EEBus Initiative e.V.
Butzweilerhof Allee 4
50829 Cologne
GERMANY

Rue d'Arlon 25
1050 Brussels
BELGIUM

Phone: +49 221 / 47 44 12 - 20
Fax: +49 221 / 47 44 12 - 1822

info@eebus.org
www.eebus.org

District court: Cologne
VR 17275

EEBus SPINE Technical Specification

Resource Specification

Version 1.1.1

Cologne, 2018-12-21

The EEBus concept was developed as part of E-Energy.

Supported by: Federal Ministry of Economics and Technology on the basis of a decision by the German Bundestag

Table of contents

List of figures

List of tables

829

830

1 Introduction

The SPINE data model is described within this technical specification (please keep in mind that some information may only be specified in the SPINE XSDs, see below). It contains the SPINE resource model, consisting of the device types, entity types and feature types as well as descriptions of the (standard and complex) classes as they are defined through the SPINE XSDs that are part of each release of the SPINE specifications. Additionally, some general rules, relations and data type descriptions are included.

The underlying SPINE data model of this document is identified by the XML namespace "http://docs.eebus.org/spine/xsd/v1".

1.1 References

1.1.1 EEBUS SPINE documents

[Introduction]	EEBus_SPINE_TR_Introduction.pdf
[ProtocolSpecification]	EEBus_SPINE_TS_ProtocolSpecification.pdf
[DataModelXSDs]	EEBus_SPINE_TS_ActuatorLevel.xsd, ..., EEBus_SPINE_TS_Version.xsd

1.1.2 Other EEBUS documents

[SHIPSpecification]	SHIP_Specification_V1.0.0.pdf
[SPINEUseCaseXSDs]	XSDs delivered within the EEBUS Use Case Package

1.1.3 Other documents

[SGReady]	2012-11-01_SG_Ready_Regularien_Version1.1.pdf

1.1.4 Websites

[EEBus]	http://www.eebus.org	*Official EEBus Initiative e.V. website.*
[W3C]	http://www.w3.org/	*Official World Wide Web Consortium website.*
[W3Schools]	http://www.w3schools.com/	*Tutorials and examples for XML and others.*

1.1.5 Normative References

[RFC2119]	IETF RFC 2119: 1997, Key words for use in RFCs to indicate requirement levels Please see section 1.4.5 for details.

1.2 Terms and definitions

Binding
Concept for connecting functionally matching features.

(Standard or Complex) **Class**
Set of SPINE functions used to model data for a specific functionality. A class can be considered as a topic where functions are defined for. For example the SPINE class “Measurement” is a collection of SPINE functions that are used to describe measurement values.

Classifier
Specifies whether a message serves to read, reply, write, etc.

Client
Role that specifies that a node uses data from a "server" or can change it.

Command
The functional part of a Message.

Complex Class
SPINE class that is build up by parts of SPINE standard classes and combines them in a new, ordered way.

(SPINE) **Data model**
Definition of the possible data that can be used for SPINE communications. Defined as XSD.

Device (specific node)
SPINE node that can include a set of entities. It has a "Device Type". With regards to the hierarchy of SPINE nodes a device is a root node for all functionalities offered by a device.

“device” (address information)
SPINE address part for the (physical) device.

Device Type
Specific type of physical device (e.g. "WashingMachine", "HeatPump", "FridgeFreezer", etc.).

Discovery
Process of finding appropriate partners for communication. Dependent on the context this can be either finding other devices or examination of a device’s potential functionalities.

DSO
Distribution System Operator of an electricity grid.

Element
Item (or “attribute”) of a SPINE function. Holds one information (e.g. "timestamp", "value", etc.) or contains further sub-elements.

Entity (specific node)
SPINE node that can include a set of (child) entities or features. It has an "Entity Type". With regards to the hierarchy of SPINE nodes an entity is a child element of a device or another entity.

“entity” (address information)
SPINE address part for the (logical) entity.

Entity Type
Specific type of logical device (e.g. "Freezer" is one logical part of a physical device "FridgeFreezer").

EV
Electric Vehicle

EVSE
Electric Vehicle Supply Equipment

Feature (specific node)
SPINE node that can include a set of functions (of a class). It has a "Feature Type". With regards to the hierarchy of SPINE nodes a feature is a child element of an entity.

“feature” (address information)
SPINE address part for one feature.

Feature Type
Defines optional or mandatory rules and a general behaviour of the underlying Class (standard or complex).

(SPINE) Function
A (SPINE) function is the smallest structure to model “actual data” (“functional data”). I.e. functions usually consist of child elements that each hold an information (e.g. "timestamp", "value", etc.). Information between communication partners is exchanged via the exchange of a function (as part of a so-called “payload”).

Header
SPINE Header, including elements for addressing, unique identification of messages, timestamp, etc.

Message
One SPINE transfer from a sender to a receiver.

(XML) **Namespace**
XML namespaces provide a simple method for qualifying element and attribute names used in XML documents by associating them with namespaces identified by URI references (source: www.w3.org).

Node
Common term for a SPINE instance that has a SPINE address. Dependent on the situation a node can be either a device or an entity (of a specific device) or a feature (of a specific device-entity).

Payload
SPINE Payload, containing the functional SPINE data.

Role
Each Feature has a functional role, usually either “server” (data owner) or “client”. For some special features (NodeManagement, e.g.) the role “special” is defined.

Scope (Type)
Some feature types define scope types for identifying specific functionalities unambiguously (e.g. *outsideAirTemperature*).

Server
Role that specifies that a node offers own data to be read or written by a node with role client. A server can notify its data to other nodes (with role client).

SPINE
Smart **P**remises **I**nteroperable **N**eutral-message **E**xchange

Standard Class
All basic/standard functions are defined in standard classes. Functions of standard classes follow very simple patterns and do not have deeply nested data structures.

Subscription
Enables the receiving of messages of interest from another device without polling it.

Use Case
Textual description of a re-usable functionality consisting of one or more messages of one or more participating actors. May be visualized with a sequence diagram. E.g. "A CEM shifts the energy usage of a washing machine."

User Story
Complete (but specific) business case described from the perspective of a user. Can be separated into several use cases. E.g. "The user wants to get the laundry done by 8:00pm."

XML (Extensible Markup Language)
Human- and machine-readable markup language containing data. Used to model SPINE messages.

XSD (XML Schema Definition)
Definition format for XMLs, written in XML. Specifies how a well-formed XML (in regards to this XSD) can be built. The SPINE data model is defined in XSD and supplementary documents (as not every rule can be specified with XSD only). Other formats than XML can be derived from an XSD, too (e.g. JSON).

1.3 Document structure

The main focus of this document is the description of the SPINE resources and the data model. Before going into technical details, chapter 2 presents a coarse overview of the overall concept. Chapter 3 collects a number of technical low-level details and can be left out at first reading. Chapter 4 defines types of differenct facets of a device (device types, entity types, feature types). These types contain also rules which are relevant to establish an interoperable use of exchanged data. Finally, chapter 5 describes SPINE classes with so-called “functions”. These “functions” are the basic data structures that are used within “messages” to represent or modify the content of a functionality or data point. This chapter describes the potential purpose and use of functions, whereas chapter 4 (and esp. the feature types defined in section 4.3) impose rules on their use.

1.4 How to read this document

1.4.1 Graphical representation

1.4.1.1 Introduction

Documentation provided by the EEBus Initiative e.V. sometimes makes use of graphical representations of parts of the SPINE data models in order to ease the readability. All graphical representations are created using a specific commercial tool (oXygen). Please note that these graphical representations are not standardized. Furthermore, the explanations given below are tailored to the needs of the SPINE data models. In general, the explanations are not exhaustive.

1.4.1.2 Explanation of graphical XSD representation

The graphical XSD explanation uses a set of images, which are described to accurately understand the diagrams. This section gives an overview of these images.

1.4.1.2.1 XSD "simple type"

Figure 1: XSD "simple type"

This image represents an XSD "simple type" (in this case a string). Simple types are signed with a left upper corner triangle. A simple type can either be given by the W3C (starting with "xs:" in this document; see also 1.4.2) or can be self-defined. Self-defined simple types can only be restrictions of given W3C simple types, such as making a string an enumeration (only allowing certain words/literals) or restrict a number to a certain range. The kind of restriction is not graphically represented, but can only be read in the XSD file itself.

1.4.1.2.2 XSD "complex type" with no content

Figure 2: XSD "complex type" with no content

This image represents an XSD "complex type". Complex types are signed with a square. A complex type can consist of different other complex and/or simple types, which are embedded. A complex type might, however, also be completely empty, which is the case shown in this image. An empty "complex type" has no effect if used, other than to be a placeholder for easier extensibility.

1.4.1.2.3 XSD "complex type" with content

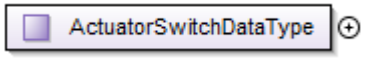


Figure 3: XSD "complex type" with content

This image shows an XSD "complex type" with a content (shown by the "+").

1.4.1.2.4 XSD element with type

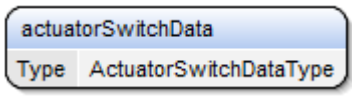


Figure 4: XSD element with type

This element represents an XML element “actuatorSwitchData”. The contents of this element are defined by the type “ActuatorSwitchDataType”. An element can be defined using a type, or can be directly defined. Using a type has the advantage of being easily extensible.

Please note that figure above may vary. This depends whether the type is empty and whether it is referenced from a separate XSD.

1.4.1.2.5 XSD sequence

Figure 5: XSD sequence

This image shows an XSD “sequence”. The element connected on the left (not shown in the image) is the one, which contains this sequence (the “parent”). The elements connected on the right are contained in that sequence (the “children”). A sequence is an ordered list of elements on the right, contained in an element connected on the left. (An unordered list is an “all”).

1.4.1.2.6 XSD choice

Figure 6: XSD choice

This image shows an XSD “choice”. The element connected on the left shall contain exactly one of the elements connected on the right (none of these elements are shown here).

1.4.1.2.7 XSD all

Figure 7: XSD all

This image shows an XSD “all”. The element connected on the left can contain any of the elements connected on the right (“children”; none of these elements are shown here). This is similar to an XSD sequence. The difference is that each “child” is implicitly optional (i.e. can be absent in an XML) and that the order of the children in an XML is arbitrary.

1.4.1.2.8 Optional vs. mandatory

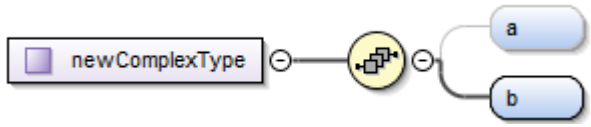


Figure 8: Optional vs. mandatory

There are different styles for the display of connections between building blocks. These styles are used to display whether a building block is optional or mandatory, among others. The image shows a complex type “newComplexType”, consisting of a sequence of an optional element “a” (light grey line) and a mandatory element “b” (dark line).

1.4.1.2.9 Lists / multiplicity

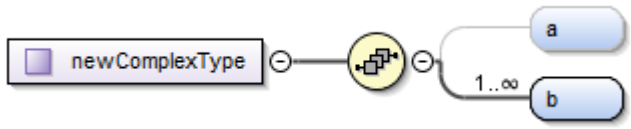


Figure 9: Lists / multiplicity

Elements might also be present multiple times. The shown image shows a complex type “newComplexType”, containing a sequence of an optional element “a”, as well as 1 up to infinite elements “b”.

1.4.1.2.10 Example: “Expanded view”

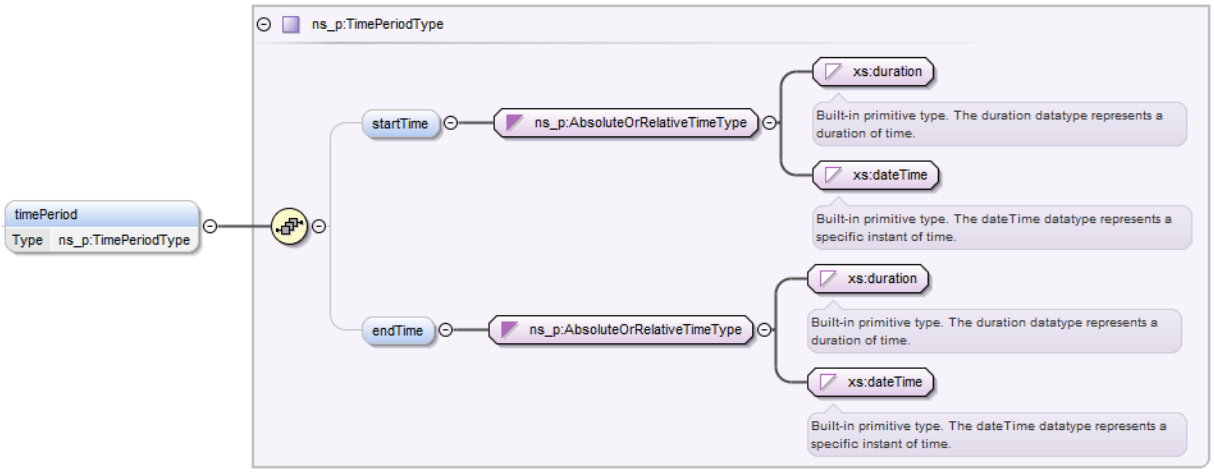


Figure 10: XSD expanded view

This image shows an element called “timePeriod”, which is based upon the Type “TimePeriodType”. Attached to it (on the right, embedded in a purple box) is the definition of the type “TimePeriodType”. The type is a sequence of two optional elements called “startTime” and “endTime”, which are both using the “AbsoluteOrRelativeTimeType” format. There are also informative comments shown in this image, which are embedded in little speech bubbles / balloons.

An XML part conforming to this XSD at least follows the structure of:


```
    <timePeriod>
        <startTime>2006-05-04T18:13:51.0Z</startTime>
        <endTime>2006-05-04T18:13:51.0Z</endTime>
    </timePeriod>
```

1.4.2 Namespaces

The SPINE data models make use of many types defined by W3C. These types are referenced within the SPINE data models under a so-called namespace of the W3C XML Schema. The SPINE data models currently denote this namespace using the prefix “xs:”. Thus, a W3C XML Schema type like “dateTime” finally leads to the reference “xs:dateTime”.

It shall just be noted that the SPINE data models put the definition of their own types and elements under namespaces of the EEBus Initiative e.V.

1.4.3 "datagram" vs. specific function – XMLs with different roots and scopes

The SPINE data models provide definitions for different levels/layers. The subsequent figure shows a (partly collapsed) graphical representation of the SPINE XML schema definition of "datagram", with a specific branch highlighted in red for a subsequent example.

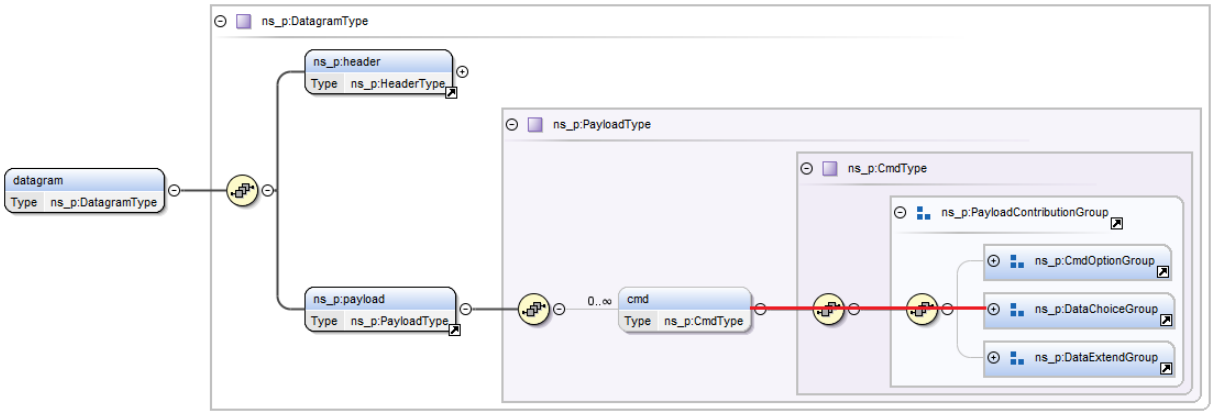


Figure 11: Datagram structure (red line: "way" to the functions)

The next figure shows a graphical excerpt of possible children of the highlighted branch:

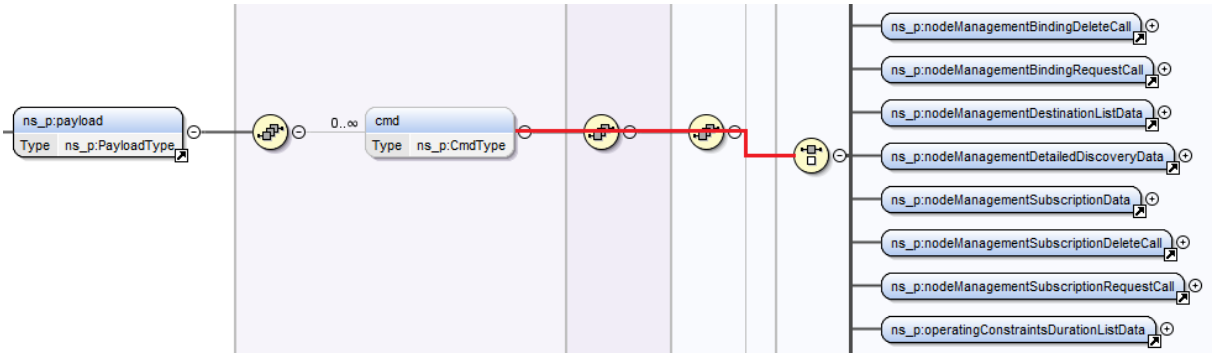


Figure 12: Payload structure (red line: "way" to the functions)

These children are also called "functions" within the SPINE specification.

It is possible to create valid XMLs with either "datagram" or any of payload's child elements (like a concrete function as from the second picture) as root element. It depends on scope and "layer" (in terms of the ISO-OSI layer model, e.g.) which kind of XML is required.

Throughout this document most explanations of classes will make use of XMLs with specific functions as root tag. However, it shall be noted that also XMLs with "datagram" as root tag will be discussed and that different kinds of "payload" contents will be explained.

1.4.4 Simplification of XML examples

Supposed the SPINE data model of "datagram" is available for download from "http://docs.eebus.org/spine/xsd/v1/EEBus_SPINE_TS_Datagram.xsd". The beginning of a valid SPINE XML with a root tag ("datagram" in the subsequent example) could look like this:

```
<?xml version="1.0" encoding="UTF-8"?>
<datagram xmlns="http://docs.eebus.org/spine/xsd/v1"
 xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
 xsi:schemaLocation="http://docs.eebus.org/spine/xsd/v1
EEBus_SPINE_TS_Datagram.xsd">
...
```

The schemaLocation information of the SPINE data models is not finally decided yet, is not always mandatory within an XML (see W3C definitions for details) and is not required for the explanation of the SPINE data models. Hence, it is usually left out throughout this document:

```
<?xml version="1.0" encoding="UTF-8"?>
<datagram xmlns="http://docs.eebus.org/spine/xsd/v1">
...
```

The W3C recommendation does not oblige the XML declaration which contains the XML version information, but it clearly recommends its use. However, this part is not of interest for the presentation of examples.

The XML namespace is omitted within this document, too, as there is no need to mention it in all examples.

Thus, the example is further simplified:

```
<datagram>
...
```

As most examples in this document are specific for a single function, the information of the header is not of interest and therefore not included in the examples. The <datagram> declaration as well as the subjacent element <payload> is not needed either. Most examples begin with the function itself, or with the <function> element, if “restricted function exchange” (with “<filter>”) is used before the actual function.

Please see [ProtocolSpecification] for more information about the structure of SPINE XMLs.

1.4.5 Used requirement keywords

The following keywords are used:

- SHALL
- SHALL NOT
- SHOULD
- SHOULD NOT
- MAY

They apply only if written in capital letters!

For the meaning of the keywords, please refer to [RFC2119].

1.4.6 Presence indications in Feature Types

If no presence indication keyword is stated for an element, the keyword "MAY" SHALL be applied (in other words: if not stated otherwise, an element is optional).

1.4.7 Presence indications in SmartEnergyManagementPs

The following abbreviations on the presence of elements or the support of entities or features are used:

- M = mandatory
- R = recommended
- O = optional
- C = “choice”, i.e. a presence depends also on the selection from multiple possibilities

meaning that a functionality SHALL be supported (mandatory), SHOULD be supported (recommended) or MAY be supported (optional). If "choice" is used, the element description needs to be considered to find out which alternative applies.

These abbreviations are used exactly the same way as stated in the section “Element presence indications” of the document [ProtocolSpecification]. The following information is a brief excerpt from this document:

In case of elements (of data model definitions), the presence indications “M” and “C” are always meant relative to the respective parent element. I.e. if a parent element is optional (“O”) and a child is mandatory (“M”) the child element can only be present if the parent element is present as well. The presence indications from the data model definitions describe general requirements on the presence of elements. These requirements can be strengthened (but not weakened) by process rules.

For more information and examples please consider the above mentioned document. In general, the design concept of section "Data model specialization: ..." of the document [ProtocolSpecification] applies here as well.

2 Overall model hierarchy concept overview

The following figure gives an overview on the miscellaneous levels that contribute to the definition and use of interoperable functionality.

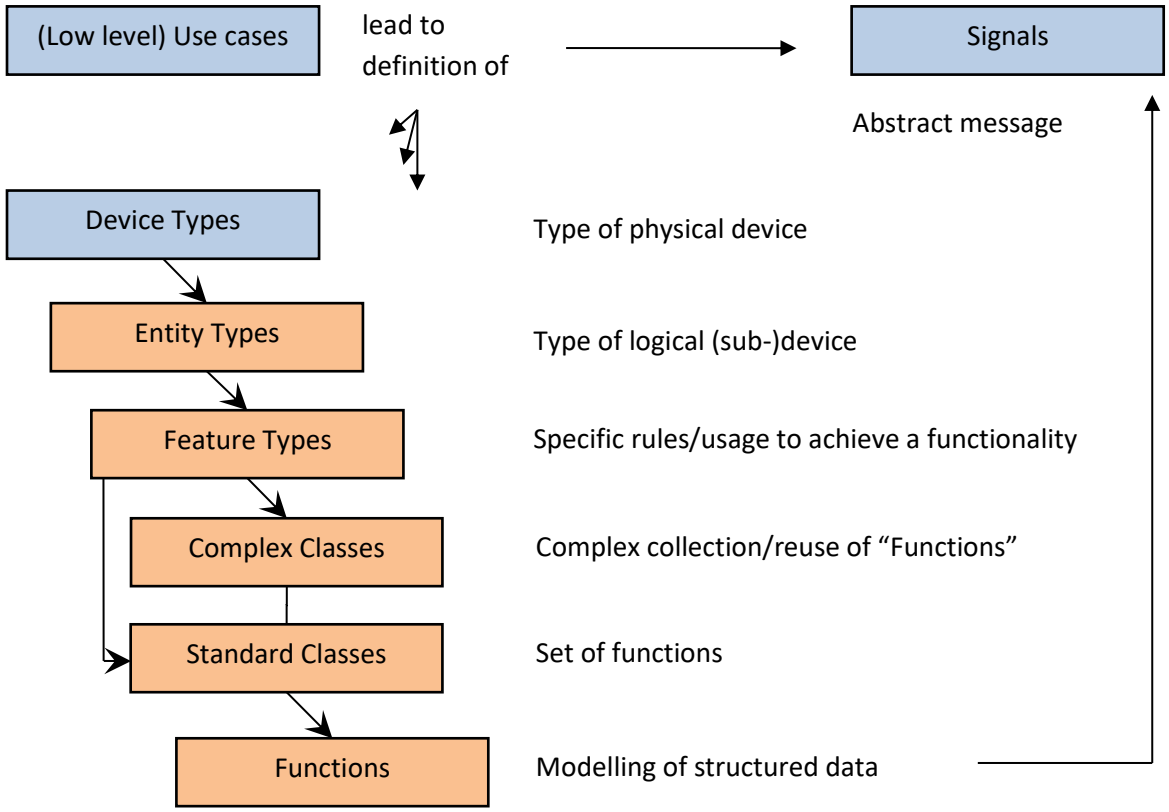


Figure 13: Dependencies of the standardised types and identifiers

Apart from user stories, the most abstract level deals with use cases and abstract signals exchanged between actors. Starting from the top of Figure 13, each level defines its own set of definitions for its own hierarchy of the problem. The device facets “device type”, “entity type”, and “feature type” define logical structures of a device and rules to achieve a required functionality. These facets are introduced in more detail in section 2.1.2. The required functionality makes use of well-defined data structures (so-called “functions”) which are organised in so-called classes. The class concept is explained in section 2.2. The data within the functions is expressed with so-called "elements". The rules and the functions together permit the modelling of signals. This is used in document [ProtocolSpecification] to model a neutral message framework.

2.1 Resource Type Definition Concept

2.1.1 Introduction

The SPINE device model consists of three layers

- Device
- Entity (with sub-Entities)
- Feature

where the following applies:

- 1 Device has 1..K Entities
- 1 Entity has 0..L sub-Entities
- 1 Entity has 1 parent (Device or Entity)
- 1 Entity has 0..M Features
- 1 Feature has 1 parent (Entity).

These relations can be combined. The following figure gives an example for this:

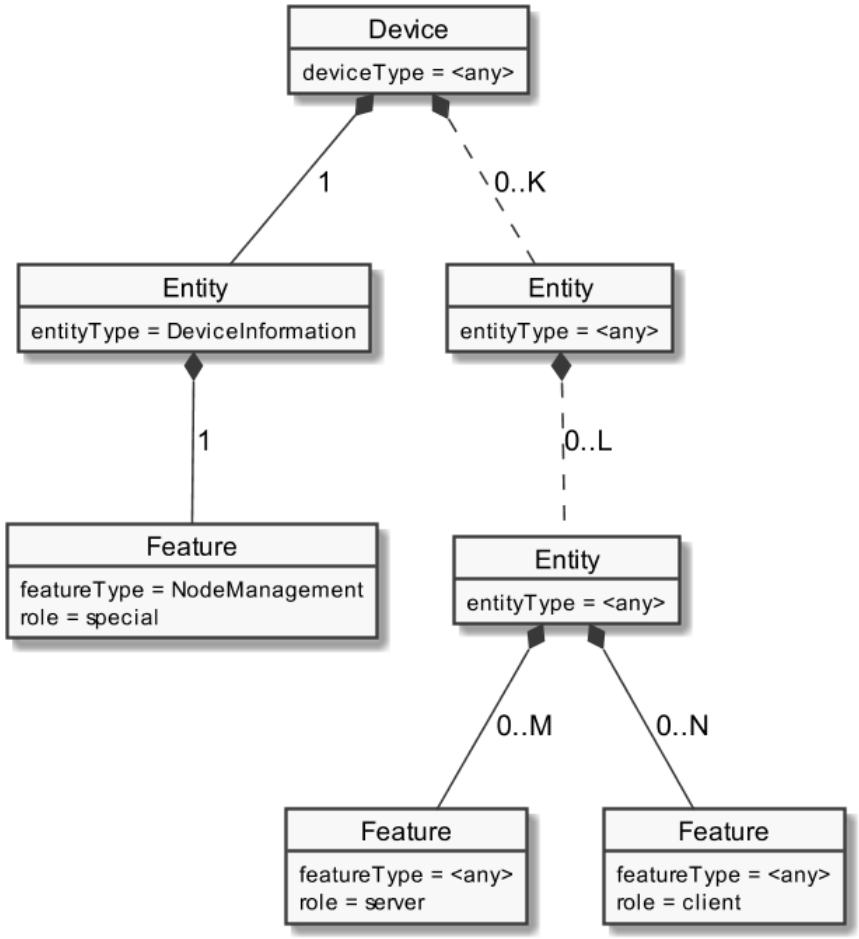


Figure 14: SPINE device model example

The dotted line in the figure indicates that further Entities MAY be present between the depicted instances.

In the following sub-sections these components are described more in detail.

2.1.2 Device Model Facets

2.1.2.1 Device Type

The description of the (physical) device (that can be seen as kind of a root instance within the SPINE device model) includes a short description of the device. As the Device Type is only of informative character, no further information is needed.

Vendors may use official Device Types as defined in section 4.1 or create own Device Types that then have to be marked as vendor specific ones (see section 2.1.3). Any kind of Device Type may contain any combination of Entity Types (official or vendor specific ones).

2.1.2.2 Entity Type

In contrast to the only informative Device Type, the Entity Type is more important for the interoperable working of SPINE. Within the Use Case definitions, Entity Types are mapped to the Use Case Actor's functionality. Hence, a discovery of the correct Entity Type (and its path, see below) may be of relevance.

As already denoted in sub-section 2.1.1, Entities may be nested, meaning an Entity may have other Entities as child. This can lead to a tree structure of Entities. This hierarchy can be retrieved from the detailed discovery (see [ProtocolSpecification] for details).

2.1.2.3 Feature Type

Feature Types define optional or mandatory rules for the underlying Class (mostly with the same name, e.g. the Feature Type Measurement is based on the underlying Class Measurement), their functions and the elements and values within the functions. As most rules must be deriveable from the elements and values presented on the SPINE Features, the corresponding rules are provided for each element and value within the tables of the “Feature Types” section.

2.1.3 Vendor specific extensions

The Device-/Entity-/Feature-Type structure mentioned in the sections above relies on definitions provided by the EEBus Initiative e.V. for ensuring interoperability. If a vendor needs type definitions that are not already provided, a vendor specific extension allows the definition of new types that add further functionality. These extensions are completely vendor specific and will not be suppported by the EEBus Initiative e.V. at all. A vendor may bring these definitions into the EEBUS working-groups and, if of interest, they potentially could become part of a future SPINE release.

These types defined by the vendors themselves SHALL be marked as vendor specific as long as they are not part of an official SPINE release. For details please see [ProtocolSpecification], section "Rules for vendor specific extensions".

2.1.4 Feature Group

Some features may need another feature to fulfil a functionality in a senseful way. E.g. the Setpoint Class should be linked to some Measurement instance, so the desired value (from Setpoint) can be compared to the actual one (Measurement). Therefore, feature groups can be used. They are always entity-wide defined. Two entities may define the same feature group for a different meaning. If two or more features shall be linked, the same feature group is assigned to them (e.g. "#1"). More than one feature group may be defined for one feature (e.g. if a Measurement instance is linked to a Setpoint instance as well as to a SmartEnergyManagementPs feature). If the features that reference

each other are only defined once on an entity, no feature group is needed (picking up the previous example: if there is only one Measurement and one Setpoint instance on an entity, no ambiguity is given). If, without the feature groups, there would be an ambiguity, feature groups SHALL be used to prevent the ambiguity and identify the associated features.

Examples:

- #1
- #1#2
- #2#5

2.1.5 Specific usage

This functionality is deprecated.

2.2 Class Concept

2.2.1 Standard Classes

The SPINE data model relies on the standard classes (like ActuatorSwitch, Measurement, etc.). They define the functions with all their elements, some examples and a general description what can be communicated with this functionality. No (or just very few) rules are given in the class sections as the definition of rules is the scope of the device model (see section 2.1.2).

Please note that some classes (or at least function type definitions) are also used in so-called complex classes (see section 2.2.2). In fact, for some classes there may even be no standard Feature Type defined if they are only used within a complex class. In this version of the specification this is the case for the classes

- BindingManagement
- SubscriptionManagement
- Version

as they are primarily re-used within the complex class NodeManagement.

2.2.2 Complex Classes

Complex classes combine one or more functionalities from standard classes in new functions that are treated the same way as functions from standard classes. They do not define new elements as leaves (in a tree structure), but only order the standard class functions in another way. Lists may be added and (leave) elements may be deleted.

2.2.3 SPINE class hierarchy (Class / Sub-Class / Function-Group / Function / Element)

To identify different levels of SPINE data-structures, the following naming conventions are defined:

Class: A SPINE class is the highest level of data-grouping. E.g. all functions relating to *PowerSequences* are described within the *PowerSequences* class. The first letter of the SPINE class name is always capitalized.

Sub-Class: Classes may have sub-classes. They are introduced with a short sub-section in the corresponding class description. E.g. the *PowerSequences* class contains the sub-classes *PowerTimeSlot*, *PowerSequence* and *PowerSequenceNode*. These are not marked as own groups in the XSD. The sub-classes are not used in any identifier and are not relevant for interoperability considerations. The first letter of the SPINE sub-class name is always capitalized.

Function-Group: A function-group consists of the function itself (xListData or xData) and (in case of a ListData) its corresponding xListDataSelectors. The first letter of the SPINE functions-group name is always lowercase.

Function: The ressources that are used for communication are called functions (e.g. *powerTimeSlotValueListData*). The first letter of the SPINE function name is always lowercase.

Element: Elements contain the actual data information (e.g. value, timestamp, etc.). The first letter of the element name is always lowercase.

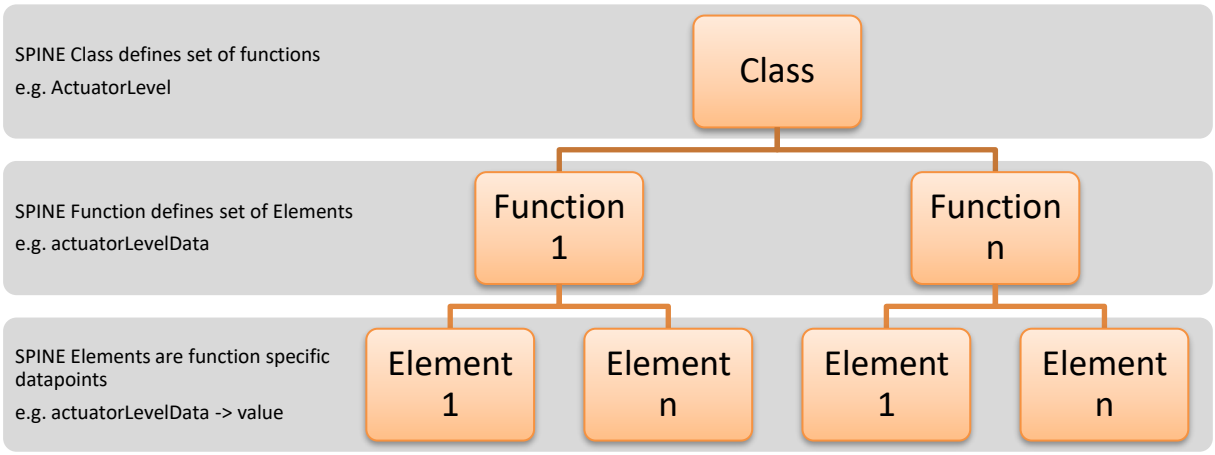


Figure 15: SPINE class hierarchy

3 Common technical details

3.1 Introduction

The subsequent sections define and discuss miscellaneous technical details that are required by classes.

3.2 Time information (absolute / relative / recurring)

Nearly all time information is of type: *AbsoluteOrRelativeTimeType* (see below).
Exceptions:
- timeInformationData -> utc
- timePrecisionData -> lastSyncAt

The simple type *AbsoluteOrRelativeTimeType* is a union of xs:dateTime and xs:duration. So it is possible to either model absolute times (xs:dateTime format, e.g. 2014-04-28T17:43:14.0Z) or relative time information (xs:duration format, e.g. P2Y8M15DT7H32M17S).

Unless specified otherwise, an xs:duration based time of *AbsoluteOrRelativeTimeType* always relates to "now".

