cmdClassifier>reply</cmdClassifier>
1243     </header>
1244     <payload>
1245         <cmd>
1246             <function>measurementListData</function>
1247             <filter>
1248                 <cmdControl>
1249                     <partial/>
1250                 </cmdControl>
1251             </filter>
1252             <measurementListData>
1253                 <measurementData>
1254                     <measurementId>5</measurementId>
1255                     <valueType>minValue</valueType>
1256                     <timestamp>2015-07-14T15:00:00.0Z</timestamp>
1257                     <value>
1258                         <number>-173</number>
1259                         <scale>-1</scale>
1260                     </value>
1261                     <evaluationPeriod>
1262                         <startTime>2015-07-14T10:00:00.0Z</startTime>
1263                         <endTime>2015-07-14T15:00:00.0Z</endTime>
1264                     </evaluationPeriod>
1265                     <valueSource>measuredValue</valueSource>
1266                     <valueTendency>rising</valueTendency>
1267                     <valueState>normal</valueState>
1268                 </measurementData>
1269             </measurementListData>
1270         </cmd>
1271     </payload>
1272 </datagram>

```

1273 In this example it is assumed the device keeps no further history of "minValue" values, i.e. it keeps just
1274 the latest "minValue".

1275 The following rules apply if cmdOptions are used:

- 1276 1. <FUNCTION> identifiers SHALL only be used for "partial" write/notify/reply commands.
- 1277 2. If <FUNCTION> identifiers are used, it is not allowed to use <SELECTORS> in a filter with
1278 cmdContol "partial".
- 1279 3. If <FUNCTION> contains at least one identifier, ALL list items included in <FUNCTION> SHALL
1280 have complete identifiers. I.e. it is not valid to have a list item without identifier and another list
1281 item with identifier within a <FUNCTION> instance.

1282

1283 5.3.3.4.6.1 Identifier hierarchy and completeness of list identifiers

1284 SPINE class functions with multiple identifiers usually assign the identifiers a "natural hierarchy" (e.g.
1285 an identifier "xId" serves as some kind of "parent identifier" for an identifier "yId"; a common example
1286 could be a postal address where the identifier "city" is a "parent identifier" of the identifier "street"). A
1287 featureType can explicitly define which identifiers are used and which hierarchy applies. If the
1288 featureType does not specify identifiers, the default identifiers of the underlying class apply and the
1289 hierarchy follows the order stated in the identifier section of the class description.

1290 The identifiers of a list item are complete if no identifier needs to be added to identify the target list
1291 item uniquely.

1292 Among others, identifiers permit "finding" or choosing specific list entries of a function containing a
1293 large list. An example is shown in section 5.3.3.4.7.

1294

5.3.3.4.7 <SELECTORS> – Explicit list item selection

This subclause describes the use of element "datagram.payload.cmd.filter.<SELECTORS>" of Table 4.

5.3.3.4.7.1 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. Please note that the latter (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 The above-mentioned requirement that all referenced child elements need to 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 might even contain a "nested list", and a proper <SELECTORS> might 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 was 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, e.g., 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/notifying the same values in/for 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.

A brief example for a "read operation" shall be given: In this example 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>
```

```

1345         <cmd>
1346             <function>measurementListData</function>
1347             <filter>
1348                 <cmdControl>
1349                     <partial/>
1350                 </cmdControl>
1351                 <measurementListDataSelectors>
1352                     <measurementId>5</measurementId>
1353                     <valueType>minValue</valueType>
1354                 </measurementListDataSelectors>
1355             </filter>
1356         </measurementListData/>
1357     </cmd>
1358 </payload>
1359 </datagram>

```

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

5.3.3.4.7.2 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, esp. due to the intrinsic list structure of "entity" address parts it shall be emphasized that still only exact matches of an "entity" address part of a selectors with an "entity" address part within a function are considered as valid matches. An example shall illustrate this:

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

```

1371 "someDevice"
1372 ...