Absolute times SHALL always be stated as UTC-Time ("Z" at the end of the time information).

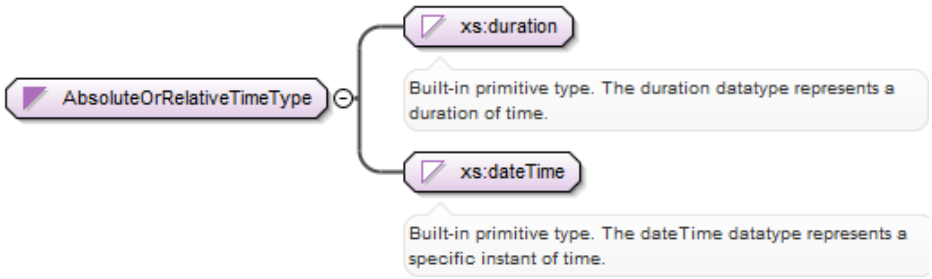


Figure 16: Absolute or relative time type

For recurring times, another type (*AbsoluteOrRecurringTimeType*, see section 3.10.2.13) is defined.

3.3 List Data concept

Some SPINE classes provide lists of data instances. These lists always consist of 0 (hence, may be empty) to unlimited count of single data instances (and nothing else). A function can either contain a (non-list) data as ressource or the list data. If a list data is defined in a SPINE class, only this list data may be used as ressource, not its non-list data (which is only used as reference in the list data).

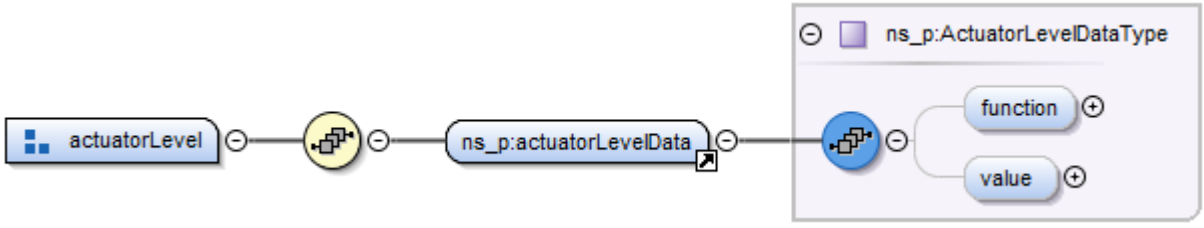


Figure 17: Non-list data function example

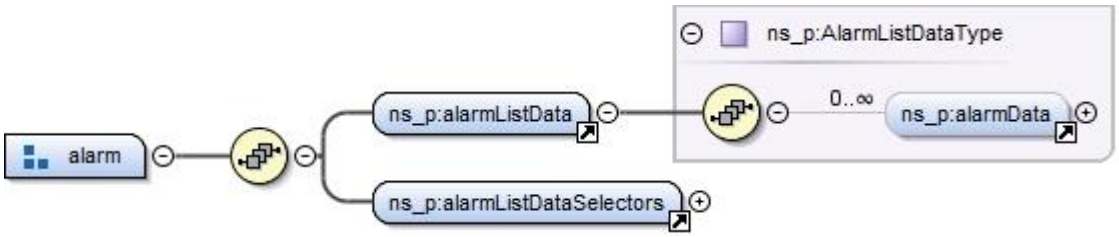


Figure 18: List data function example

3.4 Identifier concept

3.4.1 Introduction

Identifiers are used esp. in ListData-functions to identify single list entries unambiguously. They are also used for references within the class itself (to relate the static and non-static information) as well as for references from other classes (e.g. the Setpoint class (see section 5.3.21) references the Measurement class (see section 5.3.15), to announce for which actual measurand a specific setpoint is used for). References are only possible within the same entity, not between different entities or devices.

A special kind of identifiers are the address constituents “device”, “entity”, and “feature” (the types are defined in section 3.10.3). Among others, the element “entity” can occur multiple time (i.e. as list) within some address definitions. More details are explained in section 5.3.17.1.1.

Note: For some functions or use cases a specific data instance is only uniquely identified with the use of multiple identifiers (e.g. within the Power Sequences class a “slot” with number 2 of “sequence” 5 is different to “slot” 2 of “sequence” 7; this typically requires the use of sequenceId AND slotNumber in order to identify a specific slot). In some cases a data instance may carry all these identifiers "at the same place" (in the data model next or close to each other, e.g.). In other cases the identifiers may be organized rather hierarchically in a data structure (this finally depends on the chosen structure of a function; it is esp. true for so-called “complex classes”, if a “parent” identifier is placed closer to the root of a function’s data structure than a “child” identifier).

If not stated otherwise in the Feature Type, the order given in the XSD structure denotes the hierarchical order of the identifiers. An overview of all identifiers can be found in section B.7.

Feature Types define which kind of Identifier is associated to a particular element (see section 3.4.2.2).

3.4.2 Identifier rules

3.4.2.1 General

All primary identifiers SHALL be set if not stated otherwise in the feature type. Non-primary identifiers SHOULD be set, if available and if not stated otherwise in the feature type.

Primary identifiers are those that are used to reference the classes functionality within the Class itself as well as from other features (e.g. measurementId within the Measurement class, referenced from ElectricalConnection) and are unique within the feature.

3.4.2.2 Identifier keywords

In the Feature Types the following keywords are used for identifiers:

- PRIMARY IDENTIFIER: A primary identifier is unique for the whole Feature. Primary identifiers are used only as "primary" or "top-level" identifier of a Function.. MAY be combined with further SUB IDENTIFIERS if needed to identify a list entry uniquely (for lists with more than one dimension). Note: A Feature Type may define two or more PRIMARY IDENTIFIERs. In this case the Feature Type describes a relation between the PRIMARY IDENTIFIERs.
- SUB IDENTIFIER: One or more SUB IDENTIFIERS MAY be used within a function to identify the further dimensions of list entries (if the list has more than one dimension (e.g. *parameterId* within *electricalConnectionParameterDescriptionListData*)).
- FOREIGN IDENTIFIER: Used to refer to other functionality on the same entity. It is not used to create further dimensions of list entries.
  Usually, a FOREIGN IDENTIFIER contains a value that is also set as PRIMARY IDENTIFIER in another Function. This is the most common relation between list entries of different Features. However, there can also be Use Cases that make use of FOREIGN IDENTIFIERs only, i.e. without an explicit instance of a list entry where a given FOREIGN IDENTIFIER value is set as PRIMARY IDENTIFIER. Such list entries with the same FOREIGN IDENTIFIER values express a relationship between these list entries as well.

The according rules only apply if the identifier keyword is written in capital letters.

3.4.2.3 Order of identifiers

The denoted order of the identifiers in the function description within the Feature Types determines the order of the identifiers.

There might be a variance of that in some classes where a SUB IDENTIFIER appears before a PRIMARY IDENTIFIER in the data model. In this case, the order is

- PRIMATY IDENTIFIER
- SUB IDENTIFIER
- [eventually further SUB IDENTIFIERS]

It MAY be allowed to skip identifiers (e.g. if a Feature Type defines three identifiers, the one in the middle MAY be omitted in some cases). If so, it is defined within the Feature Type.

3.4.2.4 timestamp identifier rules

If timestamp is used as identifier, some special rules need to be considered:

- If used as part of a notification or reply, it may be of type *xs:duration* or *xs:dateTime*.

- If used as part of a write operation:
    - If the time in timestamp shall express "now", this can be modelled in two ways:
        - Using *xs:duration*: duration of 0 seconds (e.g. PT0S)
        - Using *xs:dateTime*: *the current time and date*

      Due to the problems that may occur when transforming the different time representations, only relative time representation SHALL be used for write operations related to "now" where timestamp is an identifier!
    - For times others than "now", the client SHALL only use time representations (absolute or relative) supported by the server. The support currently may only be discovered by analyzing the messages sent by the server.
    - For history values that are writeable, only absolute time representation SHOULD be used!
- Normally a "binary comparison" of the identifier with the ones stored on the own device finds the appropriate data. But as timestamp can be modelled using absolute or relative time representation, a more complex comparison has to be chosen!
- The feature type may define further rules (i.e. restrictions) on the use of timestamp. This may affect its use as identifier as well as a restriction of its value range (e.g. if only *xs:duration* shall be used but not *xs:dateTime*).

3.4.2.5 Optional identifiers with default values

Some identifiers in some Feature Types (e.g. "billId" in the Feature Type "Bill") are defined with a default value and can be omitted in a message if the feature contains data of at maximum one identifier value of the respective list. In this case, default values for omitted identifiers apply. However, there are typical pitfalls implementers need to consider:

1. In general, identifiers SHALL be evaluated if they are present! Esp. extensions with new use cases/applications can easily lead to a situation where a formerly absent identifier element suddenly is set because there are now two or more identifier values used!
2. As soon as an Identifier is set within one list entry, all list entries SHALL set the according Identifier.
3. As mentioned above, default values are defined for some identifiers. This must be considered properly:
    a. We assume there is a Feature "A" with a Function that omits a particular PRIMARY or SUB IDENTIFIER in its messages as the feature server applies internally the default value for the list entries. Now we consider an other Feature "B" with a Function that contains a FOREIGN IDENTIFIER to refer to respective list entries of Feature "A"'s Function with the omitted identifier: In this case the FOREIGN IDENTIFIER in "B" must be set to this default value.
    b. The rule in item 2 above applies regardless of the value of the identifier. However, implentations should in particular consider cases briefly mentioned above, where suddenly two identifier values are used: In addition to the list elements with the default identifier value (usually "1") there are now also list elements with a new identifier value ("8", e.g.). In this case also for the formerly omitted Identifiers the corresponding list entries SHALL then contain the identifier explicitly.

A typical implementation error is to ignore the possibility of multiple identifier values and the implementation of wrong default values. In this case, client features may not be able to analyze Function (data) changes properly. This can especially occur for the case described above in item 1.

Generally, it is recommended to never omit identifiers if they are set to the default value, if possible, as it is more explicit and less prone to errors.

3.4.2.6 Optional identifiers without default values

A Feature Type may also define an optional Identifier that has no default value. The following rules apply:

1. If the Identifier is omitted:
   In this case the Identifier SHALL be omitted in all list entries. This means it is not possible then to submit or refer to only particular entries of this list.
2. If the Identifier is present:
   In this case the Identifier SHALL be present in all list entries. As usual, the Identifier is then used to identify particular list entries.

A typical implementation error is to ignore that a given list with such an Identifier may sometimes omit the Identifier and sometimes may have it set. Swtiching from one list type to the other always denotes a complete replacement of the former list by the new one.

3.5 Multidimensional data representation with "flat" list-based data model

Many applications need to work with multidimensional data. The functions of SPINE standard classes, on the other hand, usually are modelled as lists (i.e. one-dimensional objects) of rather flat data structures. However, this list concept is comparable to common table-based databases (SQL, e.g.) that are also frequently used for the management of multidimensional data. As there are similar concepts, most of the SPINE standard classes can as well be used for the modelling of multidimensional data. This section gives a brief example.

For this example we will use an excerpt of the function "measurementListData" of the SPINE Measurement class. It can take a list of "measurementData" elements, each comprising of the elements

- measurementId
- valueType
- timestamp
- value
- evaluationPeriod
- and some more elements

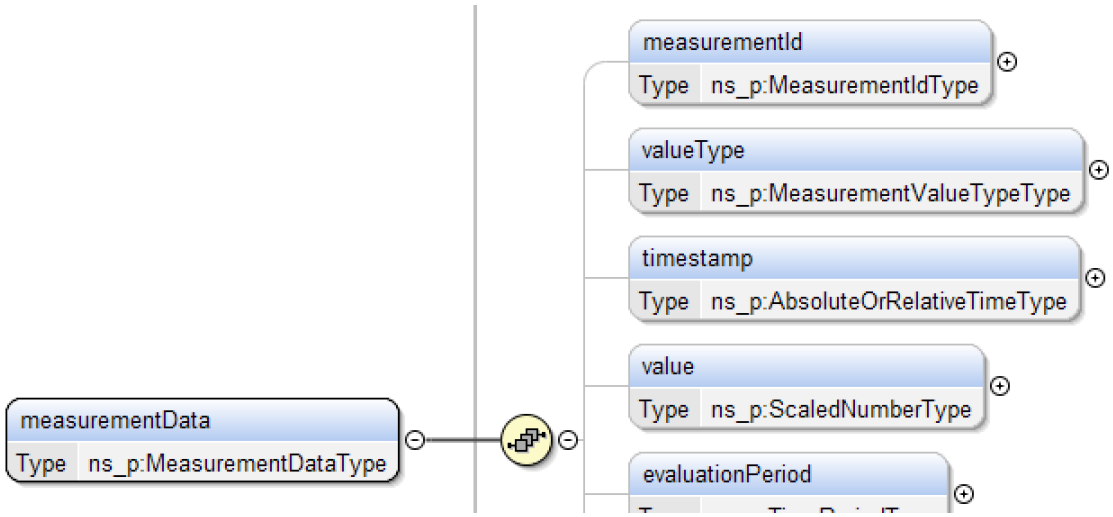


Figure 19: Excerpt of the list element "measurementData" of the SPINE function measurementListData

In the Measurement class definition, the first three elements "measurementId", "valueType", and "timestamp" are marked as so-called "identifiers". This means a particular measurementData instance is identified by these elements (just to give a comparison: In table-based databases such kinds of elements are frequently called "keys"). Mathematically, an identifier can be seen as the axe of a single dimension. The combination of distinct identifiers then creates a multidimensional space. At each position of this space the remaining elements ("value", "evaluationPeriod", etc.) can be placed with proper data:

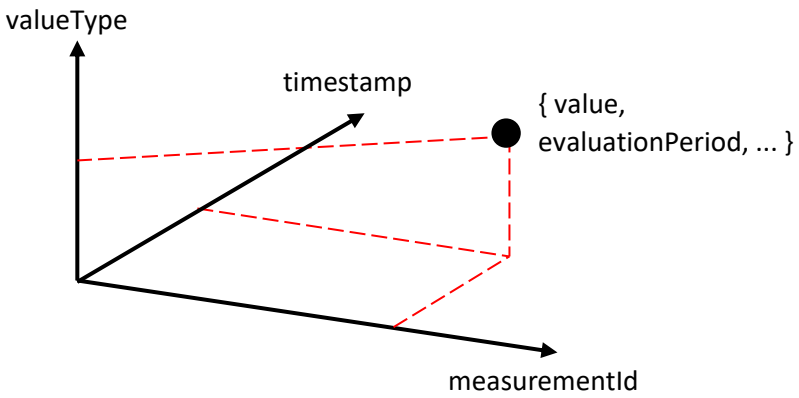


Figure 20: SPINE identifiers can create a multidimensional space of data.

As can be seen in Figure 19, the "position" in the multidimensional space is part of the measurementData structure. Therefore, a list of measurementData instances can build a complete multidimensional space with data. Among others, a particular value of measurementId can then contain data (value, evaluationPeriod, etc.) for multiple valueType values and multiple timestamp values. The following XML shows an example of a list of measurementData instances (for brevity the example just contains the element "value" and skips "evaluationPeriod" and further elements):

```
<measurementListData>
    <measurementData>
        <measurementId>1</measurementId>
        <valueType>value</valueType>
        <timestamp>2018-07-03T17:16:28.0Z</timestamp>
        <value>
```



```
            <number>15</number>
            <scale>-1</scale>
        </value>
    </measurementData>
    <measurementData>
        <measurementId>1</measurementId>
        <valueType>value</valueType>
        <timestamp>2018-07-03T17:16:33.0Z</timestamp>
        <value>
            <number>21</number>
            <scale>-1</scale>
        </value>
    </measurementData>
    <measurementData>
        <measurementId>1</measurementId>
        <valueType>minValue</valueType>
        <timestamp>2018-07-03T17:16:33.0Z</timestamp>
        <value>
            <number>1</number>
        </value>
    </measurementData>
    <measurementData>
        <measurementId>2</measurementId>
        <valueType>value</valueType>
        <timestamp>2018-07-03T17:16:33.0Z</timestamp>
        <value>
            <number>55</number>
        </value>
    </measurementData>
</measurementListData>
```

Semantically, this example XML models data for two different measurementId values, where

1. measurementId 1 has data for two different valueTypes:
    a. For valueType "value" there is data available for two different timestamps.
    b. For valueType "minValue" there is data available for only one timestamp.
2. measurementId 2 just has data for a single valueType = "value" at a particular timestamp.

Please note that SPINE identifiers have a "priority" within a SPINE class (or Feature Type, resp.), as explained in section 3.4. This means that a list-based SPINE function cannot be created in a way that identifiers of lower priority are used without the mandatory higher priority identifiers. With regards to the example above this means it would be invalid to create a measurementData instance where "valueType" is set but "measurementId" is absent. The following figure compares the flat list-based structure from Figure 19 with a virtual arrangement that considers the priorities of the identifiers.

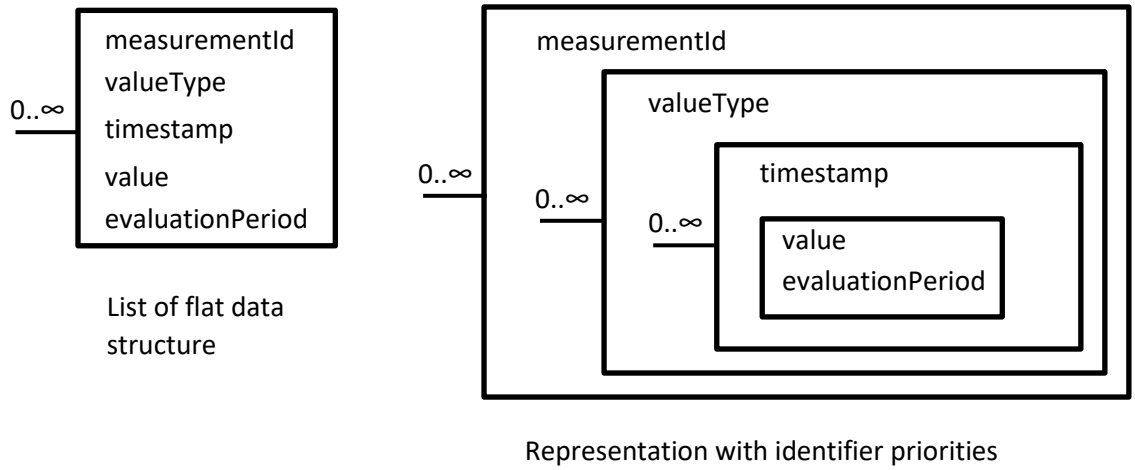


Figure 21: The flat data structure of Figure 19 can also be rearranged (virtually) to emphasize the priorities of the identifiers.

3.6 Restricted function exchange for list based functions

As defined in [ProtocolSpecification], section "Restricted function exchange with cmdOptions", in the communication between server and client not the complete server data has always to be considered but can be restricted considering some given rules as long as it is supported by the server (see [ProtocolSpecification], section "Minimum restricted function exchange support").

Further rules concerning the SPINE resources are defined in this section as default rules for all feature types that make use of SPINE standard class functions:

- If a server feature supports "write" operations with cmdControl "partial": The server feature SHALL accept received "xListData" containing at least one "xData". The server feature MAY accept xListData with more than one xData.
- If a server feature supports "write" operations with cmdControl "delete": The server feature SHALL accept the deletion request (using "<SELECTORS>") for at least a single "xData" of its "xListData". The server feature MAY support the deletion of more than one xData.
- If a server feature supports "read" operations with cmdControl "partial": The server feature SHALL accept the read request (using "<SELECTORS>") for at least a single "xData" of its "xListData". The server feature MAY return more than one xData.
- Complete function exchange SHALL be supported in any case (operations on the complete xListData).

Note: Feature types may still impose more restrictions than the above ones!

3.7 Relations between thematically connected SPINE classes

Some SPINE classes are thematically connected. The relations are indicated by the use of identifiers. To get a better understanding of the relations, the following diagrams visualize them. For details about the classes and the identifiers please consider the corresponding sections in this documentation. Relations are only possible within the same entity, not between different entities or devices.

3.7.1 Relations between Alarm / ElectricalConnection / HVAC / LoadControl / Measurement / Setpoint / SupplyCondition / TariffInformation / Threshold / TimeTable

The **Measurement** class is the central functionality of these related classes. It can be used to measure a lot of different measurands, describe the measurand and model constraints, etc.

If a setpoint shall be set, only the setpoint itself has to be stated in the **Setpoint** class. Via the *measurementId*, the type of measurand, that a setpoint is set for, is denoted. The period in time, this setpoint shall be applied can be specified in a time table from the **TimeTable** class. With the *timeTableId*, the reference is given.

The **HVAC** class references the **Setpoint** class as well as the **TimeTable** class.

If thresholds shall be set - either to activate an alarm if the threshold is exceeded or to do previously configured actions (implementation specific) - the **Threshold** class is used. The value that shall be measured to check if it is over- or underrun is the one from the Measurement class. The Threshold class does not reference the Measurement class, instead, in the Measurement class a relation function exists that lists all relevant thresholds.

If a threshold is exceeded, the **Alarm** class notifies all subscribed clients. To identify which threshold was exceeded, the Alarm class references the *thresholdId*.

The **SupplyCondition** class can reference to thresholds, too, to eventually generate events in case of exceeding some supply concerning limits.

Specifically for electrical measurands, the **ElectricalConnection** class describes all parameters needed. The values themselves are modelled with the Measurement class.

The **LoadControl** class enables a client to set limits on the server instance. To model further information about the measurand that is limited, the Measurement class is used.

The **TariffInformation** class references the **Measurement** class as well as the **TimeTable** class to model simple or complex tariff related functionality.

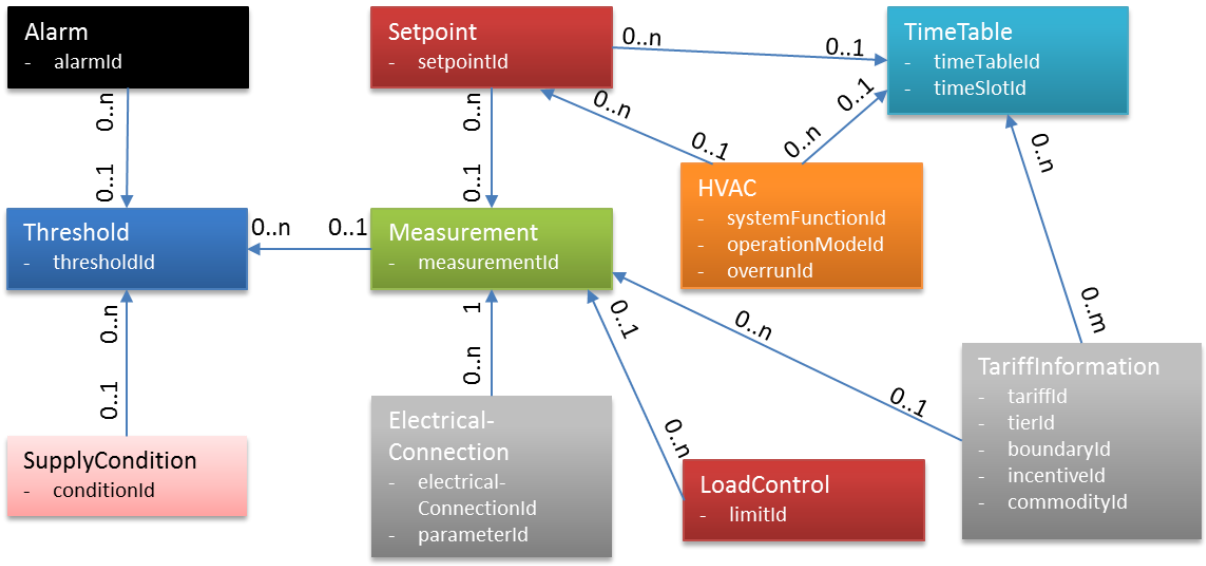


Figure 22: Relations between Alarm / ElectricalConnection / HVAC / LoadControl / Measurement / Setpoint / SupplyCondition / TariffInformation / Threshold / TimeTable

3.7.2 Relations between DirectControl / HVAC / LoadControl / OperatingConstraints / PowerSequences (SmartEnergManagementPs) / TaskManagement

The **PowerSequences** (or the **SmartEnergManagementPs** complex) class can be used to model sequences and slots, characterising a specific demand of energy that may be shifted in some range.

The **OperatingConstraints** class adds constraints to a power sequence to avoid misconfigurations of a device and models resume implications like additional energy or costs generated by pausing and then resuming some functionality. When using DirectControl (see below), the OperatingConstraints SHALL be considered, too, if linked with the same feature group (see section 2.1.4).

Pausing and reactivating devices can be done with the **DirectControl** class, which can relate to some power sequence that shall be controlled in a direct manner.

The **HVAC** class may reference power sequences that characterize specific functionality in more detail.

Another way the demand of a functionality may be affected is by some more or less external instruction (e.g. by the DSO) via the **LoadControl** class.

The **TaskManagement** class lists all the jobs from these classes (except for OperatingConstraints) and gives the needed relation in an extra function.

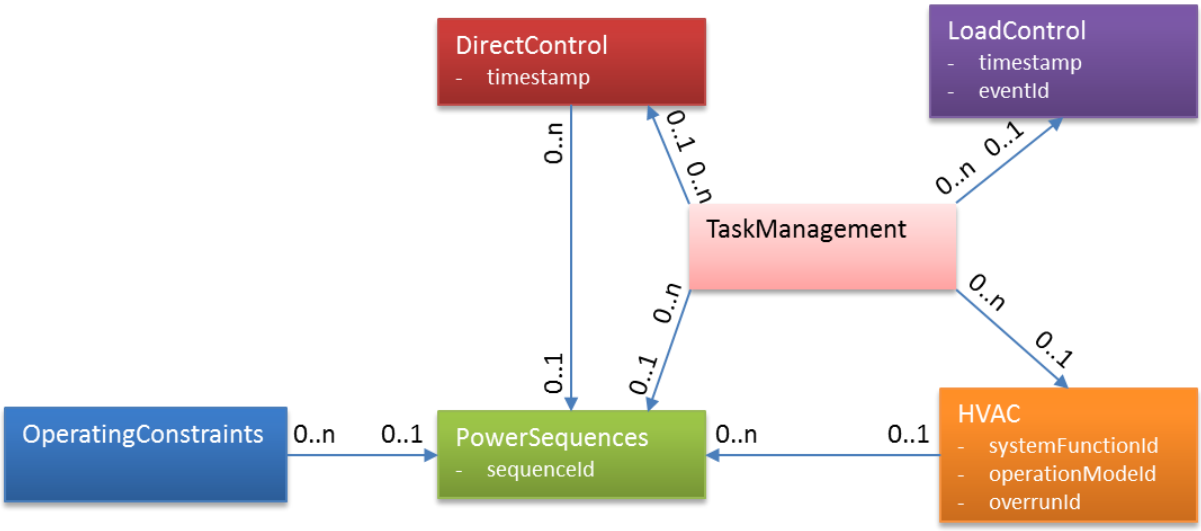


Figure 23: Relations between DirectControl / HVAC / LoadControl / OperatingConstraints / PowerSequences (SmartEnergManagementPs) / TaskManagement

3.8 On the use of "label" and "description"

In most functions with rather static information, elements called "label" and "description" are defined. These two elements MAY contain human readable strings (they are always optional) that SHALL NOT be used for automatisms (there are other elements for this purpose).

In general, the label element should contain a very short string which could be displayed below an icon, e.g. The description may contain a longer string that is only shown if the user clicks on a corresponding button or similar.

The following examples show how the strings can be used:

label: ActuatorLevel TestDevice
description: Device for testing SPINE compliance.

label: SmartPlug_1
description: SmartPlug Test Installation 01.04.2012

label: mains connection
description: 1-phase mains connection of the device

label: battery connection
description: Backup battery connection of the device

label: TempSensor1
description: Temperature Sensor 1 (Bedroom)

3.9 Empty elements as "tags"

In some cases elements shall only be used as "tags" without any content as some kind of boolean information (element is present = true, element is not present = false). This can be achieved by assigning the element a certain complex type that has no content (see section 3.10.1.1).

An example is the *DaysOfWeekType* (see section 3.10.2.10) used (e.g.) for the element *daysOfWeek* within the *start*- and *endTime* of the *timeTableListData* function (see section 5.3.29.2):

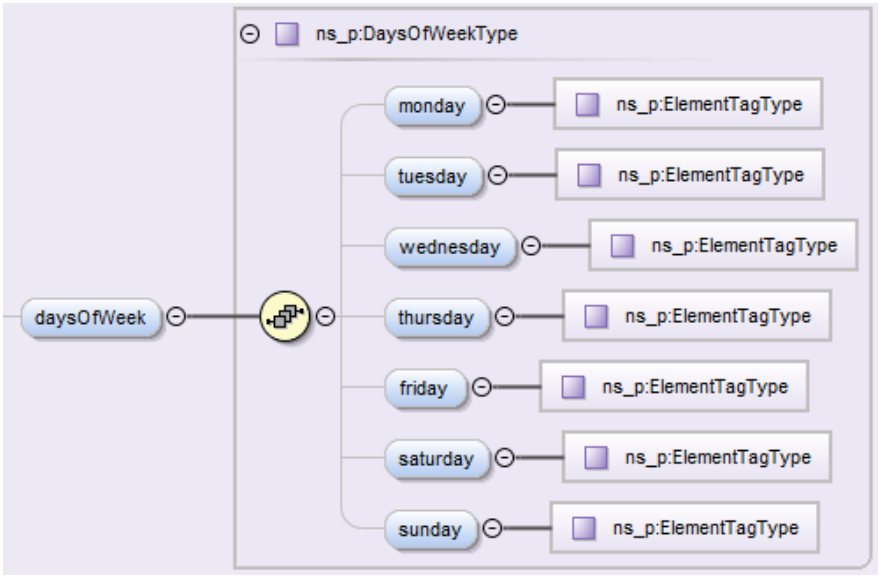


Figure 24: Empty elements example

An XML where (e.g.) the elements *monday* and *wednesday* are present would look like this:

```
...
<daysOfWeek>
    <monday/>
    <wednesday/>
</daysOfWeek>
...
```

In future versions of the SPINE specification, the type of an empty element may change and hence may include child elements, e.g. Please note that compatibility requirements and concepts always consider the content of a message (a concrete XML document, e.g.) and not the type names: The types are defined or used within XSD files, but type names are not explicitly mentioned in XML documents in this specification. Consequently, a subsequent version of the specification may well define the same (or properly extended) messages as a former version, but with partially renamed types as long as document/message compatibility is maintained. This concept permits also to extend formerly "empty" elements with child elements: If an element "foo" is defined as complex type with no further child element in version 1.0.0, then it is well permissive to assign "foo" a different complex type ("NewFooType", e.g.) with child elements in version 1.1.0.

3.10 Common data types

Some types are used by several classes. They are described in this section. For the exact naming and the complete list of all common data types please consider the underlying revision of the SPINE data model.

3.10.1 General

3.10.1.1 ElementTagType

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

3.10.1.1.1 Default rules for ElementTagType

If used as type for an element, the element SHALL have no further content in this SPINE version. Future SPINE versions may add content that SHALL be ignored by devices implementing the SPINE version at hand.

3.10.1.2 LabelType

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

3.10.1.2.1 Default rules for LabelType

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.

3.10.1.3 DescriptionType

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

3.10.1.3.1 Default rules for DescriptionType

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.

3.10.1.4 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.

3.10.1.5 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}]+
```

For details please see section 2.1.3. The maximum string length is limited to 256 characters.

3.10.1.5.1 Default rules for EnumExtendType

The values SHALL be marked as vendor specific by adding the IANA PEN code of the vendor ("i:") or some vendor name ("n:") surrounded by "_".
The IANA PEN SHOULD be used!

3.10.1.6 NumberType

Used to represent a numeric (integer) value. See also section 3.10.1.8!
A *simpleType* (*xs:long*).

3.10.1.7 ScaleType

Used to scale a numeric value. See also section 3.10.1.8!
A *simpleType* (*xs:short*).

3.10.1.8 ScaledNumberType

Numeric values (e.g. numerical values, setpoints, thresholds, etc.) use this number representation.

A complexType, containing the elements *number* (type: *NumberType*, see section 3.10.1.6) and *scale* (type: *ScaleType*, see section 3.10.1.7). See Figure 25.

3.10.1.8.1 Default rules for ScaledNumberType

Throughout this specification, this type and its elements are used as follows unless stated otherwise:

1. This type represents a real number (subsequently called “scaledNumber”) that is calculated from the type’s elements as follows:

   $$scaledNumber = number * 10^{scale}$$

2. If the element “scale” is omitted a default value of “0” SHALL be applied for “scale”. Then, item 1 applies, or (equivalently)

   $$scaledNumber = number$$

3. The element “number” SHALL be set.

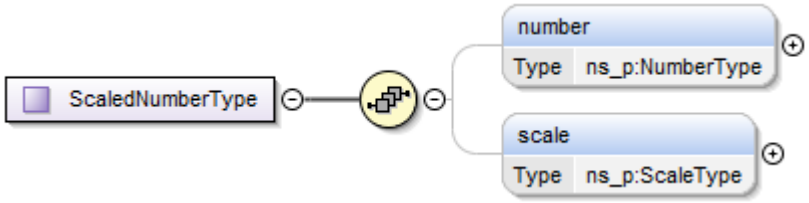


Figure 25: ScaledNumberType

3.10.1.9 ScaledNumberSetType

This type can be used to describe any sets of scaledNumbers. The set can have gaps compared to a simple value range and include many different values and value ranges.

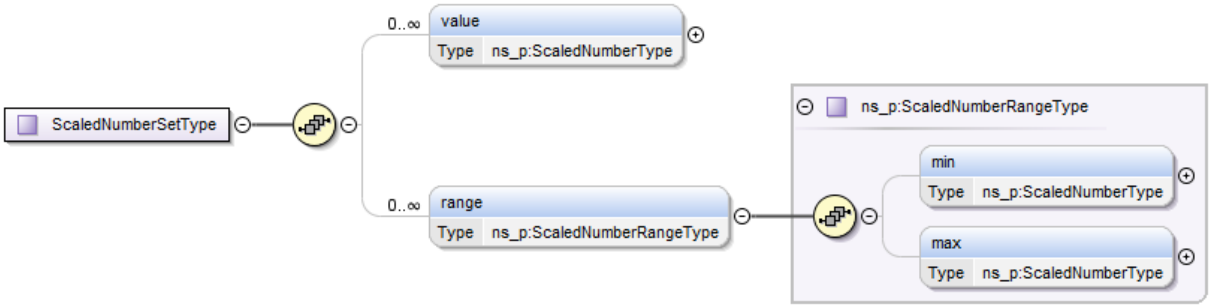


Figure 26: ScaledNumberSetType

Name	Type	Description
value (list)	*Common data type "ScaledNumberType". See section 3.10.1.8.*	Allows to describe numbers that are part of the set: Each "value" instance contributes its value as single number to the set.
range (list)	*Common data type "ScaledNumberRangeType". See section 3.10.1.10.*	Allows to describe number ranges that are part of the set. Each "range" instance contributes its value range to the set.

Table 1: ScaledNumberSetType

3.10.1.9.1 Default rules for ScaledNumberSetType

The set MAY have gaps.

3.10.1.10 ScaledNumberRangeType

This type can be used to describe a value range. All ScaledNumbers between min and max are supported in this case, at least if no further constraints apply, e.g. a value step size.

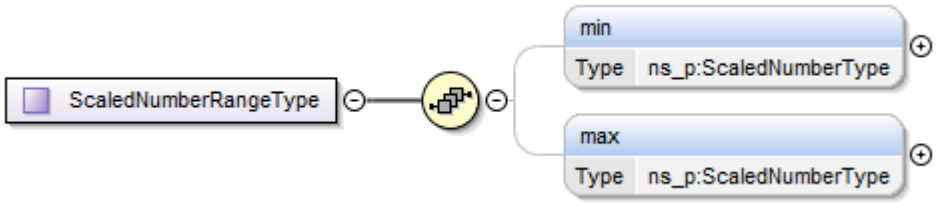


Figure 27: ScaledNumberRangeType

Name	Type	Description
min	*Common data type "ScaledNumberType". See section 3.10.1.8.*	The min element defines the lower bound of the ScaledNumberRange.
max	*Common data type "ScaledNumberType". See section 3.10.1.8.*	The max element defines the upper bound of the ScaledNumberRange.

Table 2: ScaledNumberRangeType

3.10.1.11 CommodityTypeType

Union of:

- CommodityTypeEnumType (see section 3.10.1.12)
- EnumExtendType (see section 3.10.1.5)

3.10.1.12 CommodityTypeEnumType

The commodity, used for measurements, pricing, etc.

A *simpleType* (restricted string), containing enumerations for the commodity.

Value	Description
electricity	Electricity (AC or DC).
gas	Gas, as used by a gas heater.
oil	Oil, as used by an oil heater.
water	(Drinking) water.
wasteWater	(Domestic) waste water.
domesticHotWater	Domestric hot (drinking, e.g.) water.
heatingWater	Water circulating through the heating system of a building.
steam	Steam, as used in (e.g.) a steam boiler or within a coal-fired power station.
heat	Heat (e.g. in the form of hot water, hot air, etc.).
coolingLoad	Cooling load used to cool down goods or a room / building.
air	Air, as commonly available outside (no special gas or chemical compound).

Table 3: Enumeration CommodityTypeEnumType

3.10.1.13 EnergyDirectionType

Union of:

- EnergyDirectionEnumType (see section 3.10.1.14)
- EnumExtendType (see section 3.10.1.5)

3.10.1.14 EnergyDirectionEnumType

The energy direction. Can be applied for other measurands than energy, too (such as power or current).

A *simpleType* (restricted string), containing the enumerations *consume* and *produce*.

3.10.1.14.1 Default rules for EnergyDirectionEnumType

If this type is used for an Element, the related data SHALL be interpreted in a way as defined by the rules below.

Value	Rule
consume	Positive values SHALL be interpreted as consumption; negative values SHALL be interpreted as production.
produce	Positive values SHALL be interpreted as production; negative values SHALL be interpreted as consumption.

Table 4: Enumeration EnergyDirectionEnumType value rules

3.10.1.15 EnergyModeType

Union of:

- EnergyModeEnumType (see section 3.10.1.16)
- EnumExtendType (see section 3.10.1.5)

3.10.1.16 EnergyModeEnumType

Different modes for energy demand.

A *simpleType* (restricted string), containing the enumerations for the energy mode.

Value	Description
consume	Used in cases where the functionality consumes energy.
produce	Used in cases where the functionality produces energy.
idle	Used in cases where the functionality neither consumes nor produces energy.
auto	Used in cases where the funcionality may consume or produce energy (or none of both), decided on its internal business logic.

Table 5: Enumeration EnergyModeEnumType

3.10.1.16.1 Default rules for EnergyModeEnumType

Value	Rule
consume	The related functionality SHALL only consume energy.

produce	The related functionality SHALL only produce energy.
idle	The related functionality SHALL neither consume or produce energy.
auto	The related functionality MAY consume or produce energy or be in idle-mode based on its own decision.

Table 6: Enumeration EnergyModeEnumType value rules

3.10.1.17 UnitOfMeasurementType

Union of:

- UnitOfMeasurementEnumType (see section 3.10.1.18)
- EnumExtendType (see section 3.10.1.5)

3.10.1.18 UnitOfMeasurementEnumType

Unit, used for measurements, etc.

A *simpleType* (restricted *string*), containing several units.

In general the following format is used to model mathematical operations as strings:

x/y ≡ x divided by y (i.e. x ÷ y)

x^y ≡ x raised to the power of y (i.e. x^y)

x_y ≡ x multiplied by y (i.e. x * y)

Examples:

m^3 ≡ m^3

m^3/h ≡ (m^3) ÷ h

m/s^2 ≡ m ÷ (s^2)

kg_m^2 ≡ kg * (m^2)

Unit Name	**Unit Description**
unknown	Unknown unit
1	1 (unitless)
m	metre (SI unit, used for length)
kg	kilogram (SI unit, used for mass)
s	second (SI unit, used for time)
A	ampere (SI unit, used for electric current)
K	kelvin (SI unit, used for temperature)
mol	mole (SI unit, used for the amount of a substance)
cd	candela (SI unit, used for luminous intensity)
V	volt (used for voltage)
W	watt (used for power)
Wh	watt hour (used for energy)
VA	volt-ampere (used for apparent power)
VAh	volt-ampere hour (used for apparent energy)
var	volt-ampere reactive (used for reactive power)
varh	volt-ampere-reactive hours (used for reactive energy)
degC	degree Celsius (°C, used for temperature)
degF	degree Fahrenheit (°F, used for temperature)

Lm	lumen (used for total quantity of visible light emitted by a source)
lx	lux (used for illuminance and luminous emittance, measuring luminous flux per unit area)
Ohm	ohm (used for electrical resistance)
Hz	hertz (used for frequency)
dB	decibel (used to express the ratio of one value of a physical property to another on a logarithmic scale)
dBm	decibel with a reference value of one milliwatt
pct	percentage
ppm	parts-per-million
l	litre (used for volume)
l/s	litre per second (used for flow rate)
l/h	litre per hour (used for flow rate)
deg	degree of arc (used for planar angle; a full rotation is 360 degrees)
rad	radian (used for planar angle; a full rotation is 2π)
rad/s	radian per second (used for rotational speed (angular velocity))
sr	steradian or square radian (used for solid angles in three-dimensional geometry)
Gy	gray (used for ionizing radiation dose)
Bq	becquerel (used for radioactivity)
Bq/m^3	becquerel per cubic metre (used for specific radioactive activity in air and gases)
Sv	sievert (used for ionizing radiation dose)
Rd	rutherford (used for radioactive decay)
C	coulomb (used for electric charge)
F	farad (used for electrical capacitance)
H	henry (used for electrical inductance)
J	joule (used for energy)
N	newton (used for force)
N_m	newton metre (used for torque (also called *moment*)
N_s	newton second (used for impulse)
Wb	weber (used for magnetic flux)
T	tesla (used for magnetic flux density)
Pa	pascal (used for pressure)
bar	bar (used for pressure)
atm	standard atmosphere (used for pressure)
psi	pound-force per square inch (used for pressure)
mmHg	millimetre of mercury (used for pressure)
m^2	square metre (used for area)
m^3	cubic metre (used for volume)
m^3/h	cubic metre per hour (used for (volumetric) flow rate)
m/s	metre per second (used for speed (scalar) and velocity (vector quantity which specifies both magnitude and a specific direction))
m/s^2	metre per second squared (used for acceleration)
m^3/s	cubic metre per second (used for (volumetric) flow rate)
m/m^3	metre per cubic metre (used for fuel efficiency)
kg/m^3	kilogram per cubic metre (used for density)

kg_m	kilogram metre (used for payload-distance)
m^2/s	square metre per second (used for kinematic viscosity)
W/m_K	watts per meter-kelvin, W/(m·K) (used for thermal conductivity)
J/K	joule per kelvin (used for heat capacity (or thermal capacity) and (thermodynamic) entropy)
1/s	1 per second (used for frequency)
W/m^2	watt per square metre (used for heat flux density, irradiance)
J/m^2	joule per square metre (used for surface tension, stiffness)
S	siemens (used for electric conductance, electric susceptance and electric admittance)
S/m	siemens per metre (used for electrical conductivity)
K/s	kelvin per second (used for heating rate)
Pa/s	pascal per second (used for power density)
J/kg_K	joule per kilogram kelvin, J/(kg·K) (used for specific heat capacity)
Vs	volt-second (used for magnetic flux)
V/m	volt per metre (used for electric field strength)
V/Hz	volt per hertz (used for magnetic flux)
As	ampere second (used for electric charge)
A/m	ampere per metre (used for magnetization, magnetic field strength)
Hz/s	hertz per second (used for frequency drift)
kg/s	kilogram per second (used for mass flow rate)
kg_m^2	kilogram square metre (used for moment of inertia)
J/Wh	joule per watt-hour (used for heat rate)
W/s	watt per second (used for ramp rate)
ft^3	cubic foot (used for volume)
ft^3/h	cubic foot per hour (used for (volumetric) flow rate)
ccf	centum cubic feet = 100 ft³ (used for volume)
ccf/h	centum cubic feet per hour = 100 ft³/h (used for (volumetric) flow rate)
US.liq.gal	US (liquid) gallon (used for (fluid) capacity, volume)
US.liq.gal/h	US (liquid) gallon per hour (used for flow rate)
Imp.gal	imperial (UK) gallon
Imp.gal/h	imperial (UK) gallon per hour (used for flow rate)
Btu	british thermal unit (used for heat)
Btu/h	british thermal unit per hour (used for power)
Ah	ampere hour (used for electric charge)
kg/Wh	kilogram per watt-hour (used for (e.g.) fuel economy or CO2 emission)

Table 7: Enumeration UnitOfMeasurementEnumType

3.10.1.19 CurrencyType

Union of:

- CurrencyEnumType (see section 3.10.1.20)
- EnumExtendType (see section 3.10.1.5)

1816 ***3.10.1.20 CurrencyEnumType***

1817 Currency as used by price information, e.g.

1818 A *simpleType* (restricted *string*), containing several currencies (as defined in ISO 4217:2015).

AED	BRL	CZK	HNL	KYD	MXV	RUB	TND	XBA
AFN	BSD	DJF	HRK	KZT	MYR	RWF	TOP	XBB
ALL	BTN	DKK	HTG	LAK	MZN	SAR	TRY	XBC
AMD	BWP	DOP	HUF	LBP	NAD	SBD	TTD	XBD
ANG	BYR	DZD	IDR	LKR	NGN	SCR	TWD	XCD
AOA	BZD	EGP	ILS	LRD	NIO	SDG	TZS	XDR
ARS	CAD	ERN	INR	LSL	NOK	SEK	UAH	XOF
AUD	CDF	ETB	IQD	LYD	NPR	SGD	UGX	XPD
AWG	CHE	EUR	IRR	MAD	NZD	SHP	USD	XPF
AZN	CHF	FJD	ISK	MDL	OMR	SLL	USN	XPT
BAM	CHW	FKP	JMD	MGA	PAB	SOS	UYI	XSU
BBD	CLF	GBP	JOD	MKD	PEN	SRD	UYU	XTS
BDT	CLP	GEL	JPY	MMK	PGK	SSP	UZS	XUA
BGN	CNY	GHS	KES	MNT	PHP	STD	VEF	XXX
BHD	COP	GIP	KGS	MOP	PKR	SVC	VND	YER
BIF	COU	GMD	KHR	MRO	PLN	SYP	VUV	ZAR
BMD	CRC	GNF	KMF	MUR	PYG	SZL	WST	ZMW
BND	CUC	GTQ	KPW	MVR	QAR	THB	XAF	ZWL
BOB	CUP	GYD	KRW	MWK	RON	TJS	XAG	
BOV	CVE	HKD	KWD	MXN	RSD	TMT	XAU	

1819 *Table 8: Enumeration CurrencyEnumType*

1820

1821 ***3.10.1.21 ScopeTypeType***

1822 Union of:

1823 - ScopeTypeEnumType (see section 3.10.1.22)

1824 - EnumExtendType (see section 3.10.1.5)

1825

1826 ***3.10.1.22 ScopeTypeEnumType***

1827 Some feature types use scope types for specific functionalities. These are listed in the

1828 ScopeTypeEnumType.

1829 A *simpleType* (restricted string), containing enumerations for the scope type.

Scope Type	**Description**
ac	Alternating current.
acCosPhiGrid	Alternating current cosine phi of the electricity grid.
acCurrentA	Alternating current current on phase A.
acCurrentB	Alternating current current on phase B.
acCurrentC	Alternating current current on phase C.
acFrequencyGrid	Alternating current frequency of the electricity grid.
acPowerA	Alternating current power on phase A.
acPowerB	Alternating current power on phase B.
acPowerC	Alternating current power on phase C.
acPowerLimitPct	Alternating current power limit in percentage.

acPowerTotal	Alternating current power total (all phases).
acVoltageA	Alternating current voltage on phase A.
acVoltageB	Alternating current voltage on phase B.
acVoltageC	Alternating current voltage on phase C.
acYieldDay	Alternating current yield one (this) day.
acYieldTotal	Alternating current yield total (since last reset).
dcCurrent	Direct current.
dcPower	Direct current power.
dcString1	Direct current of string 1 (no info here whether voltage/current/etc.!).
dcString2	Direct current of string 2 (no info here whether voltage/current/etc.!).
dcString3	Direct current of string 3 (no info here whether voltage/current/etc.!).
dcString4	Direct current of string 4 (no info here whether voltage/current/etc.!).
dcString5	Direct current of string 5 (no info here whether voltage/current/etc.!).
dcString6	Direct current of string 6 (no info here whether voltage/current/etc.!).
dcTotal	Direct current of all strings (no info here whether voltage/current/etc.!).
dcVoltage	Direct current voltage.
dhwTemperature	Domestic hot water temperature (for example in a domestic hot water storage).
flowTemperature	Flow temperature (for example of a heating circuit).
outsideAirTemperature	Temperature of the outside air.
returnTemperature	Return (backflow) temperature (for example of a heating circuit).
roomAirTemperature	Air temperature inside a room.
charge	(Battery-)Charge-related values.
stateOfCharge	State-of-charge-related values of a (battery) appliance.
discharge	(Battery-)Discharge-related values.
gridConsumption	Values related to the consumption from the electricity grid.
gridFeedIn	Values related to the electricity grid feed in.
selfConsumption	Self consumption (optimization).
overloadProtection	Overload protection of the local grid, e.g.
acPower	AC power.
acEnergy	AC energy.
acCurrent	AC current.
acVoltage	AC voltage.
batteryControl	Battery control (e.g. charge, discharge, idle).
simpleIncentiveTable	Simple incentive table (limiting the possible variations of the Feature Type to an easy to implement solution).

Table 9: Enumeration ScopeTypeEnumType

3.10.1.23 RoleType

If a role has to be stated, this type is used.

A *simpleType* (restricted *string*), containing the enumeration values *client*, *server* or *special*.

3.10.1.24 FeatureGroupType

Use for connecting related features (see section 2.1.4).

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.

3.10.1.25 DeviceTypeType

Union of:
- DeviceTypeEnumType (see section 3.10.1.26)
- EnumExtendType (see section 3.10.1.5)

3.10.1.26 DeviceTypeEnumType

The type of the device.
A *simpleType* (restricted *string*), containing the device types defined by the EEBus Initiative e.V. See Table 334 for all enumerations.

3.10.1.27 EntityTypeType

Union of:
- EntityTypeEnumType (see section 3.10.1.28)
- EnumExtendType (see section 3.10.1.5)

3.10.1.28 EntityTypeEnumType

The type of the entity.
A *simpleType* (restricted *string*), containing the entity types defined by the EEBus Initiative e.V. See Table 335 for all enumerations.

3.10.1.29 FeatureTypeType

Union of:
- FeatureTypeEnumType (see section 3.10.1.30)
- EnumExtendType (see section 3.10.1.5)

3.10.1.30 FeatureTypeEnumType

The type of the feature.
A *simpleType* (restricted *string*), containing the feature types defined by the EEBus Initiative e.V. See Table 336 for all enumerations.

3.10.1.31 FeatureSpecificUsageType

Deprecated.

3.10.1.32 FeatureHvacSpecificUsageEnumType

Deprecated.

3.10.1.33 FeatureMeasurementSpecificUsageEnumType

Deprecated.

3.10.1.34 FeatureSetpointSpecificUsageEnumType

Deprecated.

3.10.1.35 FunctionType

Union of:
- FunctionEnumType (see section 3.10.1.36)
- EnumExtendType (see section 3.10.1.5)

3.10.1.36 FunctionEnumType

A list of all SPINE functions defined by the EEBus Initiative e.V.
A *simpleType* (restricted *string*), containing enumerations of all SPINE functions. Please consider the class sections where all SPINE functions are specified.

3.10.1.37 PossibleOperationsClassifierType

If the element partial is used, filtering the further content is permitted. Otherwise only full operations (no filtering) are allowed.
A *complexType*, containing the element *partial* (type: *ElementTagType*, see section 3.10.1.1).

3.10.1.38 PossibleOperationsReadType

If read operations are permitted, this element is set. Elements from the base type can be used (see the corresponding section).
A *complexType* with the base type *PossibleOperationsClassifierType* (see section 3.10.1.37).

3.10.1.39 PossibleOperationsWriteType

If write operations are permitted, this element is set. Elements from the base type can be used (see the corresponding section).
A *complexType* with the base type *PossibleOperationsClassifierType* (see section 3.10.1.37).

3.10.1.40 PossibleOperationsType

Defines which operations on a function are allowed.

A *complexType*, containing the elements *read* (type: *PossibleOperationsReadType*, see section 3.10.1.38) and *write* (type: *PossibleOperationsWriteType*, see section 3.10.1.39).

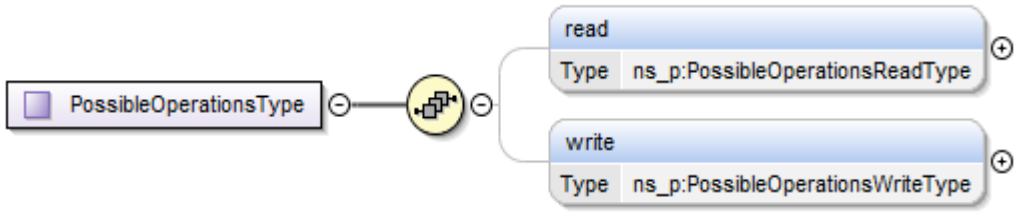


Figure 28: PossibleOperationsType

3.10.1.41 FunctionPropertyType

If a function together with its possible operations will be denoted, this complex type can be used.

A *complexType*, containing the two elements *function* (type: FunctionType, see section 3.10.1.35) and *possibleOprations* (type: PossibleOperationsType, see section 3.10.1.40).

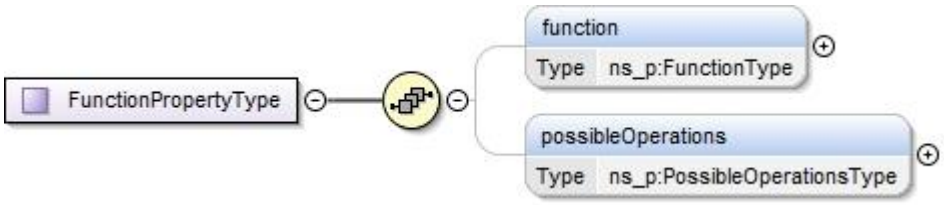


Figure 29: FunctionPropertyType

3.10.1.42 MaxResponseDelayType

Used to represent the maxResponseDelay.

A *simpleType* (*xs:duration*).

3.10.2 Time-related

3.10.2.1 TimePeriodType

The typically used type for all time periods in the SPINE data model.

A *complexType* containing the two elements *startTime* and *endTime* (both of type *AbsoluteOrRelativeTimeType*, see section 3.10.2.3).

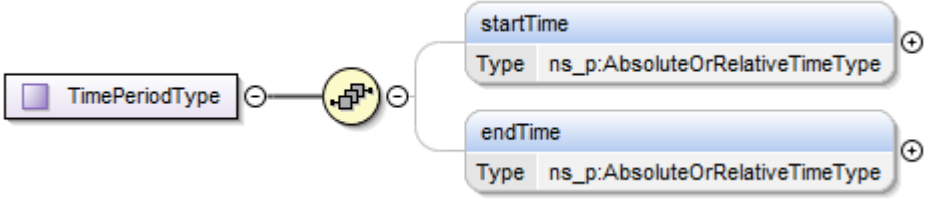


Figure 30: TimePeriodType

Name	Type	Description
startTime	*Common data type "AbsoluteOrRelativeTimeType". See section 3.10.2.3.*	The startTime element defines the start time.
endTime	*Common data type "AbsoluteOrRelativeTimeType". See section 3.10.2.3.*	The endTime element defines the end time.

Table 10: TimePeriodType

3.10.2.2 TimestampIntervalType

The typically used type for all timestamp intervals in selectors in the SPINE data model.
A *complexType*, containing the two elements *startTime* and *endTime* (both of type *AbsoluteOrRelativeTimeType*, see section 3.10.2.3).

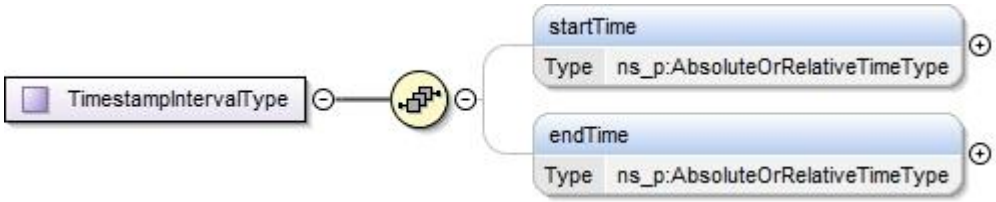


Figure 31: TimestampIntervalType

Name	Type	Description
startTime	*Common data type "AbsoluteOrRelativeTimeType". See section 3.10.2.3.*	The startTime element defines the start time.
endTime	*Common data type "AbsoluteOrRelativeTimeType". See section 3.10.2.3.*	The endTime element defines the end time.

Table 11: TimestampIntervalType

3.10.2.3 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 (see section 3.2).
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".

3.10.2.4 RecurringIntervalType

Union of:
- RecurringIntervalEnumType (see section 3.10.2.5)
- EnumExtendType (see section 3.10.1.5)

3.10.2.5 RecurringIntervalEnumType

Recurring interval length make use of this type.
A *simpleType* (restricted *string*), containing enumerations for the recurring interval length.

yearly
monthly
weekly
daily
hourly
everyMinute
everySecond

Table 12: RecurringIntervalEnumType

3.10.2.6 MonthType

All month names are included in this type.
A *simpleType* (restricted *string*), containing enumerations for the months.

january	july
february	august
march	september
april	october
may	november
june	december

Table 13: MonthType

3.10.2.7 DayOfMonthType

This type is used for stating the day of the month.
A *simpleType* (restricted *xs:unsignedByte*), with the allowed value range from "1" to "31".

3.10.2.8 CalendarWeekType

This type is used for stating the calendar week.
A *simpleType* (restricted *xs:unsignedByte*), with the allowed value range from "1" to "53".

3.10.2.9 DayOfWeekType

This type contains all weekdays as enumeration (only one selectable).
A *simpleType* (restricted *xs:string*), containing enumerations for the weekdays.

monday
tuesday
wednesday
thursday
friday
saturday
sunday

Table 14: DayOfWeekType

3.10.2.10 DaysOfWeekType

This type contains all weekdays as flags (more than one selectable).

A *complexType*, containing the elements monday, tuesday, wednesday, thursday, friday, saturday, sunday.

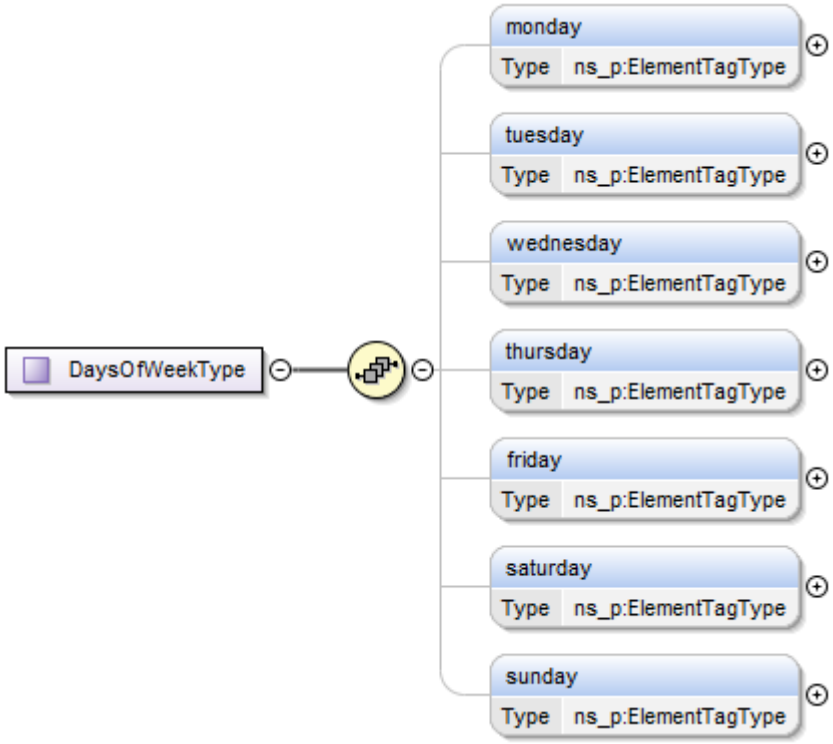


Figure 32: DaysOfWeekType

3.10.2.11 OccurrenceType

Union of:

- OccurrenceEnumType (see section 3.10.2.12)
- EnumExtendType (see section 3.10.1.5)

3.10.2.12 OccurrenceEnumType

If the occurrence of something will be modelled, this type can be used.

A *simpleType* (restricted *string*), containing enumerations for some occurrences.

Value	Description
first	First occurrence of the related event where this type is used.
second	Second occurrence of the related event where this type is used.
third	Third occurrence of the related event where this type is used.
fourth	Fourth occurrence of the related event where this type is used.
last	Last occurrence of the related event where this type is used.

Table 15: OccurrenceEnumType

3.10.2.13 AbsoluteOrRecurringTimeType

If absolute (specific point in time) or recurring (relative point in time that is repeated) times will be modelled, this type is used.

A complexType, containing the elements described in Table 16.

The typical usage is that either *dateTime* is denoted or one of the following combinations:

- month + dayOfMonth + time (specific day once per year)
- dayOfMonth + time (specific day once per month)
- calendarWeek + daysOfWeek + time (specific day of week once per year)
- month + dayOfWeekOccurrence + daysOfWeek + time (first, second, ... occurrence of specific day of week in a specific month once per year, e.g. every second Thursday of July)
- daysOfWeek + time (specific day of week once per week)
- time (every day at a specific time)

The *dayOfWeekOccurrence* element only makes sense in combination with *daysOfWeek* where *calenderWeek* is not set. It defines which week in a month was meant. For example, the *first* week, or the *second*, etc.

Instead of only one day, combinations of several days may be stated in the element *daysOfWeek* (e.g. "monday" and "tuesday" and "friday" or "saturday" and "sunday").

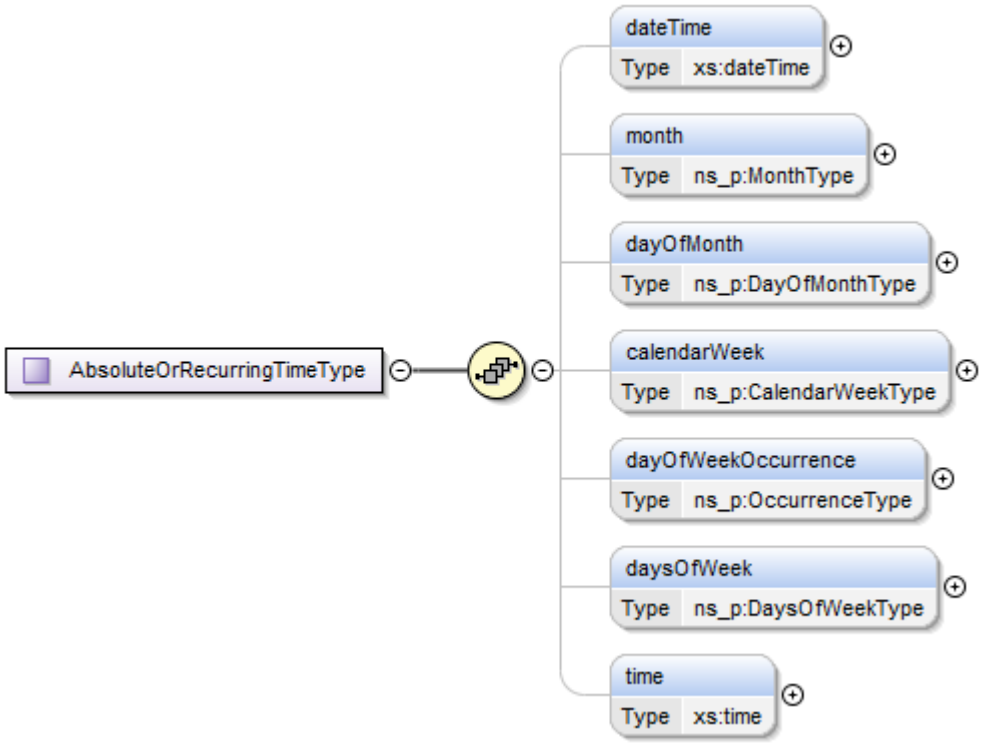


Figure 33: AbsoluteOrRecurringTimeType

Element	Type	Description
dateTime	*Common data type "RecurringIntervalType". See section 3.10.2.4.*	An absolute time.
month	*Common data type "MonthType". See section 3.10.2.6.*	The month information for the recurring time (e.g. "January").
dayOfMonth	*Common data type "DayOfMonthType". See section 3.10.2.7.*	The day of month information for the recurring time (e.g. "25").
calendarWeek	*Common data type "CalenderWeekType". See section 3.10.2.8.*	The calendar week information for the recurring time (e.g. "50").
dayOfWeekOccurrence	*Common data type "OccurrenceType". See section 3.10.2.11.*	The day of week occurrence information for the recurring time (e.g. "second", if the second occurrence of this weekday within a month is chosen).
daysOfWeek	*Common data type "DaysOfWeekType". See section 3.10.2.10.*	The days of week information for the recurring time (e.g. the elements "monday" and "wednesday" could be set).
time	xs:time (W3C standard type)	The time information for the recurring time (e.g. "18:15:31Z").
relative	xs:duration (W3C standard type)	The relative time information for the recurring time (e.g. "PT23H15M20S").

Table 16: AbsoluteOrRecurringTimeType

3.10.2.14 RecurrenceInformationType

If the recurrence of some time information will be modelled, this type is used.

A complexType, containing the elements described in Table 17.

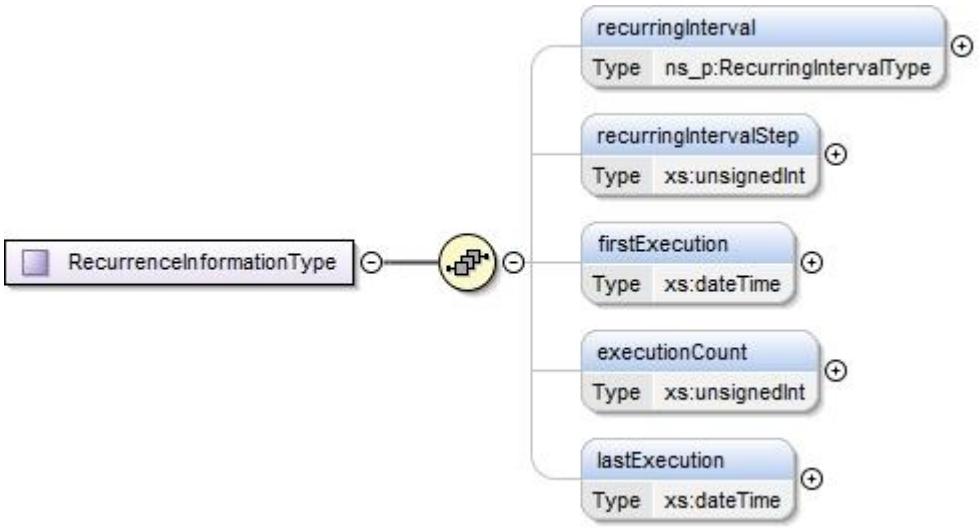


Figure 34: RecurrenceInformationType

Element	Type	Description
recurringInterval	*Common data type "RecurringIntervalType". See section 3.10.2.4.*	With the *recurringInterval* the frequency of the recurrence is defined (e.g. "weekly").
recurringIntervalStep	xs:unsignedInt (W3C standard type)	If not every occurrence will be active, the *recurringIntervalStep* is used (e.g. if only every second occurrence will be active, *recurringIntervalStep* would have a value of "2").
firstExecution	xs:dateTime (W3C standard type)	If the recurring interval will not start with the next occurrence, the *firstExecution* element is used to specify this first time.
executionCount	xs:unsignedInt (W3C standard type)	To end the recurring execution, the *executionCount* can be used (e.g. after "10" occurrences, the interval ends).
lastExecution	xs:dateTime (W3C standard type)	To end the recurring execution, the *lastExecution* can be used, stating a specific point in time after that the interval will be not active any more (the *lastExecution* point in time does not have to be an interval occurrence).

Table 17: RecurrenceInformationType

3.10.3 Address-related

3.10.3.1 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}]+
```


For details please see [ProtocolSpecification], section "Rules for devices". The maximum string length is limited to 256 characters.

3.10.3.2 AddressEntityType

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

3.10.3.3 AddressFeatureType

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

3.10.3.4 DeviceAddressType

Address type including only the device part.
A *complexType* with the element *device* (type: *AddressDeviceType*, see section 3.10.3.1).

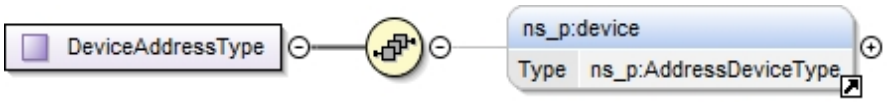


Figure 35: DeviceAddressType

3.10.3.5 EntityAddressType

Address type including the device part and a list of entities.
A *complexType* with the base type *DeviceAddressType* (see section 3.10.3.4) and a further list of element *entity* (type: *AddressEntityType*, see section 3.10.3.2).

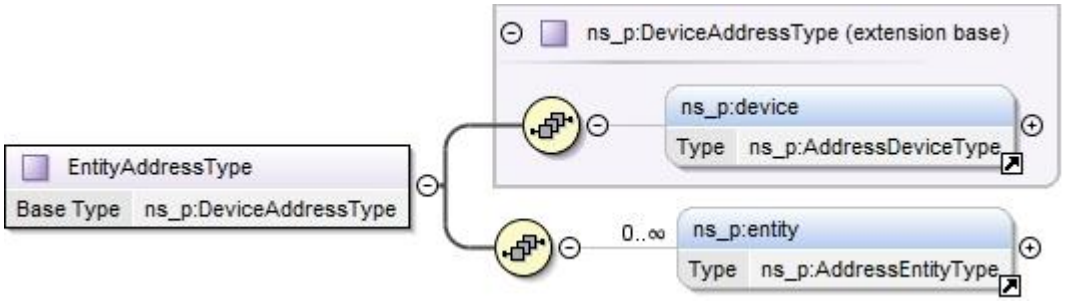


Figure 36: EntityAddressType

3.10.3.6 FeatureAddressType

Address type including the device part, a list of entities and the feature part.
A *complexType* with the base type *EntityAddressType* (see section 3.10.3.5) and a further element *feature* (type: *AddressFeatureType*, see section 3.10.3.3).