1373     +--- entity 1
1374     |         +--- entity 4 (child of "someDevice"/entity 1)
1375     |         |         +--- feature 7 (*1)
1376     ...
1377     +--- entity 4 (child of "someDevice")
1378     |         +--- feature 1 (child of "someDevice"/entity 4)
1379
1380 (*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 above 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:46925_someDevice</device>
A2	device "someDevice" / entity 1	<device>d: i:46925_someDevice</device> <entity>1</entity>
A3	device "someDevice" / entity 1 / entity 4	<device>d: i:46925_someDevice</device> <entity>1</entity> <entity>4</entity>
A4	device "someDevice" / entity 1 / entity 4 / feature 7	<device>d: i:46925_someDevice</device> <entity>1</entity> <entity>4</entity> <feature>7</feature>
A5	device "someDevice" / entity 4	<device>d: i:46925_someDevice</device> <entity>4</entity>

A6	device "someDevice" / entity 4 / feature 1	<device>d: i:46925_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):

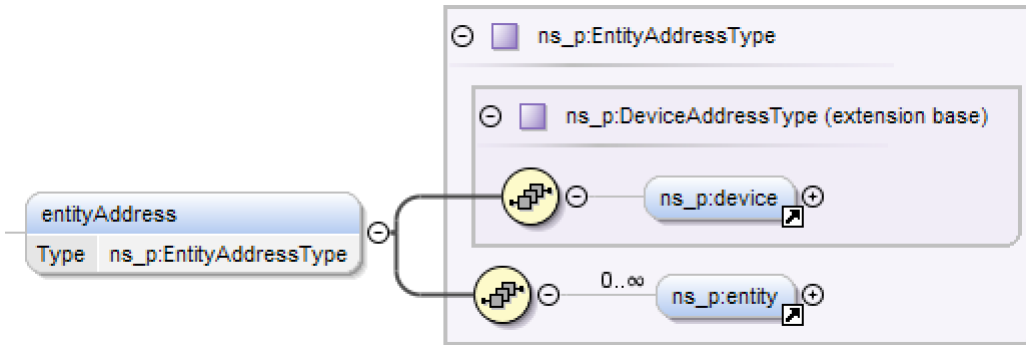


Figure 9 – Example of selectors part (extract) with entity address part

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> need not include identifiers in most cases.

<ELEMENTS> SHALL only be used in the following two cases:

- Data deletion (write/notify)
- "partial" read

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

Subsequently we iterate through all elements of a given "<ELEMENTS>" instance and call the iterated element "<element>". The following rules determine whether <element> 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:
<element> is applied.
2. If <SELECTORS> is used in the command:
<element> is applied only for list items selected via <SELECTORS>.

Among others this means that the restricted function exchange may 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

Esp. 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 section 5.5.1.1.4.5).

5.4 Communication modes

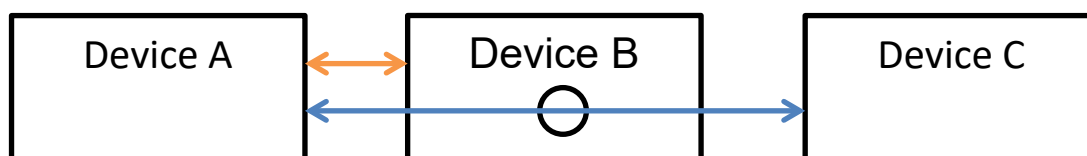


Figure 10 – Communication modes of SPINE devices A, B and C.

NOTE The circle in device B in Figure 10 symbolizes the “message forwarding” task of device B.

In Figure 10 different communication modes are briefly depicted:

1. An orange 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”.

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 need to 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 section 5.5.1.1.4.3) 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.

Please note that the communication mode may change with every message. I.e. with regard to Figure 10, device “A” may exchange messages for/from device “B” in simple communication mode, while (at

almost the same time frame) messages for/from device “C” pass device “B” as well but with enhanced communication mode.