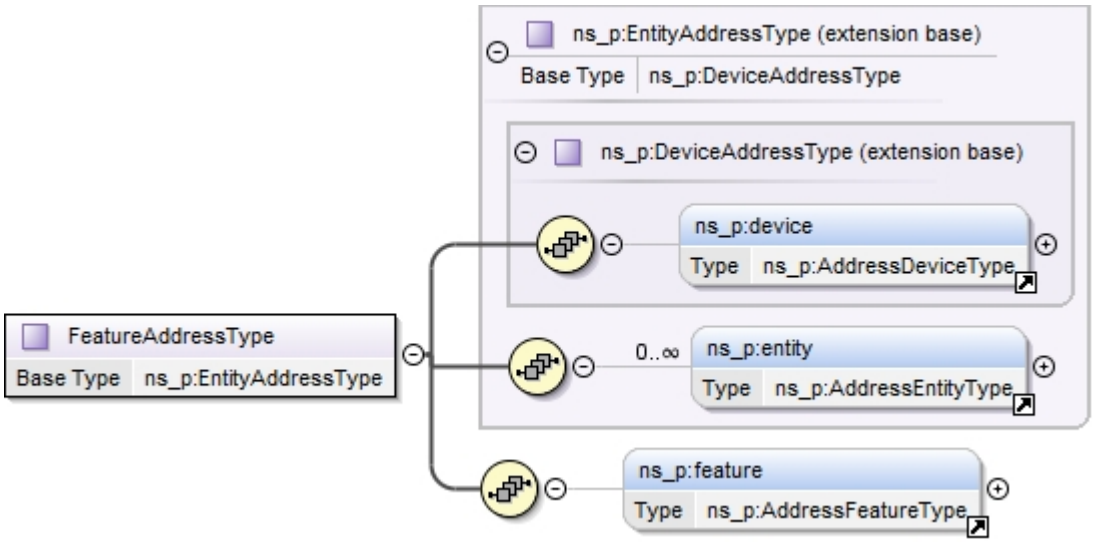


Figure 37: FeatureAddressType

3.11 Result

Note: This section discusses the SPINE function “resultData” and NOT the classifier “result”!

The result function concept is used for modelling acknowledges or errors for messages exchanged between communications partners. It is intended for these use cases:

- Notification of an error related to another function (e.g. as a reaction to a failed write operation).
- Substitution of a "regular reply" with an "error response". This applies if it is not possible to create a valid and proper reply that belongs to the function group of the corresponding read function.

Further information about the usage of result functions can be found in [ProtocolSpecification], section "Acknowledgement message".

The result function concept is modelled similar to (standard) classes. Its only function is called "resultData".

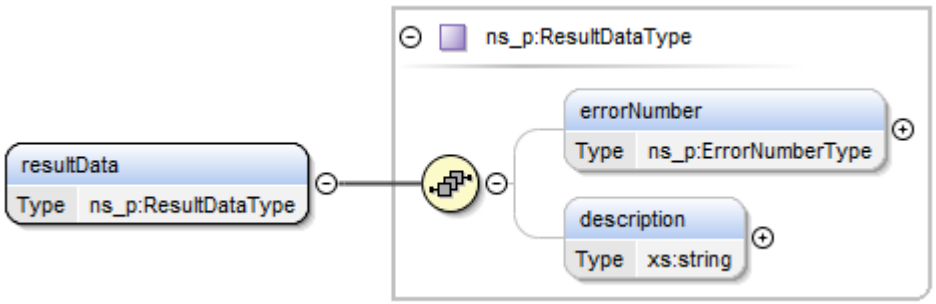


Figure 38: resultData function overview

The elements errorNumber and description are defined as follows:

Name	Type	Description
errorNumber	xs:unsignedInt (W3C standard type) See Table 19 for currently specified error numbers.	The error number specifies whether the result is "OK" (value "0") or an error (value ">0").
description	xs:string (W3C standard type)	Descriptive information on the result.

Table 18: resultData function detailed description of elements

Values specified for element **errorNumber**:

Value	Description
0	No Error.
1	General Error.
2	Timeout.
3	Overload.
4	Destination unknown.
5	Destination unreachable.
6	Command not supported.
7	Command rejected.
8	Restricted function exchange combination not supported.
9	Binding is necessary for this command.

Table 19: Values specified for element errorNumber

Further error numbers will be added in future releases.

Example of an error indication:

```
<resultData>
    <errorNumber>2</errorNumber>
    <description>Timeout</description>
</resultData>
```


4 Device model definitions

The subsequent sections define details on the use of "Device Type", "Entity Type" and "Feature Type". An overview of all types can be found in Annex B.

4.1 Device Types

4.1.1 ChargingStation

4.1.1.1 Descriptive information

Represents a Charging Station, typically used in the context of electrical vehicle charging.

4.1.2 Dishwasher

4.1.2.1 Descriptive information

Represents a Dishwasher

4.1.3 Dryer

4.1.3.1 Descriptive information

Represents a Dryer.

4.1.4 ElectricitySupplySystem

4.1.4.1 Descriptive information

Represents a set of sources or storages of electric power of a premises (for example a local PV system, the electricity grid connection point or a battery storage).

4.1.5 EnergyManagementSystem

4.1.5.1 Descriptive information

Represents an Energy Management System (EMS).

4.1.6 EnvironmentSensor

4.1.6.1 Descriptive information

Represents a Sensor typically used for outdoor measurements.

4.1.7 Generic

4.1.7.1 Descriptive information

This is a generic device, which can either be used for client-only functionality or for development purposes where no specific type of device can be stated yet.

4.1.8 HeatGenerationSystem

4.1.8.1 Descriptive information

This Device Type is used to represent the subsystem of an HVAC system that generates heat. It can be used to represent single heat generators like gas heating boilers or heat pumps, but also be used to represent a heat generator cascade (like a cascade of several gas heating boilers or a combination of heat pump appliances and gas heating boilers). It can also be used to represent a solar thermal system.

The Device Type “HeatGenerationSystem” serves as a container for specific Entity Types that represent the single heat generators.

4.1.9 HeatSinkSystem

4.1.9.1 Descriptive information

Represents the end user relevant heat sinks of the HVAC system (heating circuits, domestic hot water (DHW) circuits, air conditioning system, …) and the end user relevant system functions (space heating, space cooling, DHW, air conditioning, …)

4.1.10 HeatStorageSystem

4.1.10.1 Descriptive information

Represents a system of heat storages consisting of one or more heat storage appliances or appliances that are attached to a heat storage (for example the Entity Type InstantDHWHeater (see section 4.2.31) (fresh water station attached to a heating buffer storage)). All heat storages of the HVAC system should be represented by this Device Type.

4.1.11 HVACController

4.1.11.1 Descriptive information

Represents the central controller appliance of an HVAC System.

A device of Device Type HVACController is used to deliver overall information about the HVAC system.

4.1.12 Inverter

4.1.12.1 Descriptive information

Represents a power Inverter.

4.1.13 SubMeter

4.1.13.1 Descriptive information

A SubMeter measures the demand or generation of a commodity (e.g. electricity).

4.1.14 Washer

4.1.14.1 Descriptive information

Represents a Washing Machine.

4.2 Entity Types

4.2.1 Battery

4.2.1.1 Descriptive information and further definitions

Represents a single battery unit.

4.2.2 BatterySystem

4.2.2.1 Descriptive information

A system consisting of one or more battery storages and one or more battery inverters.

4.2.3 CEM

4.2.3.1 Descriptive information and further definitions

Abbreviation for Customer Energy Manager and typically used to represent an Energy Manager within SPINE.

4.2.4 ChargingOutlet

4.2.4.1 Descriptive information and further definitions

Represents a Charging Outlet of a Charging Station.

4.2.5 Compressor

4.2.5.1 Descriptive information and further definitions

Represents the compressor of (e.g.) an (electrical) heat pump appliance.

4.2.6 DeviceInformation

4.2.6.1 Descriptive information and further definitions

This entity has a special role and contains general information about the complete device.

4.2.7 DHWCircuit

4.2.7.1 Descriptive information and further definitions

Represents all elements for domestic hot water supply like cold water line, domestic hot water line, circulation line, tapping points.
Should not include DHW storages, these should be represented as Entity Type “DHWStorage”

4.2.8 DHWStorage

4.2.8.1 Descriptive information and further definitions

Represents a heat storage appliance for storing domestic hot water.

4.2.9 Dishwasher

4.2.9.1 Descriptive information and further definitions

Represents a Dishwasher

4.2.10 Dryer

4.2.10.1 Descriptive information and further definitions

Represents a Dryer.

4.2.11 ElectricalImmersionHeater

4.2.11.1 Descriptive information and further definitions

Represents an electrical immersion heater (electrical resistance heater) for example as part of a heat pump appliance, a heat source unit, a DHW storage.

4.2.12 ElectricityGenerationSystem

4.2.12.1 Descriptive information and further definitions

A system that is capable of producing electric power through local generation (for example PV-systems, wind turbines or block heating and power plants).

4.2.13 ElectricityStorageSystem

4.2.13.1 Descriptive information and further definitions

A system that is capable of storing electric power and releasing the majority of it as electric power (for example a battery system or a capacitor).

4.2.14 EV

4.2.14.1 Descriptive information and further definitions

Represents an Electrical Vehicle.

4.2.15 EVSE

4.2.15.1 Descriptive information and further definitions

Abbreviation for Electrical Vehicle Supply Equipment, typically used to connect an Electical Vehicle to a Charging Station.

4.2.16 Fan

4.2.16.1 Descriptive information and further definitions

Represents an air fan, for example of an air water heat pump unit or a ventilation unit.

4.2.17 GasHeatingAppliance

4.2.17.1 Descriptive information and further definitions

Represents a heating appliance that generates heat (for example for heating or domestic hot water) by burning natural gas.

4.2.18 Generic

4.2.18.1 Descriptive information and further definitions

This is a generic entity, which can either be used for client-only functionality, or for development purposes, where no specific entity can be stated yet.

4.2.19 GridConnectionPointOfPremises

4.2.19.1 Descriptive information and further definitions

The electrical connection point between the utility electricity grid and the electric circuit of a premises.

4.2.20 HeatingBufferStorage

4.2.20.1 Descriptive information and further definitions

Represents a heat storage appliance for storing heating water.

4.2.21 HeatingCircuit

4.2.21.1 Descriptive information and further definitions

Represents a single heating circuit for space heating (or space cooling like active or passive cooling), including flow and return lines, optional mixing units (for mixed heating circuits) and the heat distribution:

- directly to one or more heated rooms (see Entity Type “HVACRoom”, section 4.2.30)
- to one or more heating zones (see Entity Type “HeatingZone”, section 4.2.23)

4.2.22 HeatingObject

4.2.22.1 Descriptive information and further definitions

Represents a Heating Object, e.g. a radiator.

4.2.23 HeatingZone

4.2.23.1 Descriptive information and further definitions

Represents a heating zone that provides heating energy (or cooling energy) to one or more heated rooms (see Entity Type HVACRoom).

4.2.24 HeatPumpAppliance

4.2.24.1 Descriptive information and further definitions

Represents either a complete heat pump appliance (for example a brine water heat pump, indoor; an air water heat pump, completely indoor or completely outdoor) or the indoor part of a refrigerant split heat pump combination.

4.2.25 HeatSinkCircuit

4.2.25.1 Descriptive information and further definitions

Represents a (hydraulic) circuit of a heat generator that delivers heat energy from the heat generator to a heat sink. For example Entity Type "HeatSinkCircuit" could represent a (hydraulic) circuit of a gas heating appliance that delivers heat energy to a heating circuit.

4.2.26 HeatSourceCircuit

4.2.26.1 Descriptive information and further definitions

Represents a (hydraulic) circuit of a heat generator (especially of heat pumps) that delivers heat energy from an ambient energy source to the heat generator. For example Entity Type "HeatSourceCircuit" could represent a brine circuit of a heat pump appliance that delivers heat energy from an ambient heat source to the heat pump appliance.

4.2.27 HeatSourceUnit

4.2.27.1 Descriptive information and further definitions

Represents an additional unit that provides the heat from the environmental heat source to the heat pump appliance. This could for example be:

- an outdoor ventilation unit that transfers the heat energy from the outside air to the heat source circuit of the (indoor) heat pump appliance (i.e. air water split heat pump)
- an outdoor ventilation unit that transfers the heat energy from the outside air to the refrigerant circuit of the (indoor) heat pump appliance (refrigerant split heat pump)

4.2.28 Household

4.2.28.1 Descriptive information and further definitions

This is a rather virtual entity representing a complete household that aggregates all relevant information of the household (e.g. the complete energy demand, etc.).

4.2.29 HVACController

4.2.29.1 Descriptive information and further definitions

Represents the central controller appliance of the HVAC System (as a SPINE Entity).

4.2.30 HVACRoom

4.2.30.1 Descriptive information and further definitions

Represents one physical room in a building/house that can be heated / cooled, ventilated or air conditioned.

2319

2320 **4.2.31 InstantDHWHeater**

2321 ***4.2.31.1 Descriptive information and further definitions***
2322 Represents an appliance for instant generation of domestic hot water (for example by using heat
2323 from a heating buffer storage).

2324

2325 **4.2.32 Inverter**

2326 ***4.2.32.1 Descriptive information and further definitions***
2327 Represents a power Inverter.

2328

2329 **4.2.33 OilHeatingAppliance**

2330 ***4.2.33.1 Descriptive information and further definitions***
2331 Represents a heating appliance that generates heat (for example for heating or domestic hot water)
2332 by burning oil.

2333

2334 **4.2.34 Pump**

2335 ***4.2.34.1 Descriptive information and further definitions***
2336 Represents a pump to generate a flow rate (for example in a heating circuit).

2337

2338 **4.2.35 PVSystem**

2339 ***4.2.35.1 Descriptive information and further definitions***
2340 A system consisting of one or more PV inverters with one or more PV modules. The PVSystem
2341 provides aggregated PV related system parts of the house or premises.

2342

2343 **4.2.36 RefrigerantCircuit**

2344 ***4.2.36.1 Descriptive information and further definitions***
2345 Represents the refrigerant circuit of a heat pump appliance.

2346

2347 **4.2.37 SmartEnergyAppliance**

2348 ***4.2.37.1 Descriptive information and further definitions***
2349 An appliance that allows a client to shift the load announced by the server can be modelled with this
2350 Entity Type. Non-shiftable loads (so-called "fixed forecasts") can be provided by this Entity, too.

4.2.38 SolarDHWStorage

4.2.38.1 Descriptive information and further definitions

Represents a heat storage appliance for storing domestic hot water. The heat energy can be provided especially by solar thermal energy (compare to Entity Type "DHWStorage").

4.2.39 SolarThermalCircuit

4.2.39.1 Descriptive information and further definitions

Represents a single circuit of a solar thermal system including a solar thermal collector field and flow and return pipes to/from the collector field and a solar circuit pump.

4.2.40 SubMeterElectricity

4.2.40.1 Descriptive information and further definitions

Represents an electrical Sub-Meter.

4.2.41 TemperatureSensor

4.2.41.1 Descriptive information and further definitions

Represents a sensor that can provide a temperature measurement value.

4.2.42 Washer

4.2.42.1 Descriptive information and further definitions

Represents a laundry washer that washes clothes (laundry).

4.3 Feature Types

4.3.1 ActuatorLevel

4.3.1.1 Dependencies to other Feature Types

None.

4.3.1.2 Introduction

Independent from the *ActuatorSwitch* class, the *ActuatorLevel* class enables a user or application to model *LEVEL* commands (*start*, *up*, *percentageAbsolute*, *relative*, ...). This can be used to dim a light, set the speed of an electric motor, etc. Its simple functions can easily be mapped to other technologies.

4.3.1.3 Rules

None.

4.3.1.4 actuatorLevelData

4.3.1.4.1 Description

This function is used for setting the desired operation mode of the node or getting the actual mode respectively.

4.3.1.4.2 Rules

None.

4.3.1.4.3 Element rules

Element	Rule	Description
function	SHALL always be set. Value rules of Table 21 SHALL be applied.	The element "function" contains the mode.
value	None.	The value is only needed for some functions. E.g. setting *function* to *percentageAbsolute* just makes sense together with a proper value.

Table 20: actuatorLevelData Element rules

Element "**function**" value rules:

Value	Rule	Description
start	None.	Actuator starts.
up	None.	Actuator moves up.
down	None.	Actuator moves down.
stop	None.	Actuator stops.
percentageAbsolute	None.	The percentage position or value of the actuator (e.g. 25 %).
percentageRelative	None.	Changes the current value by a percentage value (e.g. move plus 10 percent).
absolute	None.	The absolute position or value of the actuator (e.g. 0.91 m).
relative	None.	Changes the current value by some value (e.g. move plus 0.23 m).

Table 21: Element "function" value rules

4.3.1.5 actuatorLevelDescriptionData

4.3.1.5.1 Description

The rather static descriptive data of the *actuatorLevel* class can contain a default unit, among others. This is useful for some *absolute* commands.

2406 *4.3.1.5.2 Rules*
2407 None.

2408

2409 *4.3.1.5.3 Element rules*

Element	Rule	Description
label	Default rules of section 3.10.1.2.1 SHALL be applied.	A short label of the actuator.
description	Default rules of section 3.10.1.3.1 SHALL be applied.	Descriptive information on the actuator.
levelDefaultUnit	None.	If an actuator can specify a unit for its level-controlling, it can be denoted in this element.

2410 *Table 22: actuatorLevelDescriptionData Element rules*

2411

2412 ***4.3.1.6 Descriptive information and further definitions***
2413 None.

2414

2415 **4.3.2 ActuatorSwitch**

2416 ***4.3.2.1 Dependencies to other Feature Types***
2417 None.

2418

2419 ***4.3.2.2 Introduction***
2420 Basic on/off operations on a simple actuator can be modelled with the *ActuatorSwitch* class.
2421 Whether the function turns a device itself *on* or *off*, or whether it switches a specific functionality,
2422 depends on the implementation. E.g. one could model the super freeze program of a freezer using
2423 *ActuatorSwitch* class. An *on* command would then activate the super freeze program and an *off*
2424 command would deactivate it. This example shall just give an idea how *ActuatorSwitch* can be used
2425 for more purposes than only turning devices on and off.

2426 Please note a device's different features may be modelled with separate implementations of this
2427 class, each associated to a different node address. Continuing with the example above, a "defrost"
2428 function may be modelled on another node address with *ActuatorSwitch*.

2429

2430 ***4.3.2.3 Rules***
2431 None.

2432

4.3.2.4 actuatorSwitchData

4.3.2.4.1 Description

The *actuatorSwitchData* function models the functionality described in section 5.3.2.1.

4.3.2.4.2 Rules

None.

4.3.2.4.3 Element rules

Element	Rule	Description
function	None. Value rules of Table 24 SHALL be applied.	The only element in this function to model an *on*, *off* or *toggle* command.

Table 23: actuatorSwitchData Element rules

Element "**function**" value rules:

Value	Rule
on	If used in a write, the actuator SHALL be activated, if the write is accepted. If used in a notify or reply, the actuator SHALL be active.
off	If used in a write, the actuator SHALL be deactivated, if the write is accepted. If used in a notify or reply, the actuator SHALL NOT be active.
toggle	SHALL only be used in a write. If used in a write, an inactive actuator SHALL be activated and an active actuator SHALL be deactivated, if the write is accepted.

Table 24: Element "function" value rules

4.3.2.5 actuatorSwitchDescriptionData

4.3.2.5.1 Description

The (typically) non-changing information on the actuator is modelled with this function.

4.3.2.5.2 Rules

None.

4.3.2.5.3 Element rules

Element	Rule	Description
label	Default rules of section 3.10.1.2.1 SHALL be applied.	A short label of the actuator.
description	Default rules of section 3.10.1.3.1 SHALL be applied.	Descriptive information on the actuator.

Table 25: actuatorSwitchDescriptionData Element rules

4.3.2.6 Descriptive information and further definitions

None.

4.3.3 Alarm

4.3.3.1 Dependencies to other Feature Types

- Threshold

4.3.3.2 Introduction

There are several reasons for generating an alarm. This might be due to exceeding (in a positive or negative way) a threshold (modelled with the Threshold class, see section 5.3.26), or because of some other occurrence that provokes the generation of an alarm. The Alarm class provides the needed data model.

4.3.3.3 Rules

None.

4.3.3.4 alarmListData

4.3.3.4.1 Description

This function models data for an alarm. Depending on the type of alarm, different elements are used. It can be used to notify an alarm or as a reply to a read command.

4.3.3.4.2 Rules

None.

4.3.3.4.3 Element rules

Element	Rule	Description
alarmId	SHALL be set as PRIMARY IDENTIFIER.	Identifier of the alarm.
thresholdId	If the alarm is related to a threshold, the value SHALL be set as FOREIGN IDENTIFIER. Otherwise it SHALL be omitted.	If the alarm relates to a threshold, the corresponding ID can be stated here.
timestamp	SHOULD be set.	The time of creation of the alarm.
alarmType	SHOULD be set. Value rules of Table 27 SHALL be applied.	The type of alarm is given here.
measuredValue	None.	The value that was measured which exceeded the threshold is denoted in this element.

evaluationPeriod	None.	If a time period can be stated in which
scopeType	None.	Specifies a more detailed meaning of the alarm (e.g. exceeding of a threshold).
label	Default rules of section 3.10.1.2.1 SHALL be applied.	A short label of the alarm.
description	Default rules of section 3.10.1.3.1 SHALL be applied.	Descriptive information on the alarm.

Table 26: alarmListData Element rules

Element "**alarmType**" value rules:

Value	Rule
alarmCancelled	SHALL be used, if a previously generated alarm is cancelled.
underThreshold	SHALL be used, if the value is under a related threshold (*thresholdId* SHALL be set).
overThreshold	SHALL be used, if the value is over a related threshold (*thresholdId* SHALL be set).

Table 27: Element "alarmType" value rules

4.3.3.5 Descriptive information and further definitions

None.

4.3.4 Bill

4.3.4.1 Dependencies to other Feature Types

None.

4.3.4.2 Introduction

The Bill Feature can be used to issue bills, containing positions and their costs.

4.3.4.3 Rules

None.

4.3.4.4 billListData

4.3.4.4.1 Description

billData holds the bill data, including descriptive elements like currency, unit and billType.

A billData has a "total" element, including total values and corresponding costs in its sub-values. Additionally, positions of the total data can be provided, to describe the composition of the total data. To link total costs and values in a position the valueId and costId are used within a billData.

4.3.4.4.2 Rules

The identifier positionId MAY be omitted, if only one position is included in the bill. Otherwise it SHALL be set.

The identifier valueId MAY be omitted, if only one value is included in the bill. Otherwise it SHALL be set.

The identifier costId MAY be omitted, if only one cost is included in the bill. Otherwise it SHALL be set.

4.3.4.4.3 Element rules

Element	Rule	Description
billId	SHALL be set as PRIMARY IDENTIFIER, at least if more than one bill exists.	Allows the identification of a bill.
billType	Value rules of Table 29 SHALL be applied.	The type of the bill.
scopeType	None.	Specifies the scope of the bill.
total	None.	Total information of the bill (in contrast to the positions, see below).
total. timePeriod	None.	Allows to define a time, or time period for a bill.
total. value (list)	None.	Different values can be listed for a bill. E.g. energy values or just a number of items.
total. value. valueId	This Element SHALL be interpreted as SUB IDENTIFIER. If there is only one "value" entry within "total", the "valueId" MAY be omitted. Otherwise it SHALL be stated. If omitted, "valueId" SHALL equal “1”.	Allows the identification of a value within the bill.
total. value. unit	If the value has a unit, it SHOULD be stated here.	Allows definition of a unit for the value.
total. value. value	Sales from the perspective of the Feature Server SHALL be expressed with negative values and purchases with positive values. In "total" all position sales and purchases SHALL be accumulated.	Allows to express the total amount that was bought or sold of a certain commodity during the time or time period given in total.timePeriod.
total. cost (list)	None.	Different types of costs can be listed for a value. E.g. monetary costs like a price as well as environmental costs as CO2 emmission.
total. cost. costId	This Element SHALL be interpreted as SUB IDENTIFIER. If there is only one "cost" entry within "total", the "costId" MAY be omitted. Otherwise it SHALL be stated. If omitted, "costId" SHALL equal “1”.	Allows identification of costs within bill.

total. cost. costType	Value rules of Table 30 SHALL be applied.	Allows to describe the type of costs.
total. cost. valueId	If the cost is linked to a value, the corresponding valueId SHALL be set.	Allows to link cost to a value.
total. cost. unit	None.	Costs that are not monetary may need to describe a unit (e.g. m^3).
total. cost. currency	None.	Monetary costs may need to describe a currency.
total. cost. cost	None.	Total costs generated during the time or time period given in total.timePeriod.
total. label	Default rules of section 3.10.1.2.1 SHALL be applied.	Allows to express a user friendly label for the bill.
total. description	Default rules of section 3.10.1.3.1 SHALL be applied.	Allows to express a user friendly description for the bill.
position (list)	None.	For each total data different positions can be assigned that describe the composition of the total data.
position. positionId	This Element SHALL be interpreted as SUB IDENTIFIER. If there is only one "position" entry the "positionId" MAY be omitted. Otherwise it SHALL be stated. If omitted, "positionId" SHALL equal “1”.	Identifier of a position within a specific bill.
position. positionType	Value rules of Table 31 SHALL be applied.	Allows to describe the type of the a position E.g. different types of energy consumption like self produced or electricity grid energy.
position. timePeriod	None.	Allow to define a time, or time period for a position if the time or time period is different from the overall time period described in total.
position. value (list)	None.	Different values can be listed for each position.
position. value. valueId	This Element SHALL be interpreted as SUB IDENTIFIER. If there is only one "value" entry for the "position", the "valueId" MAY be omitted. Otherwise it SHALL be stated. If omitted, "valueId" SHALL equal “1”.	Identifier of a value within a specific bill (see also "total. value. valueId").
position. value. value	Allows to describe a value for each position. This can be helpful if some positions have negative while other positions have positive values, as this would can not be express with valuePercentage.	Allows to describe a value for each position. This can be helpful if some positions have negative while other positions have positive values, as this would can not be express with valuePercentage.
position. value. valuePercentage	Percentage of the total value (see "total").	Percentage of the total value (see "total").
position. cost (list)	Different costs can be listed for each position	Different costs can be listed for each position

position. cost. costId	This Element SHALL be interpreted as SUB IDENTIFIER. If there is only one "cost" entry for the "position", the "costId" MAY be omitted. Otherwise it SHALL be stated. If omitted, "costId" SHALL equal “1”.	Identifier of a cost within a specific bill (see also "total. cost. costId").
position. cost. cost	None.	The actual costs for this position. This can be helpful if some positions have negative while other positions have positive costs, as this can not be expressed with costPercentage.
position. cost. costPercentage	None.	Percentage of the total costs (see "total").
position. label	Default rules of section 3.10.1.2.1 SHALL be applied.	Allows to provide a user friendly label for each bill position.
position. description	Default rules of section 3.10.1.3.1 SHALL be applied.	Allows to provide user friendly information for each bill position.

Table 28: billListData Element rules

Element "**billType**" value rules:

Value	Rule	Description
chargingSummary	This summary SHOULD NOT be used for actual billing as it may also contain approximated values.	This bill type allows to summarize a recharging cycle, e.g. containing total energy charged, total costs as well as billing positions for different types of energy.

Table 29: Element "billType" value rules

Element "**costType**" value rules:

Value	Rule	Description
absolutePrice	None.	Used for absolute prices.
relativePrice	None.	Used for relative prices, e.g. a price that shows how much cheaper the costs are related to the average tariff in place.
co2Emission	None.	In this case the cost value is used to express how much CO2 emission was emitted.
renewableEnergy	None.	In this case the cost value is used to express how much renewable energy was used.
radioactiveWaste	None.	In this case the cost value is used to express how much radioactive waste was generated.

Table 30: Element "costType" value rules

Element "**positionType**" value rules:

Value	Rule	Description
gridElectricEnergy	None.	This position type specifically describes the electricity grid energy portion of the total data.
selfProducedElectricEnergy	None.	This position type specifically describes the self produced energy portion of the total data.

Table 31: Element "positionType" value rules

4.3.4.5 billConstraintsListData

4.3.4.5.1 Description

The billConstraintsListData function allows a server to define certain constraints for each linked billListData list entry. This is especially important if a particular billListData list entry is writeable and allows the server to communicate the corresponding constraints before a client performs a write command. The client can match the server constraints with own constraints and perform a corresponding write command.

4.3.4.5.2 Rules

None.

4.3.4.5.3 Element rules

Element	Rule	Description
billId	SHALL be set as PRIMARY IDENTIFIER.	Allows the identification of a bill. Allows linking of the different functions to the same bill. Allows linking of the bill within other features that are placed in the same entity.
positionCountMin	If a related billData exists, it SHALL NOT contain less positions.	Minimum amount of possible positions within the specified bill.
positionCountMax	If a related billData exists, it SHALL NOT contain more positions.	Maximum amount of possible positions within the specified bill.

Table 32: billConstraintsListData Element rules

4.3.4.6 billDescriptionListData

4.3.4.6.1 Description

The billDescriptionListData function is especially important if a linked billListData list entry is writeable and allows a server to define if a bill is writeable and which billTypes are supported for the corresponding billId, so that a client can perform corresponding writes. Additionally, the server can request updates from a client with the updateRequired flag.

4.3.4.6.2 Rules

None.

4.3.4.6.3 Element rules

Element	Rule	Description
billId	SHALL be set as PRIMARY IDENTIFIER.	Allows the identification of a bill. Allows linking of the different functions to the same bill. Allows

		linking of the bill within other features that are placed in the same entity.
billWriteable	If set to "true", a client MAY write the related billData entry. If there is no billData entry with a corresponding billId, the client MAY still write the complete billData entry and the server SHALL accept it, if the write command matches the rules given in the related billDescriptionData and billConstraintsData entries for the corresponding billId.	Indicates whether the related billData entry is writeable.
updateRequired	With updateRequired the server can request an update of writeable or changeable data related to the same PRIMARY IDENTIFIER from a client. The server SHALL ensure that only one responsible client is permitted to update the related data. To request an update the server SHALL set updateRequired to “true”. Note: In this case, the server expects the responsible client to update the writeable or changeable data related to the same PRIMARY IDENTIFIER. However, also if updateRequired is set to “false” a server SHOULD in general allow updates of the data from the responsible client. The server SHALL set the updateRequired back to “false”, as soon as “billDescriptionData” was updated successfully (if writeable or changeable) OR the update of the other writeable or changeable data related to the same PRIMARY IDENTIFIER was successful. The server MAY choose to withdraw the update request at any time by setting updateRequired back to “false”. Note: The client does not need to stop an ongoing update process (e.g. if multiple functions are written), when updateRequired is set back to “false”.	Indicates whether the server requests an update for the bill.
supportedBillType (list)	None.	List of all supported types for this billId.

Table 33: billDescriptionListData Element rules

4.3.4.7 Descriptive information and further definitions

None.

4.3.5 DataTunneling

4.3.5.1 Dependencies to other Feature Types

None.

4.3.5.2 Introduction

DataTunneling provides a mechanism to submit data of proprietary protocols using the SPINE data models (esp. addressing other nodes). However, the nature of proprietary definitions imposes restrictions:

Unless stated otherwise, the term “payload” refers to the sub-element of "dataTunnelingCall" within this section (in contrast to the definition of “payload” as sub-element of “datagram” as defined in document [ProtocolSpecification]).

Please note that the use of the above mentioned elements will always be proprietary. The SPINE specification will never define any specific protocol for dataTunneling.

To put it in other words, the SPINE data tunnelling model is not intended for any kind of flexible extension of the SPINE classes. This is mainly because a so-called technology specific mapping could hardly be defined in a standardized way. Another reason is that it shall always be transparent that data tunnelling contains proprietary data. This should not be considered a limitation of SPINE as the EEBus Initiative e.V. committees may define specific classes with functions (like messagingListData, timeInformationData, e.g.) including mappings for specific protocols, if need be.

4.3.5.3 Rules

None.

4.3.5.4 dataTunnelingCall

4.3.5.4.1 Description

In general, several conditions must be fulfilled in order to make use of SPINE data tunnelling. This is explained with an example:

A manufacturer "X" of a device "A" created a proprietary extension of a communications protocol (e.g. KNX, ZigBee, LON, ... – we call it "CP" here) in order to configure device "A" with special parameters. We assume there is no specific SPINE function defined so far for these parameters. There is also no mapping defined for it.

Another manufacturer "Y" creates a device "B" that runs "SPINE applications" (i.e. applications communicating with an internal interface using XMLs of the SPINE data models). The device "B" can as well communicate via "CP".

Now, manufacturer "Y" wants to make use of the proprietary configuration of device "A". Therefore, manufacturer "Y" needs to implement the proprietary extension of manufacturer "X"'s communications protocol as well. I.e. the "CP" implementation of device "B" must be extended properly. Furthermore, manufacturer "Y" needs to define how the proprietary protocol shall be represented as SPINE data tunnelling towards the "SPINE applications" running on device "B" (i.e. define the proprietary mapping) and of course needs to implement this mapping. Finally, the applications need to implement the recognition and use of the corresponding SPINE data tunnelling XMLs.

The scenario above may look "unattractive" or even raise concerns. However, it shall be mentioned that proprietary extensions of standardized protocols are not unusual. This practice shall not be judged here.

In the aforementioned scenario it is manufacturer "B" who implements a number of extensions in order to comply with the extensions of device "A". But it could also be possible that manufacturer "A" already provided some definitions and implementation for other manufacturers. Then, manufacturer "B" could integrate this solution and potentially implements less by its own.

Some may find a comparison with Virtual Private Networks helpful (though VPNs have a different scope): Two computers "A", "B" shall each run a compatible (proprietary) program (i.e. mapping/implementation) that know how to communicate with each other (i.e. set up the communication and provide a specific authentication and encryption). There are manufacturers of VPN gateways that provide client software for different operating systems. This way a proprietary implementation is made available for others.

The combination { purposeId, channelId, sequenceId } should be considered unique for a communication from "A" to "B". Within a system (e.g. house) several communications may be used in parallel. The combinations used in these communications are in general independent of each other.

Please note that an instance of a dataTunnelingCall implementation permits bidirectional communication in general. I.e. if a dataTunnelingCall functionality of “B” submits a message to a dataTunnelingCall functionality of “A”, it is also permissive that the dataTunnelingCall functionality of “A” submits dataTunnelingCall messages to the dataTunnelingCall functionality of “B”. However, similar to a VPN, it may well be that the establishment for such message exchanges is “unidirectional” to some extent. I.e. it may be that the dataTunnelingCall functionality of “A” determines which kind of messages (or permissive values for “purposeId”) are accepted on this functionality and that the functionality first of all acts as some kind of “listening port”. This means the functionality of “A” may wait for an initial incoming message before it also sends dataTunnelingCall messages back.

Please note that the example above does not prevent device “B” from implementing another dataTunnelingCall functionality as “listening port” with own permitted values for “purposeId” as well. In that case device “A” could send a proper initial message to this “listening functionality” of “B”.

SPINE data tunnelling is intended to be used for call-operations only. No read-reply or write procedures are supported.

4.3.5.4.2 Rules

None.

4.3.5.4.3 Element rules

Element	Rule	Description
header	SHALL be set.	A data tunneling specific header.
header. purposeId	SHALL be set as PRIMARY IDENTIFIER. The string-length SHOULD NOT be longer than 128 characters. If it is longer, the sender SHALL consider the possibility that the receiver will shorten the string to 128 characters.	A string that identifies the purpose / kind of tunnelled data / protocol.
header. channelId	SHALL be set as SUB IDENTIFIER.	Between two nodes there might be several channels used in parallel using the same purposeId. channelId helps to separate these channels. It can also be used to differentiate subsequent (rather than parallel) sessions.
header. sequenceId	SHALL be set as SUB IDENTIFIER.	This helps to give the payload for each combination of { purposeId, channelId } a chronological order. Note: This “sequenceId” is data tunneling specific and should not be confused with “sequenceId” elements of other classes!
payload	SHALL be set.	The actual proprietary data.

Table 34: dataTunnelingCall Element rules

4.3.5.5 Descriptive information and further definitions

The DataTunneling Feature Type permits exchange of arbitrary data.

Only "call" operations are permitted with this Feature Type.

Unlike “conventional” features (i.e. features with functions where read/reply or write operations are supported) the DataTunneling feature does not apply the “data owner” concept in a usual way at first glance. However, the role concept still applies as follows:

An instance of Feature Type DataTunneling and role “server” defines which kind of data is exchanged. Specifically, the permitted values for the element “header.purposeId” (if used) are determined by the server feature only.

4.3.6 DeviceClassification

4.3.6.1 Dependencies to other Feature Types

None.

4.3.6.2 Introduction

Each device can have associated information useful for its identification and which kind of power it requires. This information (mostly plaintext) is bundled in the *DeviceClassification* class. It is intended to describe the physical device, but may be used for logical devices (i.e. *entities*), too.

4.3.6.3 Rules

None.

4.3.6.4 deviceClassificationManufacturerData

4.3.6.4.1 Description

Information which is usually NOT writable (hard coded in the device or preconfigured information, e.g.), is modelled with *deviceClassificationManufacturerData*.

4.3.6.4.2 Rules

No specific element is mandatory, but at least one SHALL be set.

For visualisation purposes, the content of the element *vendorName* SHOULD be used as name of the vendor only if the element *brandName* is not set. Otherwise brandName SHOULD be used.

4.3.6.4.3 Element rules

Element	Rule	Description
deviceName	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.	The name of the (physical or logical) device as defined by the manufacturer.
deviceCode	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.	A device code for the (physical or logical) device as defined by the manufacturer.

serialNumber	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.	The serial number of the (physical or logical) device as defined by the manufacturer. Usually the same as printed on the case.
softwareRevision	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.	The software revision of the (physical or logical) device as defined by the manufacturer.
hardwareRevision	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.	The hardware revision of the (physical or logical) device as defined by the manufacturer.
vendorName	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.	The name of the vendor of the (physical or logical) device as defined by the manufacturer.
vendorCode	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.	A code for the vendor of the (physical or logical) device as defined by the manufacturer.

brandName	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.	The name of the brand. Useful where the name of the brand and the vendor differs.
powerSource	Value rules of Table 36 SHALL be applied.	This element describes for which kind of primary power source the device is designed for. It is only used for informative purposes. It is NOT used to describe how or to which kind of power source the device is finally connected (this means it is not considered here whether the device's single mains is fed by a house's mains supply or a backup sytem, e.g.).
manufacturerNodeIdentification	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.	A node identification for the (physical or logical) device as defined by the manufacturer. This could be used for the identification of a (physical or logical) device, even if it was removed from the network and rejoined later with changed node address.
manufacturerLabel	Default rules of section 3.10.1.2.1 SHALL be applied.	A short label of the (physical or logical) device as defined by the manufacturer.
manufacturerDescription	Default rules of section 3.10.1.3.1 SHALL be applied.	A description for the (physical or logical) device as defined by the manufacturer.

Table 35: deviceClassificationManufacturerData Element rules

Element "**powerSource**" value rules:

Value	**Rule**
unknown	None.
mainsSinglePhase	None.
mains3Phase	None.
battery	None.
dc	None.

Table 36: Element "powerSource" value rules

4.3.6.5 deviceClassificationUserData

4.3.6.5.1 Description

Information which is usually writable (changeable by a user / application) is modelled with *deviceClassificationUserData*.

4.3.6.5.2 Rules

No specific element is mandatory, but at least one SHALL be set.

4.3.6.5.3 Element rules

Element	Rule	Description
userNodeIdentification	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.	Similar to *manufacturerNodeIdentification* but defined by a user or application.
userLabel	Default rules of section 3.10.1.2.1 SHALL be applied.	Similar to *manufacturerLabel* but defined by a user or application.
userDescription	Default rules of section 3.10.1.3.1 SHALL be applied.	Similar to *manufacturerDescription* but defined by a user or application.

Table 37: deviceClassificationUserData Element rules

4.3.6.6 Descriptive information and further definitions

None.

4.3.7 DeviceConfiguration

4.3.7.1 Dependencies to other Feature Types

None.

4.3.7.2 Introduction

Some information cannot be represented by the "normal" SPINE classes, but are important for the interoperable functionality of a device, hence should not be modelled with the DataTunneling class (see section 5.3.6) or other proprietary ways. For this purpose, the DeviceConfiguration class should be used. It provides a key-value based modelling of information. Some data, important for interoperability, is already defined by the EEBus Initiative e.V. and listed in Table 187.

The information is split into a rather static part (see section 4.3.7.5) and the actual value (see section 4.3.7.4).

4.3.7.3 Rules

None.

4.3.7.4 deviceConfigurationKeyValueListData

4.3.7.4.1 Description

The actual values belonging to a specific key are modelled within this function.

4.3.7.4.2 Rules

None.

4.3.7.4.3 Element rules

Element	Rule	Description
keyId	SHALL be set as PRIMARY IDENTIFIER.	Enables the identification of different keys on one SPINE feature.
value	Exactly one of the child elements SHALL be set. This SHALL match with the content of element *valueType* within the key value description part (see section 5.3.8.3.2). Value rules of Table 39 SHALL be applied.	The actual value belonging to the *keyId*.
isValueChangeable	If set to "true" the server SHALL accept changes of the element "value" by appropriate clients (e.g. the bound client). Otherwise, the server SHALL decline change requests.	States whether the value belonging to the *keyId* is changeable by a client or not.

Table 38: deviceConfigurationKeyValueListData Element rules

Element "**value**" sub-element rules:

Element	Rule	Description
boolean	If the Element "boolean" is set, the "valueType" Element within the related deviceConfigurationDescriptionData entry SHALL be set to "boolean".	Used, if the *value* is of type *boolean*.
date	If the Element "date" is set, the "valueType" Element within the related deviceConfigurationDescriptionData entry SHALL be set to "date".	Used, if the *value* is of type *date*.
dateTime	If the Element "dateTime" is set, the "valueType" Element within the related deviceConfigurationDescriptionData entry SHALL be set to "dateTime".	Used, if the *value* is of type *dateTime*.
duration	If the Element "duration" is set, the "valueType" Element within the related deviceConfigurationDescriptionData entry SHALL be set to "duration".	Used, if the *value* is of type *duration*.
string	If the Element "string" is set, the "valueType" Element within the related deviceConfigurationDescriptionData entry SHALL be set to "string".	Used, if the *value* is of type *string*.

	The string-length of the Element SHOULD NOT be longer than 512 characters. If it is longer, the sender SHALL consider the possibility that the receiver will shorten the string to 512 characters.	
time	If the Element "time" is set, the "valueType" Element within the related deviceConfigurationDescriptionData entry SHALL be set to "time".	Used, if the *value* is of type *time*.
scaledNumber	If the Element "scaledNumber" is set, the "valueType" Element within the related deviceConfigurationDescriptionData entry SHALL be set to "scaledNumber".	Used, if the *value* is of type *scaledNumber*.

Table 39: Element "value" sub-element rules

4.3.7.5 deviceConfigurationKeyValueDescriptionListData

4.3.7.5.1 Description

The rather static part of the device configuration keys is modelled with this function, including an identifier for the key, a name that defines the meaning of the key and further additional elements.

4.3.7.5.2 Rules

None.

4.3.7.5.3 Element rules

Element	Rule	Description
keyId	SHALL be set as PRIMARY IDENTIFIER.	The key identifier.
keyName	Value rules of Table 31 SHALL be applied.	A certain key name. If one of the keys defined by the EEBus Initiative e.V. is used, the meaning is defined in Table 187. Vendors may define own key names (but then have to mark them as vendor specific, see [ProtocolSpecification], section "Rules for vendor specific extensions").
valueType	Most *keyName* definitions specify which *valueType* shall be used. Value rules of Table 31 SHALL be applied.	The type of the "value" Element within the corresponding "deviceConfigurationKeyValueData" entry is denoted here.
unit	If set, the unit SHALL be applied to the value of the key.	The unit in which the value of the key is given. Not all keys need the element "unit".
label	Default rules of section 3.10.1.2.1 SHALL be applied.	A label for this key.
description	Default rules of section 3.10.1.3.1 SHALL be applied.	A description for this key.

Table 40: deviceConfigurationKeyValueDescriptionListData Element rules

Element "**keyName**" value rules:

Value	Rule	Description
pvCurtailmentLimitFactor	This value SHALL be in range of 0 to 100. If used, the content of the element "valueType" SHALL be "scaledNumber" and the element "unit" SHALL be set to "pct".	The percentage of the peak power of the pv system that is permitted to be fed into the electricity grid through the electricity grid connection point.
peakPowerOfPvSystem	This value SHALL be positive or 0. If used, the content of the element "valueType" SHALL be "scaledNumber" and the element "unit" SHALL be set to "W".	The nominal peak power of the production of the installed PV System.
asymmetricChargingSupported	If used, the content of the element "valueType" SHALL be "boolean" and the element "unit" SHALL be omitted.	This configuration parameter is used to indicate if it is possible to charge with different current per phase.
communicationsStandard	If used, the content of the element "valueType" SHALL be "string" and the element "unit" SHALL be omitted. As content of the element "value.string" within the linked deviceConfigurationKeyValueData entry, only one of the following strings SHALL be used: - iso15118-2ed1 - iso15118-2ed2 - iec61851	Defines the used communications standard for the entity where the DeviceConfiguration feature is located.

Table 41: Element "keyName" value rules

Element "**valueType**" value rules:

Value	Rule	Description
boolean	SHALL be used, if the corresponding key value is of type xs:boolean.	The corresponding key value is of type xs:boolean.
date	SHALL be used, if the corresponding key value is of type xs:date.	The corresponding key value is of type xs:date.
dateTime	SHALL be used, if the corresponding key value is of type xs:dateTime.	The corresponding key value is of type xs:dateTime.
duration	SHALL be used, if the corresponding key value is of type xs:duration.	The corresponding key value is of type xs:duration.
string	SHALL be used, if the corresponding key value is of type xs:string.	The corresponding key value is of type xs:string.
time	SHALL be used, if the corresponding key value is of type xs:time.	The corresponding key value is of type xs:time.
scaledNumber	SHALL be used, if the corresponding key value is of type *"ScaledNumberType" (see section 3.10.1.8*).	The corresponding key value is of type *"ScaledNumberType" (see section 3.10.1.8*).

Table 42: Element "valueType" value rules

4.3.7.6 deviceConfigurationKeyValueConstraintsListData

4.3.7.6.1 Description

In cases where a configuration value has some range, it could make sense to specify the upper and lower bounds that are possible for this value as well as a step size. This can be achieved with the deviceConfigurationKeyValueConstraintsListData function.

4.3.7.6.2 Rules

None.

4.3.7.6.3 Element rules

Element	Rule	Description
keyId	SHALL be set as PRIMARY IDENTIFIER.	Enables the identification of different keys on one SPINE feature.
valueRangeMin	If set, the child element SHALL match with the content of the Element *valueType* within the related deviceConfigurationDescriptionData entry. For rules on the sub-elements see Table 39.	The lower bound that is possible for the related value.
valueRangeMax	If set, the child element SHALL match with the content of the Element *valueType* within the related deviceConfigurationDescriptionData entry. For rules on the sub-elements see Table 39.	The upper bound that is possible for the related value.
valueStepSize	If set, the child element SHALL match with the content of the Element *valueType* within the related deviceConfigurationDescriptionData entry. For rules on the sub-elements see Table 39.	The step size that is possible for the related value.

Table 43: deviceConfigurationKeyValueListData Element rules

4.3.8 DeviceDiagnosis

4.3.8.1 Dependencies to other Feature Types

None.

4.3.8.2 Introduction

This class contains information about the state of a device, heartbeat counting and service data.

4.3.8.3 Rules

None.

4.3.8.4 deviceDiagnosisStateData

4.3.8.4.1 Description

The *deviceDiagnosisStateData* function provides information about dynamically changing state information.

4.3.8.4.2 Rules

No specific element is mandatory, but at least one SHALL be set.

4.3.8.4.3 Element rules

Element	Rule	Description
timestamp	None.	The time of creation of the state.
operatingState	Value rules of Table 45 SHALL be applied.	The current operating state of the device.
vendorStateCode	The string-length SHOULD NOT be longer than 128 characters. If it is longer, the sender SHALL consider the possibility that the receiver will shorten the string to 128 characters.	A vendor specific state. May contain any state the vendor uses for its devices.
lastErrorCode	The string-length SHOULD NOT be longer than 128 characters. If it is longer, the sender SHALL consider the possibility that the receiver will shorten the string to 128 characters. Even if the device's "operationState" has a value of *normalOperation* again, the error code SHOULD remain in the element lastErrorCode.	The last error code that occurred.
upTime	If used, the following rules apply: - If located on entity "DeviceInformation" (see section 4.2.6), the up time of the complete device SHALL be stated here. - If located on some other entity, only the up time of the specific functionality SHALL be stated. In case of nested entities, the up	The duration the functionality is running since the last start.

	time of all "child"-entities SHALL be considered, too.	
totalUpTime	If used the following rules apply: - If located on entity "DeviceInformation" (see section 4.2.6), the total up time of the complete device SHALL stated here. - If located on some other entity, only the total up time of the specific functionality SHALL be stated. In case of nested entities, the total up time of all "child"-entities SHALL be considered, too.	The total duration the functionality is running since installation (see *installationTime* element in *deviceDiagnosisServiceData* function).
powerSupplyCondition	Value rules of Table 46 SHALL be applied.	The condition of the power supply can be stated here.

2757 *Table 44: deviceDiagnosisStateData Element rules*

2758 Element "**operatingState**" value rules:

Value	Rule	Description
normalOperation	None.	The device does its normal operation without any errors or limitations.
standby	None.	The device is in standby mode and does nothing but waits for user interaction and listens on its communications protocol for external commands.
failure	None.	A failure occurred and the device cannot run as desired. Its normal functionality is likely not given.
serviceNeeded	None.	The device needs some kind of service. Possibly the normal functionality is not given.
overrideDetected	None.	The device detected an override. Normal operation may be given or the device is in some safe mode.
inAlarm	None.	An alarm (or emergency) occurred and user interaction is needed. Normal operation may be given or the device is in some safe mode.
notReachable	None.	The device is not reachable via its non-SPINE communications protocol. This operating state can only be set by some technology gateway device that handles the normal communication to the device.
finished	None.	The device has finished its temporary operation. As long as it remains in state *finished*, the normal operation is not given.

2759 *Table 45: Element "operatingState" value rules*

2760 Element "**powerSupplyCondition**" value rules:

Value	Rule	Description
good	None.	The power supply is in a good condition.
low	None.	The power supply has low capacity. It would make sense to check it (e.g. change battery).
critical	None.	The power supply is in critical state. If it is supplied by a battery, a change is advised.
unknown	None.	The power supply condition is unknown. A notification to a user could make sense.

error	None.	The power supply is in error condition.

Table 46: Element "powerSupplyCondition" value rules

4.3.8.5 deviceDiagnosisHeartbeatData

4.3.8.5.1 Description

The *deviceDiagnosisHeartbeatData* function enables sending a specific signal, which indicates a device is still running.

4.3.8.5.2 Rules

No specific element is mandatory, but at least one SHALL be set.

4.3.8.5.3 Element rules

Name	Rule	Description
timestamp	None.	The time of creation of the data.
heartbeatCounter	The value of the heartbeatCounter element SHALL be increased after every *heartbeatTimeout* (NOT with every sending of this function). The *deviceDiagnosisHeartbeatData* function can not only be sent initially by the device itself, but can be requested by another device, too. In this case, the element *heartbeatCounter* SHALL NOT be incremented and the heartbeatTimeout has (as always) its fixed value (i.e. not the remaining time to the next (automatic) notification by the device).	An incrementing counter of the heartbeat.
heartbeatTimeout	A notification of the deviceDiagnosisHeartbeatData function SHALL be sent to all subscribed clients after each heartbeatTimeout period.	The heartbeatTimeout element contains the period, in which the deviceDiagnosisHeartbeatData function is sent by the device (NOT the remaining time till the next sending).

Table 47: deviceDiagnosisHeartbeatData Element rules

4.3.8.6 deviceDiagnosisServiceData

4.3.8.6.1 Description

The *deviceDiagnosisServiceData* function informs about service information related to a device's installation or maintenance.

4.3.8.6.2 Rules

No specific element is mandatory, but at least one SHALL be set.

4.3.8.6.3 Element rules

Name	Rule	Description
timestamp	None.	The time of creation of the data.
installationTime	None.	The point in time the device was installed (commissioned).
bootCounter	None.	The count of boot processes the device completed since *installationTime*.
nextService	None.	The point in time, the next service should be done, is stated by this element. Alternatively, the decrementing duration till the next service can be indicated.

Table 48: deviceDiagnosisServiceData Element rules

4.3.8.7 Descriptive information and further definitions

None.

4.3.9 DirectControl

4.3.9.1 Dependencies to other Feature Types

- OperatingConstraints
- PowerSequences
- SmartEnergyManagementPs

4.3.9.2 Introduction

Some kind of electricity consuming or producing devices require (in contrast to the PowerSequences Class, see section 5.3.19) a rather instantaneous energy consumption or generation control. This can be modelled with two primary information sets:

1. Notification of instantaneous consumption/generation as well as instantaneous control. This is covered by DirectControl and discussed in this section.
2. Information on the power/energy constraints and implications. This is covered by OperatingConstraints and discussed in section 4.3.20.

Information on and control of energy/power can be modelled using the directControlActivityListData Function. The directControlDescriptionData Function is used to model some general rather constant data.

4.3.9.3 Rules

None.

4.3.9.4 directControlActivityListData

4.3.9.4.1 Description

The function *directControlActivityListData* combines some information on the activity of a given time. Apart from information on state, power and energy it can contain information whether power, energy, etc. can be modified by a client with a proper (partial) write command.

4.3.9.4.2 Rules

The current values SHALL be represented by a list entry with timestamp = "zero seconds" (e.g. "PT0S" or "PT0H"). The server SHALL only accept write commands on data instances with timestamp = "zero seconds" (e.g. "PT0S" or "PT0H") to adjust current values of the Function.

4.3.9.4.3 Element rules

Name	Rules	Description
timestamp	SHALL be set as PRIMARY IDENTIFIER. The most actual directControlActivityData SHALL have a duration (i.e. relative time to "now") of zero seconds.	The timestamp of this directControlActivityData.
activityState	Value rules of Table 50 SHALL be applied.	The state of the activity.
isActivityStateChangeable	If this element is set to "true" the device's *activityState* MAY be modified by a client.	Indicates whether the "activityState" can be modified by a client or not.
energyMode	For further value rules see section 3.10.1.16.1.	Energy mode of the device.
isEnergyModeChangeable	If this element is set to *true* the device's "energyMode" MAY be modified by a client.	Indicates whether the "energyMode" can be modified by a client or not.
power	Target value that SHOULD be followed as good as possible.	The overall electrical power value belonging to this activity information. Please note: Phase-specific information is not given here.
isPowerChangeable	If this element is set to *true* the device's "power" MAY be modified by a client.	Indicates whether the "power" can be modified by a client or not.
energy	Target value that SHOULD be followed as good as possible.	The electrical energy value belonging to this activity information.
isEnergyChangeable	If this element is set to *true* the device's "energy" MAY be modified by a client.	Indicates whether the "energy" can be modified by a client or not.
sequenceId	If the activity is related to a power sequence, the according sequenceId SHOULD be stated here as FOREIGN IDENTIFIER.	The related power sequence identifier (if available).

Table 49: directControlActivityListData Element rules

Element "**activityState**" value rules:

Value	Rule	Description
running	If *isActivityStateChangeable* is set to *true*, it MAY be paused or deactivated by a client (if *sequenceId* is set, further information from the related sequence SHALL be considered).	The activity is running.
paused	The activity is paused. If *isActivityStateChangeable* is set to *true*, it MAY be activated or deactivated by a client (if *sequenceId* is set, further information from the related sequence SHALL be considered).	The activity is paused.
inactive	The activity is inactive. If *isActivityStateChangeable* is set to *true*, it MAY be activated by a client (if *sequenceId* is set, further information from the related sequence SHALL be considered).	The activity is inactive.

Table 50: Element "activityState" value rules

4.3.9.5 directControlDescriptionData

4.3.9.5.1 Description

This function contains the rather static information available in this Feature Type.

4.3.9.5.2 Rules

None.

4.3.9.5.3 Element rules

Element	Rule	Description
positiveEnergyDirection	SHOULD be set. For further value rules see section 3.10.1.14.1.	The *positiveEnergyDirection* states whether energy consumption or production will be counted as positive value.
powerUnit	SHOULD be set, if a power is stated in the function *directControlActivityListData*.	This is the unit for all power values related to consumption or production.
energyUnit	SHOULD be set, if an energy is stated in the function *directControlActivityListData*.	This is the unit for all energy values related to consumption or production.

Table 51: directControlActivityListData Element rules

4.3.9.6 Descriptive information and further definitions

None.

4.3.10 ElectricalConnection

4.3.10.1 Dependencies to other Feature Types

- Measurement

4.3.10.2 Introduction

Electrical connections have some more complex parameters (e.g. which AC phase is measured, real/reactive/apparent measurement, amplitude/rms, number of measured harmonic, etc.), which are modelled with this class. Some electrical connection specific state information (e.g. AC or DC, number of connected phases (only AC), positive energy direction, label, etc.) is modelled, too.

Measuring electrical values is still done with the Measurement class (see section 5.3.15). But the above-mentioned additional parameters are only described within the ElectricalConnection class. The *measurementId* from the Measurement class is stated in the *electricalConnectionParameterDescriptionListData* function, to have a proper reference.

4.3.10.3 Rules

None.

4.3.10.4 electricalConnectionParameterDescriptionListData

4.3.10.4.1 Description

The aforementioned parameters for electrical connection measurement are described in this function. The value itself is modelled in the Measurement class (section 5.3.15). Only a reference is given in this function.

4.3.10.4.2 Rules

None.

4.3.10.4.3 Element rules

Element	Rule	Description
electricalConnectionId	SHALL be set as PRIMARY IDENTIFIER.	Reference to the electrical connection, described in the *electricalConnectionDescriptionListData* function (see section 5.3.11.5).
parameterId	SHALL be set as SUB IDENTIFIER.	One electrical connection can have multiple parameter combinations. The *parameterId* helps to distinguish them.
measurementId	The FOREIGN IDENTIFIER MAY be set. If set, the related electrical connection data SHALL be linked to a measurand or data of another Feature that uses the same measurementId.	This is a foreign identifier.
voltageType	If *powerSupplyType* in *electricalConnectionDescriptionData* has the value "dc", this element MAY be ommitted. If stated it SHALL be set	Specifies which kind of electricity is measured ("ac" or "dc").

	to "dc". If *powerSupplyType* is of value "ac", the *voltageType* can either have the value "ac" (*voltageType* then MAY be ommitted), or "dc". In the latter case, the DC-offset of the ac connection is measured and this element SHALL be set accordingly. Value rules of Table 53 SHALL be applied	
acMeasuredPhases	In case of *powerSupplyType* = "dc", this element SHALL be ommitted. Value rules of Table 54 SHALL be applied.	In case of *powerSupplyType* = "ac", this element states the phase, which is measured. Combinations of phases (e.g. "ab") are possible.
acMeasuredInReferenceTo	In case of *powerSupplyType* = "dc", this element SHALL be ommitted. Value rules of Table 55 SHALL be applied.	In case of *powerSupplyType* = "ac", this element states the phase, which *acMeasuredPhases* is measured against (e.g. "neutral").
acMeasurementType	In case of *powerSupplyType* = "dc", this element SHALL be ommitted. Value rules of Table 56 SHALL be applied.	In case of *powerSupplyType* = "ac", this element states the kind of ac measurement is done (e.g. "real").
acMeasurementVariant	In case of *powerSupplyType* = "dc", this element SHALL be ommitted. Value rules of Table 57 SHALL be applied.	In case of *powerSupplyType* = "ac", this element states the variation of the ac measurement (e.g. "amplitude").
acMeasuredHarmonic	In case of *powerSupplyType* = "dc", this element SHALL be ommitted.	In case of *powerSupplyType* = "ac", this element states the harmonic, which is measured. A value of "1" indicates that the normal value is measured.
scopeType	None.	A certain meaning of the parameter combination.
label	Default rules of section 3.10.1.2.1 SHALL be applied.	A label for this electrical connection parameter.
description	Default rules of section 3.10.1.3.1 SHALL be applied.	A description for this electrical connection parameter.

Table 52: electricalConnectionParameterDescriptionListData Element rules

Element "**voltageType**" value rules:

Value	Rule	Description
ac	SHALL NOT be used if *powerSupplyType* in *electricalConnectionDescriptionData* has the value "dc".	Alternating current.
dc	None.	Direct current.

Table 53: Element "voltageType" value rules

Element "**acMeasuredPhases**" value rules:

Value	rule	Description
a	None.	Phase a (or L1).
b	None.	Phase b (or L2).
c	None.	Phase c (or L3).
ab	None.	Phase a and b (or L1 and L2).

bc	None.	Phase b and c (or L2 and L3).
ac	None.	Phase a and c (or L1 and L3).
abc	None.	Phase a, b and c (or L1, L2 and L3).
neutral	None.	The neutral conductor of a connection.
ground	None.	The ground conductor of a connection.
none	None.	No phase specified or available.

Table 54: Element "acMeasuredPhases" value rules

Element "**acMeasuredInReferenceTo**" value rules:

Value	**Description**	**Description**
a	None.	Phase a (or L1).
b	None.	Phase b (or L2).
c	None.	Phase c (or L3).
ab	None.	Phase a and b (or L1 and L2).
bc	None.	Phase b and c (or L2 and L3).
ac	None.	Phase a and c (or L1 and L3).
abc	None.	Phase a, b and c (or L1, L2 and L3).
neutral	None.	The neutral conductor of a connection.
ground	None.	The ground conductor of a connection.
none	None.	No phase specified or available.

Table 55: Element "acMeasuredInReferenceTo" value rules

Element "**acMeasurementType**" value rules:

Value	**Description**	**Description**
real	None.	AC measurement of the real part.
reactive	None.	AC measurement of the reactive part.
apparent	None.	AC measurement of the apparent part.
phase	None.	AC measurement of the phase.

Table 56: Element "acMeasurementType" value rules

Element "**acMeasurementVariant**" value rules:

Value	**Description**	**Description**
amplitude	None.	Measurement of the amplitude.
rms	None.	Measurement of the RMS (Root Mean Square).
instantaneous	None.	Measurement of the instantaneous value.
angle	None.	Measurement of the angle.
cosPhi	None.	Measurement of the cosine phi.

Table 57: Element "acMeasurementVariant" value rules

4.3.10.5 electricalConnectionPermittedValueSetListData

4.3.10.5.1 Description

If an electrical connection parameter has some defined value(s) or value range(s) that are allowed or possible, they can be modelled with this function. It enables the server to state single values, value ranges or combinations of both.

2882

2883 *4.3.10.5.2 Rules*
2884 None.

2885

2886 *4.3.10.5.3 Element rules*

Element	Rule	Description
electricalConnectionId	SHALL be set as PRIMARY IDENTIFIER.	Identifier of the electrical connection, this rather static information belongs to.
parameterId	SHALL be set as SUB IDENTIFIER.	Relation to the according parameter.
permittedValuesSet (list)	At least one set of permitted values SHALL be stated.	The permittedValueSet allows to define an arbitrary set of permitted data. Compared to a simple value range the permittedValueSet also may have gaps. Please refer to the ScaledNumberSetType for more details.

2887 *Table 58: electricalConnectionPermittedValueSetListData Element rules*

2888

2889 ***4.3.10.6 electricalConnectionStateListData***

2890 *4.3.10.6.1 Description*
2891 Some electrical connection specific state information is modelled in this function.

2892

2893 *4.3.10.6.2 Rules*
2894 None.

2895

2896 *4.3.10.6.3 Element rules*

Element	Rule	Description
electricalConnectionId	SHALL be set as PRIMARY IDENTIFIER.	Electrical connection, this state information belongs to.
timestamp	None.	Point in time, the state information was generated.
currentEnergyMode	For further value rules see section 3.10.1.16.1.	Whether a device is consuming, producing or idle is stated by this element.
consumptionTime	SHOULD be set to the duration the device is in energy mode "consume" since the last reboot.	The duration the device is in energy mode "consume" since the last reboot (see section 5.3.9.2).
productionTime	SHOULD be set to the duration the device is in energy mode "produce" since the last reboot.	The duration the device is in energy mode "produce" since the last reboot (see section 5.3.9.2).

totalConsumptionTime	SHOULD be set to the duration, the device is in energy mode "consume" since its *installationTime*.	The duration the device is in energy mode "consume" since its *installationTime* (see section 5.3.9.4).
totalProductionTime	SHOULD be set to the duration, the device is in energy mode "producing" since its *installationTime*.	The duration the device is in energy mode "producing" since its *installationTime* (see section 5.3.9.4).

Table 59: electricalConnectionStateListData Element rules

4.3.10.7 electricalConnectionDescriptionListData

4.3.10.7.1 Description

Some electrical connection specific information, that are rather static, are modelled within this function.

4.3.10.7.2 Rules

None.

4.3.10.7.3 Element rules

Element	Rule	Description
electricalConnectionId	SHALL be set as PRIMARY IDENTIFIER.	Electrical connection, this information belongs to.
powerSupplyType	SHOULD be set.	States whether the electrical connection is of type "ac" or "dc".
acConnectedPhases	If *powerSupplyType* = "ac", this element SHOULD be set.	If powerSupplyType="ac", this element contains the number of phases, this electrical connection has (typically 1, 2 or 3).
acRmsPeriodDuration	None.	If the "rms" value of a parameter combination (see element *acMeasurementVariant* in section 5.3.11.2.2) is measured, this element states the duration, the rms value is determined in.
positiveEnergyDirection	SHOULD be set. If it is omitted, it SHALL be interpreted as *positiveEnergyDirection* = "consume".	This element states whether energy consumption or production will be counted as positive value.
label	Default rules of section 3.10.1.2.1 SHALL be applied.	A label for this electrical connection.
description	Default rules of section 3.10.1.3.1 SHALL be applied.	A description for this electrical connection.

Table 60: electricalConnectionDescriptionListData Element rules

4.3.10.8 Descriptive information and further definitions

None.

4.3.11 Generic

4.3.11.1 Dependencies to other Feature Types

None.

4.3.11.2 Introduction

None.

4.3.11.3 Rules

None.

4.3.11.4 Function and Identifier rules

None.

4.3.11.5 Element rules

None.

4.3.11.6 Descriptive information and further definitions

The Generic Feature Type does not contain any information. It SHALL only be used as a client feature! Such a client MAY send any read, write or call messages to servers from this feature. This can be very useful if a client wants to communicate with different Feature Types from the same address. Additionally, unneccessary duplications of server Feature Types on client side are avoided (e.g. it is not needed to have a Measurement and a Setpoint client Feature; instead, only this Generic Feature is used by the client). Therefore, it is preferable that a client uses the Generic Feature Type.

4.3.12 HVAC

4.3.12.1 Dependencies to other Feature Types

- Measurement
- PowerSequences
- Setpoint
- SmartEnergyManagementPs
- TimeTable

4.3.12.2 Introduction

Heating, Ventilation and Air Conditioning (HVAC) devices have some special functionality which is covered by this class.

All system functions (like heating, cooling, domestic hot water preparation (dhw), ventilation, etc.) have their specific HVAC operation modes (like on, off, auto, etc.). Relations between these are modelled as well as relations to (optional) power sequences, which represent a concrete run of a device's program.

HVAC overruns (like an one time domestic-hot-water (DHW) preparation), which do have an impact on the normal operation mode of a HVAC system function, are modelled in further SPINE functions of the HVAC Class.

4.3.12.3 Rules

None.

4.3.12.4 hvacSystemFunctionListData

4.3.12.4.1 Description

The hvacSystemFunctionListData function announces the current status of the HVAC system functions, including the current HVAC operation mode, a flag that indicates, if the HVAC operation mode can be changed by some external communications partner (client) and a flag that indicates whether an HVAC overrun is currently active or not.

4.3.12.4.2 Rules

None.

4.3.12.4.3 Element rules

Element	Rule	Description
systemFunctionId	SHALL be set as PRIMARY IDENTIFIER.	Identifier for a specific HVAC system function.
currentOperationModeId	SHALL be set as FOREIGN IDENTIFIER to the according operationModeId of the currently selected operation mode. If the function hvacSystemFunctionOperationModeRelationListData is used, only a operationModeId value related there to the according systemFunctionId SHALL be used here.	States which HVAC operation mode is currently active for this specific HVAC system function
isOperationModeIdChangeable	If set to "true", a client MAY change the currentOperationModeId. If absent or set to	Denotes whether the

	"false" currentOperationModeId SHALL NOT be writeable.	current HVAC operation mode identifier is changeable (by some external communication partner) or not.
currentSetpointId	The setpointId SHALL be set here as FOREIGN IDENTIFIER, if a setpoint is linked to the system function. If the function hvacSystemFunctionSetpointRelationListData is used, only a setpointId value related there to the according systemFunctionId SHALL be used here.	States which setpoint is active for this specific HVAC system function
isSetpointIdChangeable	If no currentSetpointId is set, this element SHALL NOT be set. If set to "true", a client MAY change the currentSetpointId. If absent or set to "false" currentSetpointId SHALL NOT be writeable.	Denotes whether the current setpoint identifier is changeable (by some external communication partner) or not.
isOverrunActive	If an overrun is active that has an effect on this system function, this element SHOULD be set to "true".	If an HVAC overrun is currently active, this element is set to *true*. The HVAC overrun itself is modelled with other functions (see section 5.3.12.8 and 5.3.12.9) and may be located on a different feature or even a different entity or device.

Table 61: hvacSystemFunctionListData Element rules

4.3.12.5 hvacSystemFunctionOperationModeRelationListData

4.3.12.5.1 Description

Each HVAC system function has its own relation to all for this HVAC system function available HVAC operation modes. E.g. there may be three HVAC operation modes available on the feature: *on*, *off*, *auto* (see section 5.3.12.7). HVAC System function 1 only supports HVAC operation mode 1 (*on*) and 2 (*off*). HVAC System function 2 supports all three HVAC operation modes. The enumeration of the available HVAC system functions is described in section 5.3.12.6.2.