5.4.1 Simple communication mode

In this mode, two SPINE devices “A” and “B” communicate with the following restrictions:

1. The devices “A” and “B” are considered having a direct connection, i.e. there is no third SPINE device “C” in the communication between devices “A” and “B”.
2. In datagrams exchanged between devices “A” and “B” the source and destination address elements of the datagram’s header SHALL NOT contain any other address information than those of device “A” or “B”. I.e. there is no address information of any other device.
3. The devices “A” and “B” SHALL expect that payload exchanged between these devices belongs to device “A” or “B” only. I.e. they SHALL not expect to convey information of any third device.

In simple communication mode, the device part of the addresses (source and destination) SHOULD be set in order to easily identify the communication partners.

Advice: A device SHOULD NOT be implemented in a way that it supports the simple communication mode ONLY. Instead, a device SHOULD support the enhanced communication mode as well. Simply put, a device SHOULD be implemented in a way that its own networkFeatureSet (see section 5.5.1.1.4.3) can be assigned a DIFFERENT value than “simple”.

Please consider also section 5.5.1.1.4.3.

5.4.2 Enhanced communication mode

The enhanced communication mode can only apply if both communication partners (source and destination) have their networkFeatureSet element (see section 5.5.1.1.4.3) set to a value different than “simple”. Of course, all devices that are involved in the communication as intermediate device (hop / forwarding device) shall have a networkFeatureSet value different than “simple”, too.

The enhanced communication mode provides the possibility to send messages via an intermediate SPINE device. This situation already occurs for technological gateways, i.e. SPINE-capable devices that bring devices of other communications technologies into the SPINE world by representing them with an own device address within a SPINE network. These “native” devices can only be accessed via the SPINE gateway. Additionally, this kind of “forwarding” is rather generic and can be extended to SPINE devices which are no technological gateways, but that are also not just “simple”. Among others, this applies to devices with networkFeatureSet set to “smart”.

If a SPINE device supports the simple as well as the enhanced communication mode, it SHALL set its own networkFeatureSet property to a value different than “simple” (e.g. “smart” or “gateway” or “router”). See section 5.5.1.1.4.3 for details.

If a SPINE device “X” itself can act as intermediate device, this means basically it is capable of forwarding a received message to another SPINE device (provided that it knows how to access both devices). I.e. device “X” may receive a message with a SPINE source “device” address of device “Y” and a SPINE destination “device” address of device “Z”. Such kind of message forwarding REQUIRES that device “X” supports “enhanced communication mode” AND that it has a proper “DestinationList” implemented (see section 5.5.2).

NOTE The networkFeatureSet values of the SPINE devices “X”, “Y”, and “Z” do not need to be identical.

The device part of all addresses SHALL be set when using the enhanced communication mode to identify where the message originates from and where it shall be routed to.

5.5 Functional commissioning

Functional commissioning comprises mechanisms like binding and subscription (but not any commissioning mechanisms of underlying communications technologies or layers). These mechanisms require the definition and assignment of roles with regard to a dedicated functionality (feature). Altogether this leads to clear responsibilities between devices and helps to implement rather automatic exchange of information between devices.

In order to support binding and subscription a discovery mechanism is defined that **SHOULD** be executed in advance.

The focus on binding and subscription is primarily on the definition of “ownership” of data. The owners **SHALL** push information updates to dedicated recipients (further operations are permitted but skipped right now for convenience).

Besides this approach to push information it is still possible to pull information (i.e. read specific information from the owner of the data) without a previously configured binding or subscription. However, the latter might be manufacturer specific and is not considered further in this subclause unless for cases explicitly described.

The subsequent sections describe messages (more precise: message parts) and rules on NodeManagement instances.

The message parts of the subsequent sections only show the SPINE function specific payload part of the element “datagram. Payload”.

5.5.1 Detailed discovery

Every device has some functionality which it can announce to other devices via the discovery mechanisms of SPINE, as shown in Figure 11.