4.3.12.5.2 Rules

None.

4.3.12.5.3 Element rules

Element	Rule	Description
systemFunctionId	SHALL be set as PRIMARY IDENTIFIER.	Identifier for a specific HVAC system function.
operationModeId (list)	SHALL be set as FOREIGN IDENTIFIER (at least one) to the according operationModeId of the referenced operation mode.	Lists the HVAC operation modes that are available (selectable) for a specific HVAC system function.

Table 62: hvacSystemFunctionOperationModeRelationListData Element rules

4.3.12.6 hvacSystemFunctionSetpointRelationListData

4.3.12.6.1 Description

The setpoint class (see section 5.3.21) may be used to model some HVAC relevant setpoints like desired room temperature. An HVAC system function may relate to some of these setpoints. The relation is modelled with this SPINE function.

4.3.12.6.2 Rules

None.

4.3.12.6.3 Element rules

Element	Rule	Description
systemFunctionId	SHALL be set as PRIMARY IDENTIFIER.	Identifier for a specific HVAC system function.
operationModeId	SHALL be set as SUB IDENTIFIER.	Identifier for a specific HVAC operation mode.
setpointId (list)	MAY be set as FOREIGN IDENTIFIER to the according setpointId of the referenced setpoint.	List of setpoints that are related with a specific HVAC system function.

Table 63: hvacSystemFunctionSetpointRelationListData Element rules

2997

2998 ***4.3.12.7 hvacSystemFunctionPowerSequenceRelationListData***

2999 *4.3.12.7.1 Description*

3000 Power sequences (see section 5.3.19) describe tasks of a functionality in detail. An HVAC system
3001 function may relate to some of these power sequences. The relation is modelled with this function.

3002

3003 *4.3.12.7.2 Rules*

3004 None.

3005

3006 *4.3.12.7.3 Element rules*

Element	Rule	Description
systemFunctionId	SHALL be set as PRIMARY IDENTIFIER.	Identifier for a specific HVAC system function.
sequenceId (list)	MAY be set as FOREIGN IDENTIFIER to the according sequenceId of the referenced power sequence.	List of power sequences that are related with a specific HVAC system function.

3007 *Table 64: hvacSystemFunctionPowerSequenceRelationListData Element rules*

3008

3009 ***4.3.12.8 hvacSystemFunctionDescriptionListData***

3010 *4.3.12.8.1 Description*

3011 The rather static information about the HVAC system functions is modelled within this SPINE
3012 function.

3013

3014 *4.3.12.8.2 Rules*

3015 None.

3016

3017 *4.3.12.8.3 Element rules*

Element	Rule	Description
systemFunctionId	SHALL be set as PRIMARY IDENTIFIER.	Identifier for a specific HVAC system function.
systemFunctionType	SHALL be set. Value rules of Table 66 SHALL be applied.	Denotes the type of the HVAC system function.
label	Default rules of section 3.10.1.2.1 SHALL be applied.	A short label of the HVAC system function.
description	Default rules of section 3.10.1.3.1 SHALL be applied.	Descriptive information on the HVAC system function.

3018 *Table 65: hvacSystemFunctionDescriptionListData Element rules*

3019 Element "**systemFunctionType**" value rules:

Value	Rule	Description
heating	None.	HVAC system function for space heating.
cooling	None.	HVAC system function for space cooling.
ventilation	None.	HVAC system function for ventilation.
dhw	None.	HVAC system function for domestic hot water preparation.

Table 66: Element "systemFunctionType" value rules

4.3.12.9 hvacOperationModeDescriptionListData

4.3.12.9.1 Description

HVAC operation modes are used to describe which behaviour the HVAC system functions have. Each HVAC operation mode may be used by several HVAC system functions.

4.3.12.9.2 Rules

None.

4.3.12.9.3 Element rules

Element	Rule	Description
operationModeId	SHALL be set as PRIMARY IDENTIFIER.	Identifier of the HVAC operation mode.
operationModeType	SHALL be set. Value rules of Table 68 SHALL be applied.	Denotes the type of the HVAC operation mode.
label	Default rules of section 3.10.1.2.1 SHALL be applied.	A short label of the HVAC operation mode.
description	Default rules of section 3.10.1.3.1 SHALL be applied.	Descriptive information on the HVAC operation mode.

Table 67: hvacOperationModeDescriptionListData Element rules

Element "**operationModeType**" value rules:

Value	Rule	Description	
auto	None.	**HVAC System Function**	**Meaning**
		heating / cooling / dhw / ventilation	HVAC System function is controlled depending on a time table (i.e. the current setpoint for the HVAC system function is changed depending on a time table)
on	None.	**HVAC System Function**	**Meaning**
		heating / cooling / dhw / ventilation	HVAC System function is controlled (e.g. Setpoint) independant from time tables (i.e. the current setpoint for the HVAC system function is not depending on any time table).
off	None.	**HVAC System Function**	**Meaning**
		heating / dhw	Protection mode against freezing (no physically turning off). The (temperature)

			setpoint for freezing protection may be stated with a setpointId.
		cooling	Space cooling function not active.
		ventilation	Ventilation function not active (protection functions like humidity protection may be active).
eco	None.	**HVAC System Function**	**Meaning**
		Heating / cooling / dhw / ventilation	HVAC System function is controlled (e.g. current Setpoint) independant from time tables (i.e. the current setpoint for the HVAC system function is not depending on any time table). HVAC System function is controlled by a different setpoint as for “on”

Table 68: Element "operationModeType" value rules

4.3.12.10 hvacOverrunListData

4.3.12.10.1 Description

The normal HVAC operation may be overrided by some HVAC specific functionality. There are some HVAC overruns defined by the EEBus Initiative e.V., but vendors may add further vendor-specific HVAC overruns, too (which are then out of interoperability scope).

When the HVAC overrun is active, some (not all! See section 5.3.12.9) HVAC system functions have a specific behaviour, other than the normal operation would be. After the HVAC overrun is inactive again, the HVAC system returns to its normal operation.

4.3.12.10.2 Rules

None.

4.3.12.10.3 Element rules

Element	Rule	Description
overrunId	SHALL be set as PRIMARY IDENTIFIER.	Identifier for the HVAC overrun.
overrunStatus	SHALL be set. Value rules of Table 70 SHALL be applied.	The current status of the HVAC overrun.
timeTableId	None.	Reference to a time table. Needed, if an overrun has some start and end point(s).
isOverrunStatusChangeable	If absent, the default value of "false" will be assumed.	Whether the overrun status is changeable (element set to "true") by a client or not (element set to "false") is denoted in this element.

Table 69: hvacOverrunListData Element rules

Element "**overrunStatus**" value rules:

Value	Rule	Description
active	None.	HVAC overrun triggered, but not started yet.
running	None.	HVAC overrun running.
finished	MAY only be used as notification directly after HVAC overrun was finished. "overrunStatus" SHALL be changed to "inactive" after that. SHOULD NOT be used as part of a reply message!	HVAC overrun finished.
inactive	MAY be used as reply or as notification. The latter only, if HVAC overrun was not finished in a usual way (deactivated by user, e.g.).	HVAC overrun inactive.

Table 70: Element "overrunStatus" value rules

4.3.12.11 hvacOverrunDescriptionListData

4.3.12.11.1 Description

Each HVAC overrun is defined by its description part. Here, the HVAC overrun type is specified as well as the list of HVAC system functions that are affected by the HVAC overrun (if this information is available).

4.3.12.11.2 Rules

None.

4.3.12.11.3 Element rules

Element	Rule	Description
overrunId	SHALL be set as PRIMARY IDENTIFIER.	Identifier for the HVAC overrun.
overrunType	SHALL be set. Value rules of Table 72 SHALL be applied.	The type of the overrun.
affectedSystemFunctionId (list)	If system functions are affected by this overrun, their according systemFunctionId SHOULD be stated here.	Identifiers of all HVAC system functions that are affected by this HVAC overrun.
label	Default rules of section 3.10.1.2.1 SHALL be applied.	A short label of the HVAC overrun.
description	Default rules of section 3.10.1.3.1 SHALL be applied.	Descriptive information on the HVAC overrun.

Table 71: hvacOverrunDescriptionListData Element rules

Element "**overrunType**" value rules:

Value	Rule	Description

oneTimeDhw	None.	A one-time domestic hot water loading (independend from current HVAC operation mode or time table).
party	None.	During a party, the HVAC system will stay in the normal day-operation mode and not turn off at the usual time. Example: HVAC system function “heating” is controlled by the day setpoint, independend from operation mode and time table, until HVAC overrun “party” is finished.
sgReadyCondition1	None.	The SGReady condition 1 will be applied. Please consider [SGReady].
sgReadyCondition3	None.	The SGReady condition 3 will be applied. Please consider [SGReady].
sgReadyCondition4	None.	The SGReady condition 4 will be applied. Please consider [SGReady].
oneDayAway	None.	Overrides the normal daily routine to a out-of-home configuration (e.g. eco-mode the whole day).
oneDayAtHome	None.	Overrides the normal daily routine to an at-home configuration (e.g. on-mode the whole day).
oneTimeVentilation	None.	Activates the ventilation once, independent from the configured routine.
hvacSystemOff	None.	Deactivates the HVAC system (frost protection is still active and the system is in a network-standby mode, in which it is reachable for re-configuration).

Table 72: Element "overrunType" value rules

4.3.12.12 Descriptive information and further definitions

Relation between "overrunStatus" element in function "hvacOverrunListData" and "isOverrunActive" element in function "hvacSystemFunctionListData":

overrunStatus	**isOverrunActive**
active	false
running	true
finished	false
inactive	false

Table 73: Relation between "overrunStatus" and "isOverrunActive"

The content of the according elements SHALL match the relation stated above.

4.3.13 Identification

4.3.13.1 Dependencies to other Feature Types

None.

4.3.13.2 Introduction

The Identification Class can be used to express certain identification values of the parent Entity.

4.3.13.3 Rules

None.

4.3.13.4 identificationListData

4.3.13.4.1 Description

The identificationListData function allows to express certain identification values of the corresponding Entity (the parent Entity of the Identification Feature).

4.3.13.4.2 Rules

None.

4.3.13.4.3 Element rules

Element	Rule	Description
identificationId	SHALL be set as PRIMARY IDENTIFIER.	The identificationId is used to identify the different list entries within IdentificationListData.
identificationType	If set, the value rules of the IdentificationTypeEnumType SHALL apply.	The identificationType allows to describe a certain type of identification, e.g. a MAC address or an RFID tag.
identificationValue	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.	This holds the value of the identification and therefore is the essential element of IdentificationData.

Table 74: identificationListData Element rules

Element "**identificationType**" value rules:

Value	Rule	Description
eui48	The identificationValue SHALL be interpreted as eui48 MAC address. The following pattern SHALL be used: `(([A-F]|[0-9]){2}\-){5}([A-Z]|[0-9]){2}`	eui48 MAC address
eui64	The identificationValue SHALL be interpreted as eui64 MAC address. The following pattern SHALL be used: `(([A-F]|[0-9]){2}\-){7}([A-Z]|[0-9]){2}`	eui64 MAC address

Table 75: Element "identificationType" value rules

4.3.14 IncentiveTable

4.3.14.1 Dependencies to other Feature Types

None.

4.3.14.2 Introduction

The IncentiveTable data model allows to define tiers with different incentives and boundaries. In the case the incentive is price based, it also could be called a priceTable. Within the boundaries of a tier the incentives of the tier apply. This allows to distribute e.g. different types of power to different power consumers. E.g. a photovoltaic system produces more power than can be consumed and feed into the electricity grid (surplus power), in this case the power is for free and can be distributed by the CEM to consumers that have the flexibility to increase their consumption. When all surplus power is consumed the next tier would be the electricity grid feed-in power, which in this example would be cheaper then the electricity grid power. In the given example we would have three tiers, 1. surplus power on top of that the 2. electricity grid feed in power and beyond that the 3. electricity grid power. However, the electricity grid power can also get cheaper, e.g. in case the electricity grid has too much power available an incentive to flexible consumers could be given.

Although in this example power was used, the IncentiveTable may also be used for other things.

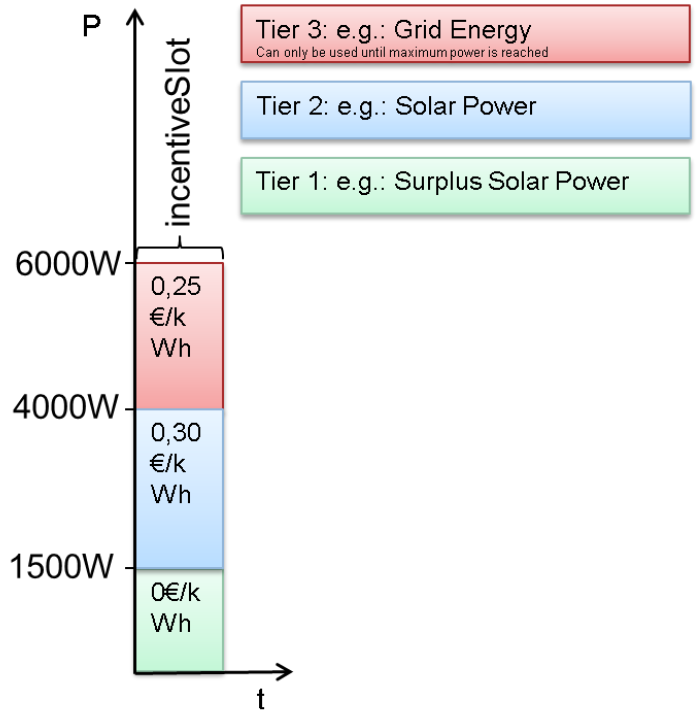


Figure 39: IncentiveTable example (single incentiveSlot) for different tiers

Example 1: If the actual power value is 5000W:

- 1500W are for free
- 2500W cost 0,30€/kWh
- and the last 1000W cost 0,25€/kWh.

Example 2: If the actual power value is 1000W:

- Only the lowest tier is used
- the 1000W are for free.

Example 3: If the actual power value is 2000W:

- 1500W are for free
- and the last 500W cost 0,30€/kWh.

The IncentiveTable has three different functions:

1. incentiveTableData: holds the actual values of the boundaries and incentives
2. incentiveTableDescriptionData: holds data that describes the values of incentiveTable
3. incentiveTableConstraintsData: allows a server to define certain constraints regarding the incentiveTableData, this is especially important if the IncentiveTable is writeable

4.3.14.3 Rules

Each tier is uniquely defined by the value pair of "tariffId" and "tierId".

4.3.14.4 incentiveTableData

4.3.14.4.1 Description

The incentiveTable within the incentiveTableData function holds the values of the IncentiveTable. This includes the boundary values as well as the incentive values for each tier.

The Function "incentiveTableData" models rather time dependent information of each tier within the respective element "tier". However, the time dependency is not modelled within the "tier" elements directly. Instead, each "tariff" is split into time slots which are expressed by the element "incentiveSlot". This means each "incentiveSlot" entry finally contains time dependent data of one or more "tier" entries.

4.3.14.4.2 Rules

If the same tier is used within different "incentiveSlot" entries the same "tierId" value SHALL be used in these "incentiveSlot" entries.

Each tier defines its supply (difference between upperBoundaryValue and lowerBoundaryValue) that can be consumed or produced for the incentive(s) (e.g. price) given by the tier. If the boundaries of a

tier are negative the supply can be produced. If the boundaries of a tier are positive the supply can be consumed. The supply of each tier can only be consumed or produced, if the supply of the tiers between this tier and zero are already consumed.

If (e.g.) the lowerBoundaryValue of tier x is set to 2 units and the upperBoundaryValue is 4 units, tier x can provide a supply of 4-2 = 2 units. These 2 units can be consumed for the incentive(s) given by tier x. If the actual value of a device surpasses the upperBoundaryValue, the device consumes the whole 2 units of tier x for the corresponding incentive(s). If the actual value of a device is below the upperBoundaryValue but above the lowerBoundaryValue (e.g. 3 units), the device consumes the difference between the actual value and the lowerBoundaryValue of tier x (e.g. 3-2 = 1) for the incentive(s) given by the tier. As negative values are used for production, the meaning of the boundaries switch. For production the lowerBoundaryValue of a tier will have the higher distance to zero then the upperBoundaryValue. Therefore, the upperBoundaryValue for production has the same meaning as the lowerBoundaryValue for consumption and the lowerBoundaryValue for production has the same meaning as the upperBoundaryValue for consumption.

For more examples, please refer to Figure 39.

4.3.14.4.3 Element rules

Element	Rule	Description
incentiveTable (list)		
incentiveTable. tariff		
incentiveTable. tariff. tariffId	SHALL be set as PRIMARY IDENTIFIER.	Allows the identification of a incentiveTable. Allows linking of the different functions within the IncentiveTable Feature. Allows linking of the incentiveTable within other Features that are placed in the same entity.
incentiveTable. incentiveSlot (list)		
incentiveTable. incentiveSlot. timeInterval	SHALL be set if an incentive has a time restriction or the incentiveTable contains more than one incentiveSlot entry. The timeInterval of different incentiveSlots within an incentiveTable SHALL NOT overlap in time.	Time period of the incentiveSlot.
incentiveTable. incentiveSlot. timeInterval. timeSlotId	MAY be used as SUB IDENTIFIER of tariffId to allow partial modification of the incentiveSlot list if slotIdSupport is set to “true” for this incentiveTable. SHALL NOT be used if the Element "slotIdSupport" within the Function "incentiveTableDescriptionData"	Allows identification of an incentiveSlot within an incentiveTable.

	is not set to "true". If used in a incentiveSlot it SHALL be used in all incentiveSlots of an incentiveTable. Please refer to section 3.4.2.6 for details.	
incentiveTable. incentiveSlot. timeInterval. startTime	SHALL be set.	
incentiveTable. incentiveSlot. timeInterval. startTime. dateTime		Absolute start time.
incentiveTable. incentiveSlot. timeInterval. startTime. relative		Relative start time containing a value relative to now. Positive values are in the future, negative ones are in the past and zero seconds represents "now".
incentiveTable. incentiveSlot. timeInterval. endTime	SHALL be set for the last slot. SHALL be set if a time gap follows after this slot. MAY be set for all other slots.	
incentiveTable. incentiveSlot. timeInterval. endTime. dateTime		Absolute end time.
incentiveTable. incentiveSlot. timeInterval. endTime. relative		Relative end time containing a value relative to now. Positive values are in the future, negative ones are in the past and zero seconds represents "now".
incentiveTable. incentiveSlot. tier (list)		
incentiveTable. incentiveSlot. tier. tier		
incentiveTable. incentiveSlot. tier. tier. tierId	SUB IDENTIFIER of tariffId to identify a tier of a tariff. SHALL be set if the incentiveTable contains more than one tier. If the tierId is omitted, a default value of "1" SHALL be applied.	Allows identification of a tier within a incentiveTable.
incentiveTable. incentiveSlot. tier. boundary (list)	The boundary list defines the boundaries of a tier. The boundary range of different tiers within an incentiveSlot defined by lowerBoundaryValue and upperBoundaryValue SHALL NOT overlap for boundaries with the same boundaryId. The boundary range of a tier includes all values that are equal or greater than lowerBoundaryValue and lower than upperBoundaryValue.	

incentiveTable. incentiveSlot. tier. boundary. boundaryId	SHALL be used as SUB IDENTIFIER of tierId. SHALL be set if the tier contains more than one boundary entry. If the boundaryId is omitted, a default value of "1" SHALL be applied.	Allows identification of a boundary within a tier.
incentiveTable. incentiveSlot. tier. boundary. lowerBoundaryValue	Within a boundary lowerBoundaryValue SHALL be smaller than upperBoundaryValue. **Lower and upper boundaries of adjacent tiers**: If lowerBoundaryValue is omitted and upperBoundaryValue is set, the lowerBoundaryValue SHALL be interpreted as if it is equal (without a gap) to the upperBoundaryValue of the next lower tier within the incentiveSlot. Gaps between boundaries of adjacent "tier" instances can only be modelled by setting the upperBoundaryValue of the lower "tier" AND the lowerBoundaryValue of the higher "tier" instance explicitly. **Minus infinity lowerBoundary rule**: If lowerBoundaryValue is omitted and there exists no lower tier with a lowerBoundaryValue or upperBoundaryValue within the incentiveSlot, the omitted lowerBoundaryValue SHALL be interpreted as minus infinity.	This represents the lower end value range boundary.
incentiveTable. incentiveSlot. tier. boundary. upperBoundaryValue	Within a boundary lowerBoundaryValue SHALL be smaller than upperBoundaryValue. **Lower and upper boundaries of adjacent tiers**: If upperBoundaryValue is omitted and lowerBoundaryValue is set, the upperBoundaryValue SHALL be interpreted as equal (without any gap) to the lowerBoundaryValue of the next higher tier within the incentiveSlot. Gaps between boundaries of adjacent "tier"	This represents the upper end value range boundary.

	instances can only be modelled by setting the upperBoundaryValue of the lower "tier" AND the lowerBoundaryValue of the higher "tier" instance explicitly. **Infinity upperboundary rule**: If upperBoundaryValue is omitted and there exists no higher tier with a lowerBoundaryValue or upperBoundaryValue within the incentiveSlot, the omitted upperBoundaryValue SHALL be interpreted as infinity.	
incentiveTable. incentiveSlot. tier. incentive (list)		
incentiveTable. incentiveSlot. tier. incentive. incentiveId	SHALL be used as SUB IDENTIFIER of tierId. SHALL be set if the tier contains more than one incentive entry. If the incentiveId is omitted, a default value of "1" SHALL be applied.	Allows identification of an incentive within a tier.
incentiveTable. incentiveSlot. tier. incentive. value		The value of the incentive.

Table 76: incentiveTableData Element rules

4.3.14.5 incentiveTableDescriptionData

4.3.14.5.1 Description

The incentiveTableDescription within the incentiveTableDescriptionData function holds the rather static description of the IncentiveTable. This includes the tier, boundary and incentive type as well as further elements.

4.3.14.5.2 Rules

None.

4.3.14.5.3 Element rules

Element	Rule	Description
incentiveTableDescription (list)		
incentiveTableDescription. tariffDescription		
incentiveTableDescription. tariffDescription. tariffId	SHALL be set as PRIMARY IDENTIFIER of an incentiveTable, if more than one incentiveTable is defined within IncentiveTable.	Allows the identification of a incentiveTable. Allows linking of the data in the different functions of IncentiveTable. Allows linking of the incentiveTables within

		other features that are placed in the same entity.
incentiveTableDescription. tariffDescription. measurementId	SHALL be set as FOREIGN IDENTIFIER, if a measurand or other feature is linked with the measurementId	If a measurand is linked to the incentive table, its identifier is denoted here.
incentiveTableDescription. tariffDescription. tariffWriteable	SHALL be set to "true" if the incentive table is writeable by a client. Otherwise it may be omitted.	Whether the incentive table is writeable by a client or not can be denoted in this element.
incentiveTableDescription. tariffDescription. updateRequired	With updateRequired the server can request an update of writeable or changeable data related to the same PRIMARY IDENTIFIER from a client. The server SHALL ensure that only one responsible client is able to update the related data. To request an update the server SHALL set updateRequired to “true”. Note: In this case, the server expects the responsible client to update the writeable or changeable data related to the same PRIMARY IDENTIFIER. However, also if updateRequired is set to “false” a server SHOULD in general allow updates of the data from the responsible client. The server SHALL set the updateRequired back to “false”, as soon as “incentiveTableDescriptionListData” was updated successfully (if writeable or changeable) OR the update of the other writeable or changeable data related to the same PRIMARY IDENTIFIER was successful. Note: The client does not need to stop an ongoing update process (e.g. if multiple functions are written), when updateRequired is set back to “false”. The server MAY choose to withdraw the update request at any time by setting updateRequired back to “false”.	Allows to request an update from the bound client.
incentiveTableDescription. tariffDescription. scopeType		A scopeType for the incentive table can be stated here.
incentiveTableDescription. tariffDescription. label	Default rules of section 3.10.1.2.1 SHALL be applied.	A short label on the incentive table.
incentiveTableDescription. tariffDescription. description	Default rules of section 3.10.1.3.1 SHALL be applied.	A description for the incentive table.

incentiveTableDescription. tariffDescription. slotIdSupport	If omitted, a default of “false” SHALL be applied. If set to “false” no incentiveSlot SHALL contain a timeSlodId. If set to “true” timeSlotId MAY be used within the incentiveSlot as described for the element timeSlotId within the function incentiveTableData.	Indicates support of timeSlotId within the corresponding tariff in the Function incentiveTableData.
incentiveTableDescription. tier (list)		Allows to describe tiers and the incentives and boundaries that are used by the tier. Please refer to the detailed description of *IncentiveTableDescriptionTierType elements. See* Table 152.
incentiveTableDescription. tier. tierDescription		
incentiveTableDescription. tier. tierDescription. tierId	SHALL be set as SUB IDENTIFIER of tariffId, if the incentiveTable contains more than one tier. If the tierId is omitted, a default value of "1" SHALL be applied.	Allows identification of a tier within a incentiveTable.
incentiveTableDescription. tier. tierDescription. tierType	TierTypeEnumType value rules SHALL apply, described below this table.	Allows to define the type of the tier.
incentiveTableDescription. tier. tierDescription. label	Default rules of section 3.10.1.2.1 SHALL be applied.	Allows to define user friendly name for the tier.
incentiveTableDescription. tier. tierDescription. description	Default rules of section 3.10.1.3.1 SHALL be applied.	Allows to define a user friendly description for the tier.
incentiveTableDescription. tier. boundaryDescription (list)		
incentiveTableDescription. tier. boundaryDescription. boundaryId	SHALL be set as SUB IDENTIFIER of tierId, if the tier contains more than one boundary entry. If the boundaryId is omitted, a default value of "1" SHALL be applied.	Allows identification of a boundary within a incentiveTable.
incentiveTableDescription. tier. boundaryDescription. boundaryType	TierBoundaryTypeEnumType value rules SHALL apply, described below this table. There SHALL NOT be multiple boundaries with the same boundaryType for the same tier.	Allows to define the type of the boundary.
incentiveTableDescription. tier. boundaryDescription. switchToTierIdWhenLower	FOREIGN IDENTIFIER used to refer to the tier that is going to be active as soon as the lowerBoundaryValue is underrun.	Link to the tier that will become active if the lowerBoundaryValue is underrun.
incentiveTableDescription. tier. boundaryDescription. switchToTierIdWhenHigher	FOREIGN IDENTIFIER used to refer to the tier that is going to be active as soon as the upperBoundaryValue is exceeded.	Link to the tier that will become active if the upperBoundaryValue is exceeded.

incentiveTableDescription. tier. boundaryDescription. boundaryUnit	If set, the unit SHALL be applied to the lowerBoundaryValue and upperBoundaryValue.	Allows to define the unit of the boundary values.
incentiveTableDescription. tier. incentiveDescription (list)		
incentiveTableDescription. tier. incentiveDescription. incentiveId	SHALL be set as SUB IDENTIFIER of tierId, if the tier contains more than one incentive entry. If the incentiveId is omitted, a default value of "1" SHALL be applied.	Allows identification of an incentive within a incentiveTable.
incentiveTableDescription. tier. incentiveDescription. incentiveType	IncentiveTypeEnumType value rules SHALL apply, described below this table.	Allows to define the type of the incentive.
incentiveTableDescription. tier. incentiveDescription. incentivePriority	Values SHALL be greater than zero. A value of "1" SHALL be interpreted as highest priority. The priority decreases with increasing value of incentivePriority.	Allows to define the priority of the incentive. E.g. the priority that the customer has assigned for this incentive.
incentiveTableDescription. tier. incentiveDescription. currency	If set, the currency SHALL be applied to the value of the incentive	Monetary incentives may need to describe a currency.
incentiveTableDescription. tier. incentiveDescription. unit	If set, the unit SHALL be applied to the value of the incentive	Allows definition of a unit, if the incentive has a unit.
incentiveTableDescription. tier. incentiveDescription. label	Default rules of section 3.10.1.2.1 SHALL be applied.	Allows to express a user friendly label for the incentive.
incentiveTableDescription. tier. incentiveDescription. description	Default rules of section 3.10.1.3.1 SHALL be applied.	Allows to express a user friendly description for the incentive.

3178 *Table 77: incentiveTableDescriptionData Element rules*

3179 Element "**incentiveTableDescription. tier. tierDescription. tierType**" value rules:

Value	Rule
fixedCost	The tier has a cost incentive that SHALL NOT vary over time
dynamicCost	The tier has a cost incentive that MAY vary over time

3180 *Table 78: Element "tier. tierDescription. tierType" value rules*

3181 Element "**incentiveTableDescription. tier. boundaryDescription. boundaryType**" value rules:

Value	Rule
powerBoundary	The boundary SHALL be an electrical power. E.g. if the surplus power is consumed, more expensive power has to be used. Positive powerBoundary values describe which power can be consumed with certain incentives, while negative powerBoundary values describe which power can be produced with certain incentives.
energyBoundary	The boundary SHALL be an electrical energy. E.g. if the stored self produced energy in a battery is consumed, more expensive energy has to be used.

	Positive energyBoundary values describe which energy can be consumed with certain incentives, while negative energyBoundary values describe which energy can be produced with certain incentives.
countBoundary	The boundary SHALL be a count. E.g. after a certain count the price of a good may be reduced.

Table 79: Element "tier. boundaryDescription. boundaryType" value rules

Element "**incentiveTableDescription. tier. incentiveDescription. incentiveType**" value rules:

Value	Rule	Description
absoluteCost	Each related value SHALL be an absolute cost with a specific currency. Positive values SHALL be interpreted as costs. Negative values SHALL be interpreted as profits. If also a unit is set, the cost SHALL contain the cost per unit. E.g. EUR/Wh.	Absolute costs with a currency and a unit. The total costs can be calculated.
relativeCost	Positive values SHALL be interpreted as costs. Negative values SHALL be interpreted as profits. The values SHALL have no currency and unit SHALL be set to percentage (pct). E.g. the price is relative to the fixed price in the tariff contract. The value SHALL be in the range from 0% to 100%.	Relative costs that cannot be interpreted as some monetary costs.
renewableEnergyPercentage	The unit of the incentive SHALL be set to percentage (pct) in this case. The value SHALL be in the range from 0% to 100%.	Percentage of renewable energy of the total obtained energy. This is an ecological incentive that allows optimizing consumption of renewable energy instead of costs.
co2Emission	The unit of the incentive SHALL be set to "kg/Wh". Only positive values SHALL be used.	CO2 emission related to the obtained energy. This is an ecological incentive that allows optimizing the CO2 emission instead of costs.

Table 80: Element "tier. incentiveDescription. incentiveType" value rules

4.3.14.6 incentiveTableConstraintsData

4.3.14.6.1 Description

The incentiveTableConstraints within the incentiveTableConstraintsData function holds additional constraints of the IncentiveTable, regarding how much tiers, boundaries, incetives and slots are allowed.

4.3.14.6.2 Rules

None.

4.3.14.6.3 Element rules

Element	Rule	Description
incentiveTableConstraints (list)		
incentiveTableConstraints. tariff		
incentiveTableConstraints. tariff. tariffId	SHALL be set as PRIMARY IDENTIFIER of a incentiveTable, if more than one incentiveTable is defined within IncentiveTable.	Allows the identification of a incentiveTable. Allows linking of the different functions within the IncentiveTable Feature. Allows linking of the incentiveTable within other Features that are placed in the same entity.
incentiveTableConstraints. tariffConstraints		
incentiveTableConstraints. tariffConstraints. maxTiersPerTariff	If set the incentiveTable SHALL NOT include more tiers than given in maxTiersPerTariff.	Allows the server to define a maximum tier count for this IncentiveTable. This is especially important if a client wants to write an IncentiveTable.
incentiveTableConstraints. tariffConstraints. maxBoundariesPerTier	If set the tier within the incentiveTable SHALL NOT include more boundaries in a tier than given in maxBoundareisPerTier.	Allows the server to define a maximum boundarie count for tiers. This is especially important if a client wants to write an IncentiveTable.
incentiveTableConstraints. tariffConstraints. maxIncentivesPerTier	If set the tier within the incentiveTable SHALL NOT include more incentives than given in maxIncentivesPerTier.	Allows the server to define a maximum incentive count for tiers. This is especially important if a client wants to write an IncentiveTable.
incentiveTableConstraints. incentiveSlotConstraints		
incentiveTableConstraints. incentiveSlotConstraints. slotCountMax	If set the incentiveTable SHALL NOT include more slots than given in slotCountMax.	Allows the server to define a maximum slot count for this IncentiveTable. This is especially important if a client wants to write an IncentiveTable.

Table 81: incentiveTableConstraintsData Element rules

4.3.14.7 Descriptive information and further definitions

None.

4.3.15 LoadControl

4.3.15.1 Dependencies to other Feature Types

4.3.15.1.1 measurementId

LoadControl can use the FOREIGN IDENTIFIER measurementId from the Measurement Feature Type to define a relation to functions of an ElectricalConnection or Measurement Feature.

If a measurementId is given within a LoadControlLimitDescriptionData entry:

- If there is an electricalConnectionPermittedValueSetData entry that relates to the same measurementId, then the value Element of the according loadControlLimitData entry SHOULD only contain values that are defined within the permittedValueSet of the according electricalConnectionPermittedValueSetData entry.
- If there is an electricalConnectionParameterDescriptionData entry that relates to the same measurementId, then the elements of the related electricalConnectionParameterDescriptionData entry SHOULD be considered for the according loadControlLimitData entry.

If there is a measurementDescriptionData entry that relates to the same measurementId, then the unit Element of this measurementDescriptionData entry SHALL be ignored for the according loadControlLimitData entry.

4.3.15.2 Introduction

The LoadControl class has two different approaches to control the load of an appliance:

- Event-based load control with "loadControlEvent" (see section 5.3.14.3) and "loadControlState" (see section 5.3.14.4)
- Limit-based load control with "loadControlLimit" (see section 5.3.14.5) and "loadControlLimitDescription" (see section 5.3.14.7)

The event-based load control is reserved for future use.

The limit-based load control is used where limits (changeable by a client) shall affect the behaviour of a node automatically. E.g. the power demand of an appliance could be limited to a certain value to protect the household of an overload.

Additionally, with the information given in loadControlNode (see section 5.3.14.2), a node can define whether it is (at the moment) remote controllable or not.

4.3.15.3 Rules

None.

3234

3235 ***4.3.15.4 loadControlNodeData***

3236 *4.3.15.4.1 Description*

3237 The loadControlNodeData includes node-wide information related to load control (e.g. whether the
3238 node is remote controllable).

3239

3240 *4.3.15.4.2 Rules*

3241 If the function is absent, a default of "true" SHALL be applied for the element
3242 *isNodeRemoteControllable*.

3243

3244 *4.3.15.4.3 Element rules*

Element	Rule	Description
isNodeRemoteControllable	SHOULD be set. If set to "false", write operations on this feature SHALL be declined. If absent, a default value of "true" SHALL be applied.	Defines whether the node is (at the moment) remote controllable by a client, or not.

3245 *Table 82: loadControlNodeData Element rules*

3246

3247 ***4.3.15.5 loadControlEventListData***

3248 Reserved for future use.

3249

3250 ***4.3.15.6 loadControlStateListData***

3251 Reserved for future use.

3252

3253 ***4.3.15.7 loadControlLimitListData***

3254 *4.3.15.7.1 Description*

3255 The loadControlLimitListData function provides the value of the limit (and an optional time period for
3256 that limit) and further information whether the limit is active or not and if it is changeable by a client
3257 or not.

3258

3259 *4.3.15.7.2 Rules*

3260 None.

3261

3262 *4.3.15.7.3 Element rules*

Element	Rule	Description

limitId	SHALL be set as PRIMARY IDENTIFIER.	Identifier for the limit.
isLimitChangeable	SHOULD be set. If set to "false", the timePeriod, value and isLimitActive element SHALL NOT be writeable by a client. If omitted or set to "true", the timePeriod, value and isLimitActive element SHALL be writeable by a client.	States whether the limit may be changed by a client or not.
isLimitActive	SHOULD be set. If set to "false", the limit and its timePeriod and value element SHALL be ignored. If set to "true" or omitted, the timePeriod and value element SHALL be applied, at least if timePeriod or value are set.	Indicates whether the limit is currently active or not.
timePeriod	MAY be used to define a timePeriod for the value. In this case the value SHALL only be applied within the given timePeriod.	The period where the limit shall be active.
value	If *isLimitActive* is set to "true", the *value* SHALL be set. Otherwise the element MAY be omitted. If *isLimitActive* is set to "false", but *value* is set, the content of *value* SHALL be ignored. If in the related loadControlLimitDescriptionData entry (with the same PRIMARY IDENTIFIER limitId) measurementId is set, the rules of section 4.3.15.1 SHALL be applied.	The actual limit.

Table 83: loadControlLimitListData Element rules

4.3.15.8 loadControlLimitConstraintsListData

4.3.15.8.1 Description

Constraints that must be kept when trying to change a limit, are modelled with the loadControlLimitConstraintsListData function, including a mininum value, a maximum value and the stepsize of the values between.

4.3.15.8.2 Rules

None.

4.3.15.8.3 Element rules

Element	**Rule**	**Description**
limitId	SHALL be set as PRIMARY IDENTIFIER.	Identifier for the limit.
valueRangeMin	SHOULD be set, if not already defined by other functionality (e.g. *electricalConnectionPermittedValueSetListData*). If set, the content of element *value* within *loadControlLimitListData* SHALL NOT contain a value below *valueRangeMin*.	This element allows to define a minimum limit for the element value within *loadControlLimitListData*.
valueRangeMax	SHOULD be set, if not already defined by other functionality (e.g. *electricalConnectionPermittedValueSetListData*).	This element allows to define a maximum limit for the element value within *loadControlLimitListData*.

	If set, the content of element *value* within *loadControlLimitListData* SHALL NOT contain a value above *valueRangeMax*.	
valueStepSize	SHOULD be set, if there is a step size limitation.	The minimum step size between two different limit values.

Table 84: loadControlLimitConstraintsListData Element rules

4.3.15.9 loadControlLimitDescriptionListData

4.3.15.9.1 Description

The rather static information about the available limit(s) are modelled within the loadControlLimitDescriptionListData function.

4.3.15.9.2 Rules

None.

4.3.15.9.3 Element rules

Name	Rule	Description
limitId	SHALL be set as PRIMARY IDENTIFIER.	Identifier for the limit.
limitType	SHALL be set. Value rules of Table 86 SHALL be applied.	Whether the limit shall be used as a minumum or maximum is denoted within this element.
limitCategory	SHOULD be set. Value rules of Table 87 SHALL be applied.	There are different levels of how important the limit is (see enumeration description).
limitDirection	SHOULD be set. If omitted, the limitDirection SHALL be interpreted as “consume”. Default rules of section 3.10.1.14.1 SHALL be applied.	If a device supports both, consumption and production, the direction which will be limited is stated in this element (e.g. if only the consumption of a battery (i.e. it is recharging) will be limited, the enumeration "consume" will be set; if both, consumption and production will be limited, two limits have to be set).
measurementId	SHALL be set as FOREIGN IDENTIFIER, if a measurand or other feature is linked with the measurementId. If set, the rules of section 4.3.15.1 SHALL be applied.	This is a foreign identifier.
unit	SHOULD be set, if there could be ambiguity how to interpret the value without the unit.	The unit, which is used for the limit value, is denoted with this element.
scopeType	None.	A certain meaning of the limit.
label	Default rules of section 3.10.1.2.1 SHALL be applied.	A label for the limit.

description	Default rules of section 3.10.1.3.1 SHALL be applied.	A description for the limit.

Table 85: loadControlLimitDescriptionListData Element rules

Element "**limitType**" value rules:

Value	Rule	Description
minValueLimit	The limit value SHALL be interpreted as a minimum limit. The corresponding value SHALL stay over or equal to the given value of a minValueLimit, if isLimitActive = true and limitType = obligation.	The limit set in the "value" Element of the according "loadControlLimitData" entry shouldn't be underrun (value >= minValueLimit).
maxValueLimit	The limit value SHALL be interpreted as a maximum limit. The corresponding value SHALL stay under or equal to the given value of a maxValueLimit, if isLimitActive = true and limitType = obligation.	The limit set in the "value" Element of the according "loadControlLimitData" entry shouldn't be overrun (value <= maxValueLimit).

Table 86: Element "limitType" value rules

Element "**limitCategory**" value rules:

Value	Rule	Description
obligation	If this category is used, the limit SHALL be kept under any circumstances. A loss of comfort MAY be risked for as long as needed.	Used for limits that are absolutely important to be hold. A loss of comfort could happen and will be accepted. The normal operation will be interrupted for keeping the limit (if needed).
recommendation	If this category is used, the limit SHOULD be kept wherever possible. A loss of comfort MAY be taken into account for a short period of time.	Used for limits that are quite important for the energy management. A loss of comfort could happen and will be accepted for a short period of time. The normal operation should not be interrupted.
optimization	If this category is used, the limit MAY be considered as good as possible. A loss of comfort SHOULD NOT be accepted.	Used for limits that just optimize the energy management. No loss of comfort will be accepted. The normal operation shall not be interrupted.

Table 87: Element "limitCategory" value rules

4.3.15.10 Descriptive information and further definitions

None.

4.3.16 Measurement

4.3.16.1 Dependencies to other Feature Types

- ElectricalConnection
- Threshold

4.3.16.2 Introduction

The SPINE Measurement class provides functionality for transmitting measured, calculated or empirically obtained values from sensors. Mainly these are sensors that provide some kind of information about the environment or physical parameters. Clients can use the information for further actions or just visualize them for the information of a user.

Due to the nature of measurement, the measured values are only readable. Some control systems permit a user to select a preferred value (e.g. a desired room temperature). Such “desired” values obviously require writable fields. This can be achieved with the "Setpoint" class (see section 5.3.21).

4.3.16.3 Rules

None.

4.3.16.4 measurementListData

4.3.16.4.1 Description

The *measurementListData* command is used to communicate measurement values. Typically, a sensor sends this command as notification as soon as it changes more than a specific threshold. Of course, it may be readable, too.

4.3.16.4.2 Rules

None.

4.3.16.4.3 Element rules

Element	Rule	Description
measurementId	SHALL be set as PRIMARY IDENTIFIER.	Enables the identification of different sensors on one SPINE feature.
valueType	SHOULD be set as SUB IDENTIFIER. Value rules of Table 89 SHALL be applied.	It is possible to model different types of measurement values. For example “value”, “averageValue”, “minValue”, etc. (see the SPINE data model for all possible value types). There can be multiple measurementData entries with different valueTypes for the same measurementId. This allows to express different facets of the measurand.
timestamp	SHOULD be set as SUB IDENTIFIER. If set, the timestamp of the creation of the message SHOULD be used.	The time of creation of a measurement value element (i.e. the time it was measured, calculated, ...). Some sensors and communications technologies may already provide this information natively. In other cases the technology

		interface may set timestamp to the time of creation of a measurementData instance. There can be multiple measurementData entries for the same measurementId, but with different timestamps. This allows to express historic values or a measurement timeline.
value	SHALL be set, if a value is available. Otherwise the whole list entry SHALL be omitted or the element *valueState* SHALL be set to "error". If *valueState* is set to "error", but *value* is set, the content of *value* SHALL be ignored by the client. If *valueState* is set to "outOfRange" and *measurementConstraintsListData. valueRangeMax* is set and *value* is equal to or greater than *valueRangeMax*, value SHALL be interpreted as greater than *valueRangeMax*. If *valueState* is set to "outOfRange" and *measurementConstraintsListData. valueRangeMin* is set and *value* is equal to or smaller than *valueRangeMin*, value SHALL be interpreted as lower than *valueRangeMin*. If "value" is set, *measurementDescriptionListData. measurementType* SHALL be set, too.	The measurement value itself, i.e. the magnitude according to “valueType”.
evaluationPeriod	None.	Some *valueType* values can be detailed or even require information about the time period. For example, an *averageValue* is usually computed from measured values of a finite time period. For smart devices, the start and end time of the *evaluationPeriod* can be filled in directly. For bare (“simple”) sensors, such “derived”/”accompanying” values (average, min, max, ...) could be calculated (if desired) by the technology interface component together with the corresponding *evaluationPeriod*.
valueSource	Value rules of Table 90 SHALL be applied.	Whether element “value” is measured, calculated or empirically obtained can be denoted with this element.
valueTendency	Value rules of Table 91 SHALL be applied.	This elements states whether the tendency is *rising*, *stable* or *falling*.
valueState	Value rules of Table 92 SHALL be applied.	A measurand may have a state.

3322 *Table 88: measurementListData Element rules*

Element "**valueType**" value rules:

Value	Rule	Description
value	None.	The normal measurement value.
averageValue	None.	The average measurement value over the last time, stated in the element evaluationPeriod.
minValue	None.	The minimum measurement value of the last time, stated in the element evaluationPeriod.
maxValue	None.	The maximum measurement value of the last time, stated in the element evaluationPeriod.
standardDeviation	None.	Used to quantify the amount of variation or dispersion of a set of values.

Table 89: Element "valueType" value rules

Element "**valueSource**" value rules:

Value	Rule	Description
measuredValue	None.	The measurement value was measured with a dedicated sensor. The value is rather exact.
calculatedValue	None.	The measurement value was calculated with the values from some other parameters (e.g. water temperature, outside temperature, etc.) that have a calculable impact on the measurand. This value is not as exact as a measured value, but it differs not too much from the real value.
empiricalValue	None.	The measurement value is only an empirical one that can differ a lot from the real value. It can rely on some other parameters that have a known value, but there is no exactly calculable relation between them.

Table 90: Element "valueSource" value rules

Element "**valueTendency**" value rules:

Value	Rule	Description
rising	None.	The average measurement value is currently rising.
stable	None.	The average measurement value is currently stable.
falling	None.	The average measurement value is currently falling.

Table 91: Element "valueTendency" value rules

Element "**valueState**" value rules:

Value	Rule	Description
normal	None.	The measurement value could be measured / calculated / empirically determined as expected.
outOfRange	SHALL be used if the value is out of range	The measurement value is currently out of range of the sensor (as defined by “valueRangeMin” and “valueRangeMax”, see section 5.3.15.3.2).
error	SHALL be used if the value has some error.	An error occurred while determining the measurement value.

Table 92: Element "valueState" value rules

4.3.16.5 measurementConstraintsListData

4.3.16.5.1 Description

Some constraints concerning the possible values of a measurand are modelled within this function.

4.3.16.5.2 Rules

None.

4.3.16.5.3 Element rules

Element	Rule	Description
measurementId	SHALL be set as PRIMARY IDENTIFIER.	Enables the identification of different sensors on one SPINE feature.
valueRangeMin	MAY be used to express the minimal value that can be measured.	The minimum possible measurement value measurable by the feature.
valueRangeMax	MAY be used to express the maximum value that can be measured.	The maximum possible measurement value measurable by the feature.
valueStepSize	MAY be used if values can only be delivered in a certain stepsize.	The minimum step size between two different measurement values.

Table 93: measurementConstraintsListData Element rules

4.3.16.6 measurementDescriptionListData

4.3.16.6.1 Description

Descriptive information on a specific measurement type is modelled with a *measurementDescriptionListData* function. This information is rather constant in comparison to measurement values. Typically, it is requested once at the beginning, when a node (an external display, e.g.) is configured to present or use a specific measurement value. As usual, for such scenarios it is recommended the the server notifies changes to all relevant communication partners, because a server cannot expect clients to periodically request this information.

4.3.16.6.2 Rules

None.

4.3.16.6.3 Element rules

Element	Rule	Description
measurementId	SHALL be set as PRIMARY IDENTIFIER.	Enables the identification of different measurands on one SPINE feature.
measurementType	SHALL be set, if a measurement value is available in *measurementListData*.	To identify which type of measurand is measured (for example “temperature”).

commodityType	SHOULD be set if a commodity is measured.	If a measurand of a commodity is measured, the type of commodity is stated here.
unit	SHALL be set, if there could be ambiguity how to interpret the value without the unit.	The unit, which is used for the value. It is always related to the normal value. Other *valueType* values (e.g. *averageDerivativeValue*) can have different units (derived from *unit*).
calibrationValue	None.	Some measurement sensors need a calibration value (e.g. set during production phase). It is automatically added to the internal measured value. I.e. the value stated in the *value* element in the *measurementListData* function is the already corrected value.
scopeType	None.	Specifies a more detailed meaning of the measurand (e.g. outsideTemperature).
label	Default rules of section 3.10.1.2.1 SHALL be applied.	A (short) label can be useful to identify or group specific sensors.
description	Default rules of section 3.10.1.3.1 SHALL be applied.	A (longer) description of a measurement device can be deposited here.

3355 *Table 94: measurementConstraintsListData Element rules*

3356 Element "**measurementType**" value rules:

Value	Rule	Description
acceleration	None.	The acceleration is measured.
angle	None.	An angle is measured.
angularVelocity	None.	The angular velocity is measured.
area	None.	An area is measured.
atmosphericPressure	None.	The atmospheric pressure is measured.
capacity	None.	The capacity is measured.
concentration	None.	A concentration of a substance is measured.
count	None.	A count is measured.
current	None.	The current is measured.
density	None.	The density is measured.
distance	None.	A distance is measured.
electricField	None.	The electric field is measured.
energy	None.	The energy is measured.
force	None.	The force is measured.
frequency	None.	The frequency is measured.
harmonicDistortion	None.	The harmonic distortion is measured.
heat	None.	The heat is measured.
heatFlux	None.	The heat flux is measured.
illuminance	None.	The illuminance is measured.
impulse	None.	An impulse is measured.
level	None.	A level is measured.
magneticField	None.	The magnetic field is measured.
mass	None.	A mass is measured.
massFlow	None.	The mass flow is measured.
particles	None.	The quantity of particles is measured.
percentage	None.	The percentage is measured.

power	None.	The power is measured.
powerFactor	None.	The power factor is measured.
pressure	None.	The pressure is measured.
radonActivity	None.	The radon activity is measured.
relativeHumidity	None.	The relative humidity is measured.
resistance	None.	The resistance is measured.
solarRadiation	None.	The solar radiation is measured.
speed	None.	The speed is measured.
temperature	None.	The temperature is measured.
time	None.	The time is measured.
torque	None.	The torque is measured.
unknown	None.	An unknown measurand is measured.
velocity	None.	The velocity is measured.
voltage	None.	The voltage is measured.
volume	None.	The volume is measured.
volumetricFlow	None.	The volumetric flow is measured.

Table 95: Element "measurementType" value rules

4.3.16.7 measurementThresholdRelationListData

4.3.16.7.1 Description

The relation between a measurand and dedicated thresholds (see section 5.3.26) is modelled with this function.

4.3.16.7.2 Rules

None.

4.3.16.7.3 Element rules

Element	Rule	Description
measurementId	SHALL be set as PRIMARY IDENTIFIER.	Enables the identification of different sensors on one SPINE address.
thresholdId (list)	SHALL be set as FOREIGN IDENTIFIER, if a threshold or other feature is linked with the thresholdId.	Related threshold identifier(s).

Table 96: measurementThresholdRelationListData Element rules

4.3.16.8 Descriptive information and further definitions

None.

4.3.17 Messaging

4.3.17.1 Dependencies to other Feature Types

None.

4.3.17.2 Introduction

Messaging is basically intended for simple textual information transfer for displays and users. The textual information is not intended to be used for any automatisms. It should be used for textual information that are for example event based or otherwise will change over time.

Some devices may keep a list of messages rather than (or in addition to) a single message.

4.3.17.3 Rules

None.

4.3.17.4 messagingListData

4.3.17.4.1 Description

The messagingListData function contains the neccessary elements to model a simple text message.

4.3.17.4.2 Rules

None.

4.3.17.4.3 Element rules

Element	Rule	Description
timestamp	SHOULD be set as SUB IDENTIFIER to the current timestamp at the creation of the message.	The time when the message was generated.
messagingNumber	SHALL be set as PRIMARY IDENTIFIER.	An identifier for one specific message. If a message is marked as obsolete (see below) this number can be used to identify the original message.
type	SHALL be set. Value rules of Table 98 SHALL be applied.	The type of the message.
text	If "type" has the value "obsolete", this element SHOULD be omitted. Otherwise it SHALL be set. The string-length SHOULD NOT be longer than 4096 characters. If it is longer, the sender SHALL consider	The message itself.

	the possibility that the receiver will shorten the string to 4096 characters.	

Table 97: messagingListData Element rules

Element "**type**" value rules:

Value	Rule	Description
logging	Used for messages that should be stored in a log file.	Used for messages that are stored in a log file.
information	Messages that should be presented to the customer on a display; lower priority.	Messages that are presented to the customer on a display; lower priority.
warning	Messages that should be presented to the customer on a display; medium priority.	Messages that are presented to the customer on a display; medium priority.
alarm	Messages that should be presented to the customer on a display; high priority.	Messages that are presented to the customer on a display; high priority.
emergency	Messages that should be presented to the customer on a display; very high priority.	Messages that are presented to the customer on a display; very high priority.
obsolete	Previously sent messages that shall be marked as obsolete and eventually not be displayed anymore.	Used to mark previously sent messages as obsolete (so that they are not displayed anymore).

Table 98: Element "type" value rules

4.3.17.5 Descriptive information and further definitions

None.

4.3.18 NetworkManagement

Reserved for future use.

4.3.19 NodeManagement

All rules and procedures are described in the [ProtocolSpecification].

4.3.20 OperatingConstraints

4.3.20.1 Dependencies to other Feature Types

None.

4.3.20.2 Introduction

This class is designed to model some constraints of an appliance (or functionality) with regards to its energy consumption or production. I.e. this class is NOT intended to control a device. Rather, it enables an appliance to express what needs to be considered especially for the case it permits being

controlled by another device. An example is a central energy management system that uses the class DirectControl to control the appliance; in this case the energy management system needs to be aware of the appliance's operating constraints.

4.3.20.3 Rules

None.

4.3.20.4 operatingConstraintsInterruptListData

4.3.20.4.1 Description

This function models basic constraints for an interrupt of a device's current task.

4.3.20.4.2 Rules

If used, at least one constraint SHALL be set.

4.3.20.4.3 Element rules

Element	Rule	Description
sequenceId	SHALL be set as FOREIGN IDENTIFIER, if the interrupt constraints are related to another feature defining sequenceId (see section 4.3.20.10 for details). Otherwise it SHALL be omitted.	If interrupt constraints are related to a power sequence, the appropriate *sequenceId* is stated in this element.
isPausable	If set to "true", a client MAY pause the related sequence, functionality or task of a server. Otherwise the element SHALL be set to "false" or omitted.	States whether a functionality or task is pausable (true) or not (false).
isStoppable	If set to "true", a client MAY stop the related sequence, functionality or task of a server. Otherwise the element SHALL be set to "false" or omitted.	States whether a functionality or task is stoppable (true) or not (false).
notInterruptibleAtHighPower	If set to "true", a client SHALL NOT	States whether a functionality or task is NOT interruptible while it is consuming

	pause or stop the related sequence, functionality or task of a server, if the power consumption or production is high. Otherwise the element SHALL be set to "false" or omitted.	(or producing) energy (true) or whether it can be interrupted nevertheless (false).
maxCyclesPerDay	If set, the related sequence, functionality or task SHALL NOT restart more often within a day than stated in this element.	If a functionality or task may be started only a maximum number of times per day, this value is denoted here.

Table 99: operatingConstraintsInterruptListData Element rules

4.3.20.5 operatingConstraintsDurationListData

4.3.20.5.1 Description

This function models constraints on the minimum/maximum duration of active ("on") and pause ("off") phases. Example: A compressor can be damaged if it is started and then switched off too early.

4.3.20.5.2 Rules

If used, at least one constraint SHALL be set.

4.3.20.5.3 Element rules

Element	Rule	Description
sequenceId	SHALL be set as FOREIGN IDENTIFIER, if the interrupt constraints are related to another feature defining sequenceId (see section 4.3.20.10 for details). Otherwise it SHALL be omitted.	If duration constraints are related to a power sequence, the appropriate *sequenceId* is stated in this element.
activeDurationMin	If set, the related sequence, functionality or task SHALL be active without interruption for at least the time stated in this element.	This is the minimum duration a device has to run without interruption.
activeDurationMax	If set, the related sequence, functionality or task SHALL at maximum be active without interruption for as long as the time stated int this element.	This is the maximum duration a device can run without interruption.

pauseDurationMin	If set, a pause of the related sequence, functionality or task SHALL be at least as long as the time stated in this element.	This is the minimum duration a device has to pause after the end of an activity.
pauseDurationMax	If set, a pause of the related sequence, functionality or task SHALL be at maximum as long as the time stated in this element.	This is the maximum duration a device can pause after the end of an activity.
activeDurationSumMin	If set, the related sequence, functionality or task SHALL be in sum active for at least the time stated in this element.	This is the minimum duration a device has to run in total for a given task (summation of all active times).
activeDurationSumMax	If set, the related sequence, functionality or task SHALL be in sum active for maximum the time stated in this element.	This is the maximum duration a device can run in total for a given task (summation of all active times).

Table 100: operatingConstraintsDurationListData Element rules

4.3.20.6 operatingConstraintsPowerDescriptionListData

4.3.20.6.1 Description

This function models basic information relevant for *operatingConstraintsPowerRangeListData* and *operatingConstraintsPowerLevelListData*.

4.3.20.6.2 Rules

None.

4.3.20.6.3 Element rules

Element	Rule	Description
sequenceId	SHALL be set as FOREIGN IDENTIFIER, if the interrupt constraints are related to another feature defining sequenceId (see section 4.3.20.10 for details). Otherwise it SHALL be omitted.	If function elements are related to a power sequence, the appropriate *sequenceId* is stated in this element.
positiveEnergyDirection	SHALL be set, if there could be ambiguity how to interpret the value of the related sequence, functionality or task. For further value rules see section 3.10.1.14.1.	The *positiveEnergyDirection* states whether energy consumption or production is counted as positive value.
powerUnit	SHALL be set, if there could be ambiguity how to	The unit for all power values in operatingConstraintsPowerRangeListDa

	interpret the power value of the related sequence, functionality or task without the unit.	ta and operatingConstraintsPowerLevelListData.
energyUnit	SHALL be set, if there could be ambiguity how to interpret the energy value of the related sequence, functionality or task without the unit.	The unit for all energy values in operatingConstraintsPowerRangeListData and operatingConstraintsPowerLevelListData.
description	Default rules of section 3.10.1.3.1 SHALL be applied.	Descriptive information.

Table 101: operatingConstraintsPowerDescriptionListData Element rules

4.3.20.7 operatingConstraintsPowerRangeListData

4.3.20.7.1 Description

This function can be used to model a power/energy range(!) a device can operate. An alternative model for power/energy constraints is operatingConstraintsPowerLevelListData.

4.3.20.7.2 Rules

If used, at least one range SHALL be set.

4.3.20.7.3 Element rules

Element	Rule	Description
sequenceId	SHALL be set as FOREIGN IDENTIFIER, if the interrupt constraints are related to another feature defining sequenceId (see section 4.3.20.10 for details). Otherwise it SHALL be omitted.	If the power range is related to a power sequence, the appropriate *sequenceId* is stated in this element.
powerMin	If set, the related sequence, functionality or task SHALL at least consume or produce the given value of powerMin.	The minimum power a device can consume/produce.
powerMax	If set, the related sequence, functionality or task SHALL at maximum consume or produce the given value of powerMax.	The maximum power a device can consume/produce.
energyMin	If set, the related sequence, functionality or task SHALL at least consume or produce the given value of energyMin.	The minimum energy a device can consume/produce.
energyMax	If set, the related sequence, functionality or task SHALL at maximum consume or produce the given value of energyMax.	The maximum energy a device can consume/produce.

Table 102: operatingConstraintsPowerRangeListData Element rules

4.3.20.8 operatingConstraintsPowerLevelListData

4.3.20.8.1 Description

This function can be used to model a list of possible power values a device can operate. An alternative model for power constraints is operatingConstraintsPowerRangeListData, where a full range (not only discrete levels) can be modelled.

4.3.20.8.2 Rules

If used, at least one power value SHALL be set.

4.3.20.8.3 Element rules

Element	Rule	Description
sequenceId	SHALL be set as FOREIGN IDENTIFIER, if the interrupt constraints are related to another feature defining sequenceId (see section 4.3.20.10 for details). Otherwise it SHALL be omitted.	If the power levels are related to a power sequence, the appropriate *sequenceId* is stated in this element.
power (list)	The values stated within the list SHALL define the allowed value set that MAY be used within the value of a related sequence, functionality or task.	Each “power” item contains a possible power level.

Table 103: operatingConstraintsPowerLevelListData Element rules

4.3.20.9 operatingConstraintsResumeImplicationListData

4.3.20.9.1 Description

Pausing a device might be permitted according to above mentioned (duration) constraints. Nevertheless, it can be expected that it has implications if a device's task is paused temporarily and then resumed to continue the task. One possible implication is an additional mechanical stress which reduces the device’s lifetime and could be interpreted as additional cost. Such additional costs or additional energy consumption can be modelled with this function in order to be considered for energy management. However, these are typically empirical (thus rather constant) data, therefore should not be considered to be precise.

4.3.20.9.2 Rules

None.

4.3.20.9.3 Element rules

Element	Rule	Description
sequenceId	SHALL be set as FOREIGN IDENTIFIER, if the interrupt constraints are related to another feature defining sequenceId (see section 4.3.20.10 for details). Otherwise it SHALL be omitted.	If the resume implications are related to a power sequence, the appropriate *sequenceId* is stated in this element.
resumeEnergyEstimated	None.	Additional energy a device consumes before resuming to its normal operation (after a pause).
energyUnit	SHOULD be set, if there could be ambiguity how to interpret the value without the unit.	Unit for the energy stated in the element resumeEnergyEstimated.
resumeCostEstimated	None.	Additional costs for the resumption of a device to its normal operation.
currency	SHOULD be set, if "resumeCostEstimate" is set, too.	Currency for the price stated in the element resumeCostEstimated.

Table 104: operatingConstraintsResumeImplicationListData Element rules

4.3.20.10 Descriptive information and further definitions

If operatingConstraintsPowerRangeListData or operatingConstraintsPowerLevelListData is supported, operatingConstraintsPowerDescriptionListData SHALL be supported, too.

Section 5.3.18.1 mentions some recommendations to avoid ambiguities. Therefore, some rules are defined:

1. In every moment for each supported function “xListData” (with “x” as “prefix” placeholder for functions mentioned in section 4.3.20) exactly one of the following conditions SHALL be valid:
   1.1. “no sequenceId”: xListData contains exactly one xData AND in xData the element “sequenceId” is NOT present.
   1.2. “all sequenceId”: xListData contains one or more xData AND in every xData the element “sequenceId” is present. No two xData items have the same value in “sequenceId”.
   1.3. “currently no sequence”: xListData has no xData. This shall be treated as special case of item 1.2: In this case there is currently no related sequenceId.

   Please note that these rules DO NOT permit to have an xListData where at least one xData has “sequenceId” set AND at least one xData has no “sequenceId” set.

2. Further rules on the use of elements apply:
   2.1. If an xData item has “sequenceId” set the other elements of xData are related to the corresponding power sequence except for the following case:

2.2. The use of element “maxCyclesPerDay” of function “operatingConstraintsInterruptData” is optional. If the element is used it is NOT related to any power sequence even if the element “sequenceId” is present.
Remark: This means this specific constraint always applies to the feature and its related functionality in general.

3. Use of parallel instances of this Feature Type in the same entity for the same functionality: It is valid to have in the same entity one Feature Type instance that follows the “no sequenceId” rule (item 1.1 above) AND one Feature Type instance that follows the “all sequenceId” / “currently no sequence” rules (items 1.2 and 1.3 above) for the same functionality. However, in this case the xData child elements of these two instances SHALL be disjoint (i.e. they SHALL NOT contradict each other).
4. Remark: Esp. in case of the “no sequenceId” rule it is recommended to use the feature group concept (see section 2.1.4) in order to clearly relate the operating constraints to the actual/essential functionality.

4.3.21 PowerSequences

Reserved for future use.

4.3.22 Sensing

4.3.22.1 Dependencies to other Feature Types

None.

4.3.22.2 Introduction

The *Sensing* class is used for modelling data obtained by sensors in the meaning of more state-based sensors like contact sensors, switches and buttons, etc., in contrast to value-based sensors like temperature or humidity sensors that are treated with the *Measurement* class.

4.3.22.3 Rules

None.

4.3.22.4 sensingListData

4.3.22.4.1 Description

The list function of sensing can be used to model sensor information over time. This may be of interest for a motion detector (when someone was at the backdoor), a contact sensor (when was the window open in the last 24 hours), etc.

Some sensors may only store the latest value and therefore only support one entry in the list. To enable users and applications to query past changes, a communications technology interface may

cache states/values of simple sensors that only support the single data function. That is no mandatory feature, however.

4.3.22.4.2 Rules

If used, at least one constraint SHALL be set.

4.3.22.4.3 Element rules

Element	Rule	Description
timestamp	SHOULD be set as PRIMARY IDENTIFIER to the timestamp at the creation of the message.	The creation time of the *sensingData* instance.
state	SHALL be set. Value rules of Table 106 SHALL be applied.	The state of the sensor is denoted in this element.
value	Whether this element SHALL be set or SHOULD NOT be set depends on the state (see Table 106 for details). If no information is given, as a default, "value" SHOULD NOT be set.	Some sensors deliver a value (e.g. a *levelSwitch*).

Table 105: sensingListData Element rules

Element "**state**" value rules:

Value	Rule	Description
on	"value" SHOULD NOT be set.	"on" was detected, meaning that the according functionality was switched on.
off	"value" SHOULD NOT be set.	"off" was detected, meaning that the according functionality was switched off.
toggle	"value" SHOULD NOT be set.	"toggle" was detected, meaning that the according functionality was toggled (e.g. from "on" to "off" or vice versa).
level	"value" SHALL be set.	"level" was detected, meaning that the according functionality changed its level (e.g. from 2.4 to 8.2) that can be found in the Element "value".
levelUp	"value" SHALL be set.	"levelUp" was detected, meaning that the according functionality changed its level upwards.
levelDown	"value" SHALL be set.	"levelDown" was detected, meaning that the according functionality changed its level downwards.
levelStart	"value" SHOULD NOT be set.	"levelStart" was detected, meaning that the according functionality started to change its level.
levelStop	"value" SHOULD NOT be set.	"levelStop" was detected, meaning that the according functionality stopped to change its level.
levelAbsolute	"value" SHALL be set.	"levelAbsolute" was detected, meaning that the according functionality changed its level to an absolute value stated in the Element "value" (e.g. 1190).
levelRelative	"value" SHALL be set.	"levelRelative" was detected, meaning that the according functionality moved its level in comparison to the old one. The relative change can be found in the Element "value" (e.g. a change

		from 22 to 33 would result in a content of the Element "value" of 11).
levelPercentageAbsolute	"value" SHALL be set.	"levelPercentageAbsolute" was detected, meaning that the according functionality changed its level to an absolute percentage value stated in the Element "value" (e.g. 90%).
levelPercentageRelative	"value" SHALL be set.	"levelPercentageRelative" was detected, meaning that the according functionality changed its percentage level relative to the old one. The relative changed can be found in the Element "value" (e.g. a change from 40% to 50% would result in a content of the Element "value" of 10).
pressed	"value" SHOULD NOT be set.	"pressed" was detected, meaning that the according functionality was pressed (e.g. a button, that was pressed).
longPressed	"value" SHOULD NOT be set.	"longPressed" was detected, meaning that the according functionality was pressed longer (e.g. a button, that was pressed longer). The timing difference between "pressed" and "longPressed" is up to the vendor.
released	"value" SHOULD NOT be set.	"released" was detected, meaning that the according functionality was released (e.g. a button, that was released).
changed	"value" SHOULD NOT be set.	"changed" was detected, meaning that the according functionality changed.
started	"value" SHOULD NOT be set.	"started" was detected, meaning that the according functionality started.
stopped	"value" SHOULD NOT be set.	"stopped" was detected, meaning that the according functionality stopped.
paused	"value" SHOULD NOT be set.	"paused" was detected, meaning that the according functionality paused.
middle	"value" SHOULD NOT be set.	"middle" was detected, meaning that the according functionality (e.g. a scroll bar or a throttle) is in position "middle".
up	"value" SHOULD NOT be set.	"up" was detected, meaning that the according functionality moves up.
down	"value" SHOULD NOT be set.	"down" was detected, meaning that the according functionality moves down.
forward	"value" SHOULD NOT be set.	"forward" was detected, meaning that the according functionality moves forward.
backwards	"value" SHOULD NOT be set.	"backwards" was detected, meaning that the according functionality moves backwards.
open	"value" SHOULD NOT be set.	"open" was detected, meaning that the according functionality is open (e.g. a door).
closed	"value" SHOULD NOT be set.	"closed" was detected, meaning that the according functionality is closed (e.g. a door).
opening	"value" SHOULD NOT be set.	"opening" was detected, meaning that the according functionality is opening (e.g. a door, that is changing from "closed" to "open").
closing	"value" SHOULD NOT be set.	"closing" was detected, meaning that the according functionality is closing (e.g. a door, that is changing from "open" to "closed").

high	"value" SHOULD NOT be set.	"high" was detected, meaning that the according functionality is at a high position or value.
low	"value" SHOULD NOT be set.	"low" was detected, meaning that the according functionality is at a low position or value.
day	"value" SHOULD NOT be set.	"day" was detected, meaning that the according functionality is in day-mode.
night	"value" SHOULD NOT be set.	"night" was detected, meaning that the according functionality is in night-mode.
detected	"value" SHOULD NOT be set.	"detected" was detected, meaning that the according functionality was detected (e.g. a motion sensor detected a motion).
notDetected	"value" SHOULD NOT be set.	"notDetected" was detected, meaning that the according functionality was detected (e.g. a motion sensor reports that no motion is detected).
alarmed	"value" SHOULD NOT be set.	"alarmed" was detected, meaning that the according functionality is in alarm mode (e.g. a smoke detector).
notAlarmed	"value" SHOULD NOT be set.	"notAlarmed" was detected, meaning that the according functionality is not in alarm mode (e.g. a smoke detector).

Table 106: Element "state" value rules

4.3.22.5 sensingDescriptionData

4.3.22.5.1 Description

The rather static (non-changing) elements of the *Sensing* class are combined in the *sensingDescriptionData* function. Typically, it is only requested once by a user or an application.

4.3.22.5.2 Rules

None.