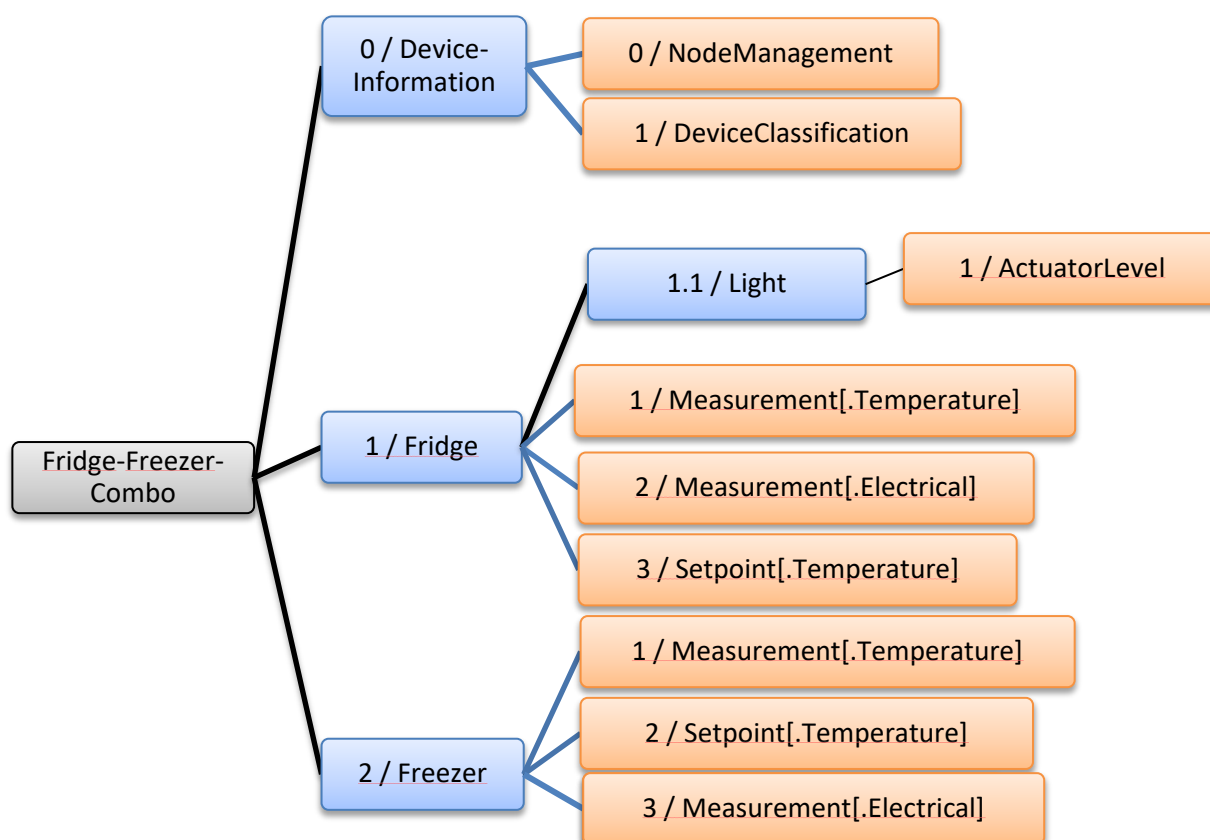


Figure 11 – Discovery example

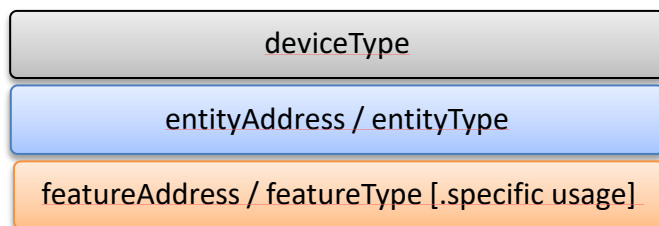


Figure 12 – Hierarchy types

NOTE In Figure 12 entities can contain child-entities; "entityAddress" contains all "entity" parts starting from the respective root entity.

The general architecture was already explained and will now be used in the context of the discovery process. In this document, a physical device is represented by a SPINE device with a *deviceType*, the SPINE device is subdivided into entities with an *entityType* and features with a *featureType*, as described in Figure 12.

1. A SPINE device holds a collection of one or more entities. The **deviceType** describes a generic context (e.g. "Washer"), but may also describe a certain minimal set of entities that can be expected on this device.
2. An entity holds a collection of one or more (child) entities and features. The **entityType** describes a generic context (e.g. "Fridge"), but may also describe a certain minimal set of (sub-)entities and features that can be expected on this entity. A special entityType "DeviceInformation" gives information on the device itself.
3. A feature with the role "server" or "special" holds a collection of one or more functions. The **featureType** describes a generic context (e.g. "ActuatorLevel"), but may also describe a certain minimal set of functions that can be expected on this feature. The role expresses that the feature is "owner" of the functions (except for "...Call" functions that are only used with the "call" classifier).
4. A feature with the role "client" may as well keep a list of functions and express its featureType. However, features of this role do not "own" the functions (i.e. they cannot be read on such a feature, among others). They can just "use" features of the role "server" or special. In the subsequent discussions the role "client" is not considered in detail.
5. A function holds a collection of one or more elements. The elements might have simple or complex data types. All readable or writeable functions can be discovered over a function list in the feature description. The information on the device, entities and features are provided by the primaryNodeManagement instance of each SPINE device and can be discovered during a detailed discovery process (see below).
6. A read command on the function provides a copy of the function with its elements (the copy might be restricted dependent on the kind of the read command).

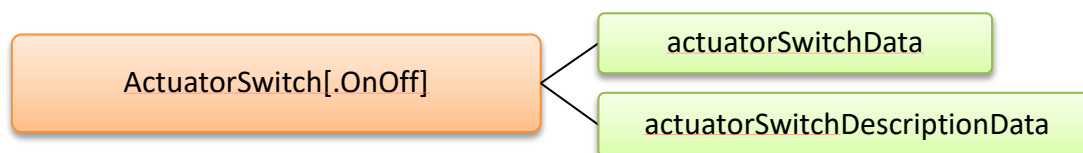


Figure 13 – Function Discovery Example over Feature Description

5.5.1.1 Basic definitions and rules

Detailed discovery requires the presence of own NodeManagement instances (presentation of own information) and the access to remote NodeManagement instances (gather information of another device). Detailed discovery SHALL be supported by all SPINE devices on their entity 0 (entity with entity address = 0) and feature 0 (feature with feature address = 0).

1566 A SPINE node discovers the remote SPINE node's device information, entity information and feature
 1567 information using the mechanisms specified in this subclause. Detailed discovery between trusted
 1568 SPINE nodes may be triggered at any time. Section 5.5.1.3 discusses the situation if a primary
 1569 NodeManagement instance alters its device information, the set of entities or features during runtime
 1570 and how this SHALL be handled.

1571 In general, a SPINE node performs a detailed discovery of another SPINE node because it is
 1572 searching for features (or more specifically for featureTypes) to connect to. As an example, a node
 1573 which has an "ActuatorSwitch" client searches for an "ActuatorSwitch" server to connect to. If a client
 1574 finds a fitting feature it is recommended to use this feature "in time" (e.g. send a "read" request), as
 1575 shown in Figure 13.

1576 NOTE A client does not need to implement a specific client feature like "ActuatorSwitch" (from the example
 1577 above) to connect to an "ActuatorSwitch" server feature. Instead, a client may handle all its functionality with one
 1578 feature, although that may cover different featureTypes on the server side.

1579 A SPINE feature type may be specialized by a specific usage. As an example, a "Measurement"
 1580 server could be specialized to say "Temperature", which means it measures temperatures.

1581 For mandatory rules about creating and deleting bindings and subscriptions, consider sections 5.5.3.

1582 5.5.1.1.1 Rules for devices

1583 The description of a (physical) device mainly consists of some general information and the list of
 1584 provided entities. Manufacturers are free to choose an entity design for their SPINE node, consisting
 1585 of standardized entity types and proprietary entity types.

1586

1587 Device type:

1588 Devices are usually specified by a device type (like "Washer", e.g.). In either case the device type is
 1589 modelled in the deviceType element of "deviceInformation.description". The specification provides
 1590 some standardized values for the device type

1591 The device type SHOULD NOT be used for any kind of automatism.

1592

1593 Device address:

1594 The address of a device is modelled as a string. Its length is restricted to a maximum of 256
 1595 characters. Up to this revision of SPINE, only the following pattern is permitted for the device address
 1596 string (notated as XML schema regular expression):

1597 `d:_(i:[1-9][0-9]*|n:[a-zA-Z0-9-]+)_[^p{Cc}\p{Cf}\p{Z}]+`

1598 The pattern requires the address to first state "d:", then the pattern of the vendor specific extensions
 1599 follows to give the address the needed uniqueness. After that, the vendor states a string containing a
 1600 vendor-wide unique address that MAY consist of the following characters: Any Unicode character
 1601 EXCEPT for such Unicode characters belonging to the so-called "general category" definitions
 1602 "Control", "Format", and "Separator". Among others, this means white space characters are not
 1603 permitted.

1604 Examples for possible device strings:

1605 - `d:_i:46925_ABCabc-123`

1606 - `d:_i:46925_0123456789`

1607 The IANA PEN SHOULD be used! The device address part after the second underline SHALL be
 1608 unique! Each vendor is responsible for the uniqueness of its device addresses.

5.5.1.1.2 Rules for entities

A standardized entity type has a set of standardized features and child-entities, which are either optional or mandatory.

In general, each entity (standardized or non-standardized) can hold a mixture of:

1. Child-entities (see section 5.2.3).
2. Features which are included in the standardized entity type.

A SPINE node SHALL NOT expect other SPINE nodes to have any other features in a standardized entity type but the ones included in the standardized entity type. This means the other node may have additional features on the entity but it is not valid to insist on the presence of these additional features just from the standardized entity type. I.e. other included features of the remote node have to be discovered manually.

Please note the architecture described in this document requires a specific mandatory SPINE entity at least: The entity containing the so-called "primary NodeManagement instance". This entity has the entityType "DeviceInformation".

Entity types used in a SPINE node, which do not follow any standardized entity descriptions, SHALL be marked as vendor specific.

A standardized entity is a SPINE node that implements a standardized entity type and has or has not been extended with non-standardized content.

5.5.1.1.3 Rules for features

A feature that uses a standardized featureType in a standardized way SHALL NOT be marked as vendor specific. A feature which uses a proprietary class SHALL NOT use a standardized feature type and SHALL be marked as vendor specific. A feature which uses a standardized class but uses a proprietary feature type SHALL be marked as vendor specific.

5.5.1.1.4 Rules for specific element usage

The subsequent sections discuss some elements that are used in the "detailed discovery" messages (see section 5.5.1.2).

5.5.1.1.4.1 Usage of element "deviceAddress. device"

According to section 5.3.2.2 the "device" information elements of two devices SHALL be different. Section 5.5.1 defines different messages for "discovery" processes between NodeManagement instances of two devices "A" and "B". Some of these messages contain the element "deviceInformation. description. deviceAddress. device". This element conveys the originator's "device" address part.

The requirement on the uniqueness of the "device" part of the address REQUIRES a device "A" to terminate a connection with a communication partner (device "B") immediately if device "A" recognizes that its own "device" value is identical to the one of device "B". Similarly, device "B" SHALL terminate a connection with device "A" if it detects the non-uniqueness of "device".

5.5.1.1.4.2 Usage of element networkManagementResponsibleAddress

The element "networkManagementResponsibleAddress" is reserved for specific network management purposes that will be defined in a later version of this document.