4.3.22.5.3 Element rules

Element	**Rule**	**Description**
sensingType	SHALL be set. Value rules of Table 108 SHALL be applied.	Which type of sensor is modelled is denoted with this element.
unit	SHALL be set, if there could be ambiguity how to interpret the sensing value without the unit.	The unit in which the values of the sensor are given.
scopeType	None.	Specifies a more detailed meaning of the sensor (e.g. shutter).
label	Default rules of section 3.10.1.2.1 SHALL be applied.	A short label of the sensor.
description	Default rules of section 3.10.1.3.1 SHALL be applied.	Descriptive information on the sensor.

Table 107: sensingDescriptionData Element rules

Element "**sensingType**" value rules:

Value	Rule	Description
switch	None.	A switch that has usually two states "on" and "off".
button	None.	A button that can be "pressed" (and eventually supports "longPressed" and "released")
level	None.	Some level functionality supporting one or more of the level-related sensing states.
levelSwitch	None.	Some level functionality supporting one or more of the level-related sensing states and a switch functionality ("on" / "off").
windowHandle	None.	A window handle supporting different positions.
contactSensor	None.	Sensor that can detect whether something is "open" or "closed".
occupancySensor	None.	Sensor that can detect whether someone is present in a room ("detected") or not ("notDetected").
motionDetector	None.	Sensor that can detect motion ("detected") and eventually supports the "notDetected" state, too.
fireDetector	None.	A sensor that detects fire ("detected") and eventually the absence of fire ("notDetected").
smokeDetector	None.	A sensor that detects smoke ("detected") and eventually the absence of smoke ("notDetected").
heatDetector	None.	A sensor that detects heat ("detected") and eventually the absence of heat ("notDetected").
waterDetector	None.	A sensor that detects water ("detected") and eventually the absence of water ("notDetected").
gasDetector	None.	A sensor that detects gas ("detected") and eventually the absence of gas ("notDetected").
alarmSensor	None.	A sensor providing alarm information ("alarmed" / "notAlarmed").
powerAlarmSensor	None.	A sensor reporting power alarms ("alarmed" / "notAlarmed").
dayNightIndicator	None.	A sensor detecting whether it is "day" or "night".

Table 108: Element "sensingType" value rules

4.3.22.6 Descriptive information and further definitions

None.

4.3.23 Setpoint

4.3.23.1 Dependencies to other Feature Types

- Measurement
- TimeTable

4.3.23.2 Introduction

Setting setpoint values is an often chosen way to control some functionality of a device. Especially in the home automation and HVAC domain, several use cases depend on setpoints. This class handles

3583 the setpoint itself. Relations to Measurement and TimeTable class enables the control of a
3584 measurand for specific time frames.

3585

3586 ***4.3.23.3 Rules***
3587 None.

3588

3589 ***4.3.23.4 setpointListData***

3590 *4.3.23.4.1 Description*
3591 The setpoint values and (if needed) the identifier of related measurands or time tables are modelled
3592 within this function. It is possible to either define an exact value, that should be reached or a range
3593 which shall not be fallen below or exceeded. Absolute or relative tolerances can be defined, too.

3594

3595 *4.3.23.4.2 Rules*
3596 One of the following SHALL be supported, at least (more of the combinations MAY be set):

3597 - value
3598 - valueMin and valueMax

3599

3600 *4.3.23.4.3 Element rules*

Element	Rule	Description
setpointId	SHALL be set as PRIMARY IDENTIFIER.	Identifier of the setpoint.
value	None.	The setpoint value itself.
valueMin	If set, the valueMin SHALL be interpreted as acceptable minimum setpoint value.	Instead of a fixed value (see above) a value range can be set. This would be the lower limit.
valueMax	If set, the valueMax SHALL be interpreted as acceptable maximum setpoint value.	Instead of a fixed value (see above) a value range can be set. This would be the upper limit.
valueToleranceAbsolute	If set, the valueToleranceAbsolute SHALL be added to the value to calculate the tolerated maximum and substracted from the value to calculate the tolerated minimum. The valueToleranceAbsolute SHALL be positive.	If the value may vary some absolute value around the desired setpoint, the maximum variation allowed (in both directions) can be stated here.
valueTolerancePercentage	If set, the valueTolerancePercentage	If the value may vary some percentage around the desired setpoint, the

	SHALL be multiplied with the value and than added to the value to calculate the tolerated maximum and the valueTolerancePercentage SHALL be multiplied with the value and substracted from the value to calculate the tolerated minimum. The valueTolerancePercentage SHALL be positive or zero.	maximum variation allowed (in both directions) can be stated here.

Table 109: setpointListData Element rules

4.3.23.5 setpointConstraintsListData

4.3.23.5.1 Description

The constraints regarding the possible setpoints allowed are modelled within this function.

4.3.23.5.2 Rules

If used, at least one constraint SHALL be set.

4.3.23.5.3 Element rules

Element	Rule	Description
setpointId	SHALL be set as PRIMARY IDENTIFIER.	Identifier of the setpoint, these constraints belong to.
setpointRangeMin	The setpoint value SHALL NOT be smaller than setpointRangeMin.	The minimum value, the setpoint can be set to.
setpointRangeMax	The setpoint value SHALL NOT be larger than setpointRangeMax.	The maximum value, the setpoint can be set to.
setpointStepSize	SHOULD be used if values are only supported in a certain stepsize. Values that do not match the step size SHALL be rounded by the server.	The minimum step size between two different values.

Table 110: setpointConstraintsListData Element rules

4.3.23.6 setpointDescriptionListData

4.3.23.6.1 Description

The rather static (non-changing) elements of the *Setpoint* class are combined in the *setpointDescriptionListData* function. Typically, it is only requested once by a user or an application.

3618 *4.3.23.6.2 Rules*

3619 None.

3620

3621 *4.3.23.6.3 Element rules*

Element	Rule	Description
setpointId	SHALL be set as PRIMARY IDENTIFIER.	Identifier of the setpoint.
measurementId	SHALL be set as FOREIGN IDENTIIER, if the setpoint is linked to a measurement or another Feature that uses the same measurementId as FOREIGN IDENTIFIER. Otherwise it SHALL be omitted.	If the setpoint relates to a measurand, the corresponding ID can be stated here.
timeTableId	SHALL be set as FOREIGN IDENTIIER to the according timeTableId, if the setpoint is linked to a timeTable or another Feature that uses the same timeTableId as FOREIGN IDENTIFIER. Otherwise it SHALL be omitted.	If the activation of the setpoint is time dependent, the ID of a corresponding time table can denoted in this element.
setpointType	SHOULD be set. If omitted, the setpoint SHALL be interpreted as *setpointType* "valueAbsolute". Value rules of Table 112 SHALL be applied.	This is the type of setpoint used for this specific setpoint.
unit	SHALL be set, if there could be ambiguity how to interpret the value without the unit.	The unit used for this setpoint (see SPINE data model for all possible values).
scopeType	None.	Specifies a more detailed meaning of the setpoint.
label	Default rules of section 3.10.1.2.1 SHALL be applied.	A short label of the setpoint.
description	Default rules of section 3.10.1.3.1 SHALL be applied.	Descriptive information on the setpoint.

3622 *Table 111: setpointDescriptionListData Element rules*

3623 Element "**setpointType**" value rules:

Value	Rule	Description
valueAbsolute	None.	An absolute value, e.g. 28°C. A unit is needed in most cases.
valueRelative	None.	A relative value in percentage. The unit is "pct" in most cases.

3624 *Table 112: Element "setpointType" value rules*

3625

3626 ***4.3.23.7 Descriptive information and further definitions***

3627 None.

3628

4.3.24 SmartEnergyManagementPs

4.3.24.1 Functions and message purposes

The following table shows which messages purposes are required or optional (see column "M/O") for a server. It also shows which function is used for the given purpose and which direction and classifier applies:

Message purpose	Used complex function	M / O	Direction	Used classifiers	Explanation
powerSequences information	smartEnergyManagementPsData	M	Out	reply, notify	Informs or updates a client about a forecast (fixed or with flexibilities).
powerSequences information request	smartEnergyManagementPsData	M	In	read	Requests current information from a server.
powerSequences configuration request call	smartEnergyManagementPsConfigurationRequestCall	O	Out	call	Informs the client that the server's power sequences need to be (re-)scheduled.
powerSequences configuration	smartEnergyManagementPsData	M	In	write (restricted)	Is used to submit a new configuration to a server.
powerSequences state change	smartEnergyManagementPsData	O	In	write (restricted)	Enables a power sequences client to change the current state of a power sequences server's sequence (if permitted by the server).
powerSequences price calculation request call	smartEnergyManagementPsPriceCalculationRequestCall	O	Out	call	Requests a price calculation from a client.
powerSequences price information	smartEnergyManagementPsPriceData	O	Out	reply, notify	Informs or updates a client about a price.
powerSequences price information request	smartEnergyManagementPsPriceData	O	In	read	Requests price information stored on the server.
powerSequences price change	smartEnergyManagementPsPriceData	O	In	write (restricted)	Enables a power sequences client to change the price

					information of a power sequences server's sequence (if permitted by the server).

Table 113: Server messages overview. The "write (restricted)" denotes if the message is intended for partial modification.

Details on the messages are defined in section 4.3.24.4.

4.3.24.2 Introduction

4.3.24.2.1 Scope of this Feature Type

The underlying SmartEnergyManagementPs complex class enables a client to shift the demand of a server within some given constraints. The corresponding rules for interoperable implementations are denoted within this feature type.

4.3.24.2.2 Theory of operation (informative)

Each power sequences server requires the establishment of a binding with a power sequences client (see 4.3.24.3.8) as only a bound power sequences client is permitted to configure a power sequences server. One power sequences client may be bound to many power sequences servers (see 4.3.24.3.8), while one power sequences server is only bound to exactly one power sequences client (see 4.3.24.3.9). This completes the basic setup for a power sequences server and client.

Please note: Binding does not imply subscription mechanisms! If a client is interested in data changes on the server (which is recommended), a subscription has to be established, too.

If the power sequences contain flexibilities and/or alternatives, the power sequences server issues a power sequences configuration request call. Once the power sequences client receives this message, it evaluates whether it will configure the according power sequences server or ignore the call (temporarily or permanent). The power sequences client configures the server by writing a power sequences configuration to the power sequence server. If the power sequences client does not configure the power sequences server, the power sequences server follows the configuration it has initially exposed, i.e. its own preference. If the power sequences server receives a configuration it should take into account, it first checks whether the configuration is valid. If the received configuration is valid, i.e. meets the constraints the power sequences server has laid out, it takes over the configuration.

During runtime, the power sequences client and server may re-negotiate the configuration multiple times, even if the according power sequence is already running (if the power sequences server supports it). However, every time the data on either the power sequence information, configuration or state changes, the power sequences server updates the subscribed nodes.

Please note: Devices that can't freely choose between different modes of operations (e.g. because they cannot interrupt an already running process) would not publish the power sequences which they cannot switch to "currently" until the situation has been resolved.

Dependent on the use case, a power sequence needs to contain different kind of data. This means there are different types of sequences possible. The description of the supported sequence types and how they are recognised (evaluated) is given in detail in section 4.3.24.3.5.5.

In the subsequent sections terms and rules are defined for the use of the complex class within this Feature Type. These rules impose restrictions on the use of the class as follows:

- A sequence contains a minimum of one slot, but may contain as many slots as needed.
- Slots which appear in a certain order within a power sequence cannot be interchanged, i.e. a slot can never be shifted so that it is executed before a formerly subsequent slot. The slot order within a power sequence cannot be changed. Thus, each time shift to a slot moves the subsequent slots accordingly.
- Within a sequence, each pause or gap is treated as regular slot of a power sequence and thus holds the same data a regular slot holds (but with an energy value of “0”).
- For fixed forecasts, no constraints are specified, because no shifting is permitted (see 4.3.24.3.5.5.2).
- etc.

4.3.24.3 Common specification details

4.3.24.3.1 Guidance to the concept and description

The concept permits numerous possibilities for the representation of a device’s energy management related data. In order to achieve interoperability, a formal definition of terms and rules is given subsequently. First of all, definitions and rules related to “slots” are given, followed by definitions and rules on “sequences”.

However, one key to achieve interoperability is to focus on those “variants” of device’s energy management information that are supported by this specification. These are called “permitted sequence types” and explained in section 4.3.24.3.5.5. It is recommended to start with this section for a first reading at least. These sequence types definitions make use of the terms that are explained in the previous sections.

Finally, rules on the evaluation of power sequences information and some necessary conventions are given.

4.3.24.3.2 Slots

4.3.24.3.2.1 Overview

Section 4.3.24.3.2 contains a number of basic terms and rules on the modelling of power sequence slots. In order to get a better overview and understanding of this section and the concepts it may be preferable to continue first with the following sections and then return to the other sections on demand:

1. Section 4.3.24.3.2.9 discusses some general duration facets of slots.
2. Section 4.3.24.3.2.7 discusses slots with configurable schedule.
3. Section 4.3.24.3.2.5 describes some requirements on power/energy values.

4. Section 4.3.24.3.2.8 describes an impact of power/energy consumption/generation through the modification of a slot's duration.

4.3.24.3.2.2 Determined vs. undetermined slot

4.3.24.3.2.2.1 Introduction (informative)

A "determined slot" can be used to build detailed power/energy curves. This provides a basis for time-dependent forecasts, process configuration (in advance) and autonomous process execution. An "undetermined slot" on the other hand can be used for processes where only few information is available in advance.

4.3.24.3.2.2.2 Definitions

A slot is called "determined" if all of the following information is present:

1. "alternatives. powerSequence. powerTimeSlot. schedule. slotNumber"
2. "alternatives. powerSequence. powerTimeSlot. schedule. defaultDuration"
3. The slot is (explicitly or implicitly) activated or deactivated (see 4.3.24.3.2.4).
4. The slot value forecast is determined (see 4.3.24.3.2.5).

The value of "alternatives. powerSequence. powerTimeSlot. schedule. defaultDuration" SHALL be greater than 0.

Remark: The element "alternatives. powerSequence. powerTimeSlot. schedule. durationUncertainty" may as well be present in a determined slot, like other elements as well. However, this is relevant for the duration estimation only and not for the classification as determined slot.

A slot is called "undetermined" if it is NOT "determined" AND all of the following conditions are fulfilled:

1. "alternatives. powerSequence. powerTimeSlot. schedule. slotNumber" is present.
2. "alternatives. powerSequence. powerTimeSlot. schedule. defaultDuration" is NOT present.
3. The slot value forecast is determined (see 4.3.24.3.2.5).

Undetermined slots have no activation or deactivation information themselves. Rather, this kind of slot is designed for a specific purpose where the final execution is determined by its power sequence's state.

A slot that is neither determined nor undetermined is malformed and not allowed in this specification.

The element "alternatives. powerSequence. powerTimeSlot. schedule. timePeriod" SHALL NOT be used to make a slot schedule "determined". Rather, "timePeriod" can be used for supplementary purposes (e.g. display) only. If used, it SHALL NOT contradict regular schedule information (i.e. sequence start time, sum of preceding (and explicitly/implicitly activated) slot durations).

4.3.24.3.2.2.3 Process rules

1. On numbering slot numbers in a power sequence:
   Each slot of a power sequence SHALL be assigned a slot number within the element "slotNumber". The slot numbers within the sequence SHALL be ascending according to the chronological order of the slots. The combination of sequence ID and slot number and (optionally) the current repetition number (see sections 4.3.24.3.2.10 and 4.3.24.3.5.3.2) SHALL remain unique at any given point in time.
2. Some more process rules on the use of defaultDuration and its dependencies to some other values are described in section 4.3.24.3.2.7.3.

4.3.24.3.2.3 Duration uncertainty

This section (4.3.24.3.2.3 and subsections) applies for determined slots only!

4.3.24.3.2.3.1 Introduction (informative)

Slots exposed by a power sequences server typically have fixed durations. In some applications, the exact duration of a slot cannot be determined. Thus, a slot duration might be rather an average value, supplemented with an uncertainty of the duration.

Note: A duration uncertainty does NOT model a configuration range of a duration. A duration uncertainty also does NOT express any potential interrupt of the slot. A duration uncertainty only models that a power sequences server cannot estimate a duration more precisely.

4.3.24.3.2.3.2 Definitions

The duration of a determined slot SHALL be rather "exact" if NO duration uncertainty is given.

A duration uncertainty is given if the following information is present:

1. The slot schedule is determined.
2. "alternatives. powerSequence. powerTimeSlot. schedule. durationUncertainty" is present.

If "alternatives. powerSequence. powerTimeSlot. schedule. durationUncertainty" is present the value SHALL be positive.

If a duration uncertainty is given, the slot's final duration (i.e. the measured duration when the slot was executed completely) SHOULD be in the range from "lower duration bound" to (defaultDuration + durationUncertainty). The value of "lower duration bound" is determined by (defaultDuration - durationUncertainty) provided that (defaultDuration - durationUncertainty) is not negative. Otherwise, "lower duration bound" is 0.

If a slot provides information about a duration uncertainty and has a configurable duration (see 4.3.24.3.2.7), the duration uncertainty is considered to be constant for all configurable durations of the slot.

4.3.24.3.2.4 Activated vs. deactivated slot

This section (4.3.24.3.2.4) applies for determined slots only.

Usually, a slot is explicitly or implicitly activated, hence contributes with its data to the schedule and execution of its power sequence. However, in case a slot is “optional” it can instead be marked as “deactivated”. A deactivated slot does not contribute to the schedule and execution of a power sequence. I.e. a power sequence is scheduled and executed as if all deactivated slots have been removed from the sequence.

4.3.24.3.2.5 Determined vs. undetermined slot value forecast

A slot can be assigned power or energy values to model a potential or expected forecast of a power sequences server’s power or energy consumption or production.

A slot value forecast is called “determined” if all of the following conditions are fulfilled:

1. At least one instance of “alternatives. powerSequence. powerTimeSlot. valueList. value” is present for the given slot AND the child element “valueType” is present AND the child element “value” is present.
2. The list of the slot’s “alternatives. powerSequence. powerTimeSlot. valueList. value” instances fulfils the value types requirement (see subsequent definition).

Otherwise the slot value forecast is called “undetermined”.

The value types requirement is defined as follows: Within each slot of a power sequence, different pairs of the elements “value” and “valueType” may be provided. The availability of these values and more important the value types depends on the process(es) the power sequences server exposes. However, a power sequences server SHALL at least provide one of the values and value types or tuples of values and value types contained in the following list:

- power
- powerMin
- powerMax
- energy
- energyMin
- energyMax
- powerExpectedValue
- powerExpectedValue + powerStandardDeviation
- powerExpectedValue + powerStandardDeviation + powerSkewness
- energyExpectedValue
- energyExpectedValue + energyStandardDeviation
- energyExpectedValue + energyStandardDeviation + energySkewness

As an example, a power sequences server might expose all slots with a valueType of “power”. A power sequences server might as well expose all slots with the valueTypes “powerExpectedValue”, “powerStandardDeviation” and “powerSkewness”. However, it is invalid to only provide the valueType combination of “powerExpectedValue” and “powerSkewness” without “powerStandardDeviation” for one or more slots.

Please note that “expectedValue” refers to the term “expected value” (also “expectation”) of probability theory. It may be used to express the estimation of an average value.

Note: See section 4.3.24.3.2.8 on a dependency between power/energy forecast and configurable slot durations!

4.3.24.3.2.6 Constrained vs. implicit slot schedule

This section (4.3.24.3.2.6 and subsections) applies for determined slots only!

4.3.24.3.2.6.1 Introduction (informative)

This section is closely related to the question whether a slot schedule can be configured or not (see 4.3.24.3.2.7). In case of configurable slot schedules there is a need to model the constraints of such a configuration. Non-configurable slots, on the other hand, do not require such constraints, hence their schedule is given implicitly.

4.3.24.3.2.6.2 Definitions

A slot schedule is called “constrained” if all of the following conditions are fulfilled:

1. The slot is determined.
2. “alternatives. powerSequence. powerTimeSlot. scheduleConstraints” is present with at least one child element.

A slot schedule is called “implicit” if all of the following conditions are fulfilled:

1. The slot is determined.
2. “alternatives. powerSequence. powerTimeSlot. scheduleConstraints” is absent or has no child element.

An implicit slot schedule does NOT permit any direct variation of the slot’s schedule itself. I.e. it is not possible to modify the slot’s duration or start time or end time directly. This means the duration of the slot is not modifiable by a value assignment, and the slot’s start time is determined implicitly (by the end of the preceding slot or by the sequence’s start time if there is no preceding slot). It also means the slot is not optional, i.e. it cannot be deactivated individually (see 4.3.24.3.2.4).

A constrained slot schedule denotes further rules that need to be considered for the configuration of the slot’s schedule. Among others, this may include the possibility for the complete deactivation of the slot (see 4.3.24.3.2.4).

4.3.24.3.2.7 Configurable vs. non-configurable schedule of a determined slot

This section (4.3.24.3.2.7 and subsections) applies for determined slots only!

4.3.24.3.2.7.1 Introduction (informative)

Some devices can permit the configuration of some slot durations or even the disabling of slots of a given power sequence. Such slot schedules are called “configurable”, whereas other slot schedules are called “non-configurable”. The following definition explains the proper conditions. Furthermore, process rules that affect both kinds of slot schedules are discussed.

4.3.24.3.2.7.2 Definitions

The schedule information of a determined slot is configurable if all of the following conditions are fulfilled:

1. The slot schedule is constrained (see 4.3.24.3.2.6).
2. The slot’s sequence permits configuration.

In this case the slot’s “alternatives. powerSequence. powerTimeSlot. schedule” can be modified with the proper modification according to section 4.3.24.3.2.7.3. However, this requires the consideration of the whole sequence’s schedule constraints (including all slot schedule constraints).

The duration of a slot is configurable if all of the following conditions are fulfilled:

1. The slot schedule is configurable.
2. In “alternatives. powerSequence. powerTimeSlot. scheduleConstraints” the child elements minDuration AND maxDuration are present.

If a slot’s duration is configurable the following rules apply:

1. minDuration and maxDuration SHALL be non-negative.
2. minDuration SHALL be smaller than maxDuration.
3. defaultDuration SHALL be in the range [minDuration, maxDuration].

Note: In case a duration uncertainty is given (see 4.3.24.3.2.3), the final duration of a slot (i.e. the measured duration once the slot was completed) may violate a boundary given by minDuration or maxDuration. This is because minDuration and maxDuration denote only configuration boundaries. This is independent from the capability of forecasting (i.e. estimating) a final duration from a given configuration.

The element “alternatives. powerSequence. powerTimeSlot. scheduleConstraints” is malformed if only one of the child elements minDuration and maxDuration is present.

A slot can be activated or deactivated if all of the following conditions are fulfilled:

1. The slot schedule is configurable.
2. In “alternatives. powerSequence. powerTimeSlot. scheduleConstraints” the child element optionalSlot is present and set to “true”.

See also section 4.3.24.3.2.7.3 on process related restrictions to schedule configurability!

The schedule information of a determined slot with implicit schedule (see 4.3.24.3.2.6) is not configurable.

4.3.24.3.2.7.3 Process rules

This section discusses rules that affect determined slots with configurable as well as non-configurable schedule.

The schedule configuration of a power sequence's slots is modelled within the element "alternatives. powerSequence" of a "powerSequences configuration" message (see section 4.3.24.4.4), where each instance of the child element "powerTimeSlot. schedule" contains the schedule configuration of a single slot.

1. Uniqueness and order:
   The slot schedule configuration list SHALL be unique and ordered: Within "alternatives. powerSequence" each slotNumber SHALL occur not more than once. The slots in the list SHALL be ordered by the slotNumber according to section 4.3.24.3.2.2.3.
2. Slots with non-configurable schedule:
   Slots with non-configurable schedule do not need to be listed in "alternatives. powerSequence" unless stated otherwise (some of the following rules REQUIRE the presence for specific cases; however, the presence is not required in general). If listed anyhow, the slot's schedule configuration MAY be empty (except for slotNumber) and SHOULD NOT contradict the slot's schedule information. If evaluated, the schedule configuration of slots with non-configurable schedule SHALL be ignored.
3. Completeness of a slot's schedule configuration:
   Within a message to configure one or more slots, each instance of "alternatives. powerSequence. powerTimeSlot. schedule" of a configurable slot SHALL be complete with regards to the configurable options. I.e. if a schedule configuration is given, it SHALL configure ALL schedule information the slot permits for configuration.
   Example: If a slot can be deactivated and the duration can be modified it is not sufficient to configure only the activation or only the duration.
4. Contributions to the schedule configuration list:
   Within "alternatives. powerSequence" there can be "alternatives. powerSequence. powerTimeSlot. schedule" instances for "past slots", the "current slot", and "future slots". A "past slot" is a slot that was already executed. The "current slot" is the slot that is currently executed. A "future slot" is a slot that is not executed yet.
5. On "past slots":
   A "past slot" SHALL be treated as non-configurable slot, even if it was configurable before its execution. Its "alternatives. powerSequence. powerTimeSlot. schedule" instance MAY be part of "alternatives. powerSequence". However, such a slot's instance of "alternatives. powerSequence. powerTimeSlot. schedule" has no further functional relevance within this specification.
6. On the "current slot":
   a. It SHALL NOT be possible to deactivate a "current slot". This means even if the slot's child element "optionalSlot" (see 4.3.24.3.2.7.2) is available and set to "true" it SHALL be interpreted as if the element is set to "false".
      Remark: This rule is only about the slot's specific flag "optionalSlot". It is independent from any possibility to modify a whole power sequence's state.

b. If a configurable slot becomes the current slot it MAY remain configurable or it MAY become non-configurable (this is finally implementation specific to the power sequences server).
c. The current slot's instance of "alternatives. powerSequence. powerTimeSlot. schedule" SHALL be listed in "alternatives. powerSequence" (regardless whether it is configurable or not; see rules above).
d. If the power sequences server provides a "alternatives. powerSequence. state" instance of the power sequence with the current slot, the following rules apply:
    i. The child element "remainingSlotTime" SHALL be set to the duration the current slot still needs to be active. I.e. in case the power sequence itself is paused for a while (hence the current slot can also not be active) the value of remainingSlotTime SHALL NOT decrease.
    ii. The child element "elapsedSlotTime" MAY be present. If it is present it SHALL be set to the duration the current slot has been active. I.e. in case the power sequence itself is paused for a while (hence the current slot can also not be active) the value of elapsedSlotTime SHALL NOT increase.
    iii. The sum of remainingSlotTime's value and the duration the current slot has been active SHALL be at least approximately equal to the slot's defaultDuration. At least a deviation according to durationUncertainty is permitted.
    Note: This also means the current slot's defaultDuration SHALL always cover the time it began (i.e. the past) up to the time it will end (i.e. the future). I.e. this value SHALL NOT be reduced just because a part of the slot's overall activity was already completed. It also means that phases where the slot (sequence) was paused do not require an update of defaultDuration.

7. On "future slots":
   All "future slots"' instances of "alternatives. powerSequence. powerTimeSlot. schedule" SHALL be listed in "alternatives. powerSequence" (regardless whether they are configurable or not; see rules above).

Please note: Some of the rules describe situations where the configurability of a slot changes. Of course, this requires a proper information update from the power sequences server to the power sequences client in general. However, this update process is not subject of this section.

4.3.24.3.2.8 Dependency between power/energy and slot duration configuration

This section (4.3.24.3.2.8) applies for slots with configurable duration according to section 4.3.24.3.2.7 only!

According to section 4.3.24.3.2.5 miscellaneous power OR energy values can be used as forecast of a slot's consumption or production. It shall be noted that this requires special consideration in conjunction with slots of configurable duration. In general a slot with a configurable duration can be modelled with an energy forecast OR a power forecast, resulting in two fundamentally different consequences:

1. A slot with a configurable duration and an exposed "valueType" element stating an energy-related information, is expected to (nearly) have the same energy for all possible

configurations of durations it offers. I.e. the power is reduced when extending the duration and increased when shortening the duration.

2. A slot with a configurable duration and an exposed “valueType” element stating a power-related information is expected to (nearly) have the same power for all possible configurations of durations it offers. I.e. the energy is reduced when shortening the duration and increased when extending the duration.

Note: A slot that provides a configurable duration SHALL NOT provide “valueType”s related to energy and power for the same slot!

4.3.24.3.2.9 Duration facets of determined slots, overview

This section (4.3.24.3.2.9) applies for determined slots only! It provides a brief overview on several aspects of a slot duration.

The following aspects can be modelled:

1. A slot’s duration is “exact” or just given with an uncertainty: See 4.3.24.3.2.3.
2. A slot’s duration is configurable (i.e. modifiable) or not: See 4.3.24.3.2.7.
3. A slot is optional (i.e. can be deactivated) or not: See 4.3.24.3.2.7.

Thus, for optional as well as non-optional slots the following combinations are possible:

1. A slot with an “exact” duration (i.e. without any duration uncertainty) and no configurable duration.
2. A slot with a (average) duration and a duration uncertainty and no configurable duration.
3. A slot with an “exact” duration (i.e. without any duration uncertainty) and a configurable duration.
4. A slot with a (average) duration and a duration uncertainty and a configurable duration.

4.3.24.3.2.10Repeated slots vs. one-time slots

Within a given sequence, each phase is represented by a slot. Each slot can be assigned a unique number. In this situation, each slot is processed only one time.

However, some implementations favour a “repetition concept” where a sequence of phases repeats a number of times. This permits to model only the first occurrence of a phase. E.g. if a sequence has three phases “A”, “B”, “C” and needs to process the phase sequence { A, B, C } two times, i.e. { A, B, C, A, B, C }, it is possible to model this with three slots only (e.g. slotNumber values 1, 2, 3) and an additional repetition counter. This leads to the case where the slots are processed in the order { 1, 2, 3, 1, 2, 3 }, i.e. each slot is repeated. However, this concept does not contradict any rule of section 4.3.24.3.2. As this concepts requires the consideration of a whole sequence, more details are discussed in section 4.3.24.3.5.3.

4.3.24.3.3 Sequences

4.3.24.3.3.1 Chronological classification of single sequence

This section defines some terms used to classify power sequences chronologically. It also imposes restrictions on the use of the power sequence's "state" information element ("alternatives. powerSequence. state").

1. "Past sequence":
   A "past sequence" is a sequence that was relevant in the past. This means the sequence was processed or was at least offered. The sequence is neither "now" nor in the future offered or considered by the power sequences server. The power sequence's "state" information element SHALL be either "completed" or "invalid".
2. "Pre-announced sequence":
   A "pre-announced sequence" is a sequence that can be relevant for a time not earlier than "now". However, it is not a "current sequence". The power sequence's "state" information element SHALL be either "scheduled" or "scheduledPaused" or "inactive".
3. "Current sequence":
   A "current sequence" is a sequence that is currently processed by a power sequences server. The power sequence's "state" information element SHALL be either "running" or "paused".
   Note: If a "pre-announced sequence" becomes a "current sequence" the natural "state" transition is as follows: "scheduled" changes to "running" and "scheduledPaused" changes to "paused".

Note: The above mentioned "relevance" does not only apply to sequences that were or will be executed. It also applies to sequences just offered as choice (to the bound power sequences client).

Note: The term "past sequence" does not say anything about the time frame the sequence was relevant for. A "past sequence" may have been a "pre-announced sequence" recently and notified with other "pre-announced sequences" within an alternatives group (see 4.3.24.3.3.2) by the power sequences server. Suddenly a sequence became "invalid" for any reason, hence is a "past sequence", now. This example shows a "past sequence" can belong to the same time frame as a "pre-announced sequence" (e.g.). Nevertheless, it is not offered any more by the power sequences server for a choice by the bound power sequences client.

4.3.24.3.3.2 Chronological classification of alternatives group

Sequences belonging to the same "alternatives" element constitute an alternatives group (see section 4.3.24.2.2). This section defines some terms used to classify such a group chronologically. It also imposes rules on the creation of such groups.

1. "Past group":
   A "past group" is a group that contains only "past sequences".
2. "Pre-announced group":
   A "pre-announced group" is a group that contains only "pre-announced sequences".
3. "Current group":
   A "current group" is a group that SHALL contain exactly one "current sequence".

Furthermore, it MAY contain any number of “past sequences”. It SHOULD NOT contain any “pre-announced sequence”.

Note: A power sequences server may face a condition where a previously notified “pre-announced sequence” is not valid anymore and becomes a “past sequence”. According to the rules above this sequence SHALL NOT be part of any update of the alternatives group it formerly did belong to. However, it is still possible to notify the state update of the sequence only.

4.3.24.3.3.3 Sequence schedule information

A sequence has “sequence schedule information” if all the following conditions are fulfilled:

1. The information element “alternatives. powerSequence. schedule” is present and at least its child element “startTime” is present as well.

Furthermore, the following rules apply:

1. A “pre-announced sequence” SHALL NOT have a startTime referring to the past.
2. If the child element “endTime” is available its value SHALL point to a time later than “startTime”.

Remark: For “sequence schedule modification” see 4.3.24.3.3.7.

4.3.24.3.3.4 Sequence selection types

This section applies to “pre-announced groups” and “current groups” only.

Esp. for a sensible control of a power sequences server it is important to identify if or which sequence of an alternatives group is “currently preferred or chosen” and if or which sequences are available as alternative (i.e. offered as choice to the bound power sequences client). The following selection types and rules are defined:

1. “Chosen sequence”:
    a. A sequence is called a “chosen sequence” if it has sequence schedule information. Its state SHALL be “scheduled” or “scheduledPaused” if it is a “pre-announced sequence”. It SHALL be “running” or “paused” if it is a “current sequence”.
    b. If it is a “pre-announced sequence” the power sequences server SHALL execute the sequence autonomously at the sequence’s startTime (i.e. it becomes a “current sequence”) provided that the sequence is still valid and no other sequence of the alternatives group has been selected (by the bound power sequences client (through configuration) or by the power sequences server (for internal reasons or by user’s choice, e.g.)) as “chosen sequence”.
    Note: The autonomous start may well result in an immediate “paused” state if the sequence’s state was “scheduledPaused” before (see section 4.3.24.3.3.1).
    c. Each “current sequence” SHALL be a “chosen sequence”.
    d. Within an alternatives group at maximum one sequence can be a “chosen sequence”.
2. “Optional sequence”:

4092 a. A sequence that has no sequence schedule information and its state information
4093 element set to “inactive” is called an “optional sequence”.
4094 b. An “optional sequence” SHALL NOT be executed by the power sequences server
4095 autonomously. If, for some reason, there is a request or need to execute this
4096 sequence it has to be modified to become a “chosen sequence”.
4097 c. Remark: Within a “pre-announced group” it is permissive to have only “optional
4098 sequences” and no “chosen sequence”.
4099 3. “Unavailable sequence”:
4100 a. A sequence that is neither a “chosen sequence” nor an “optional sequence” as called
4101 an “unavailable sequence”. It SHALL NOT have a sequence schedule information and
4102 it SHALL be a “past sequence”.
4103 Please note: This section is NOT about “past groups”.
4104 b. An “unavailable sequence” SHALL NOT be executed by the power sequences server.

4105

4106 *4.3.24.3.3.5 Sequence modifiability*

4107 *4.3.24.3.3.5.1 Introduction (informative)*
4108 Modifiability is required if a bound power sequences client wants to change any of the following data
4109 of the power sequences server’s sequence:

4110 1. The sequence overall schedule (information elements “startTime”, “endTime”).
4111 2. The sequence state (interruptibility, i.e. pausing/resuming or aborting a current sequence).
4112 3. The sequence’s slot data (slot durations, activation/deactivation, energy/power).

4113 Dependent on the data further specific conditions may be required.

4114 Furthermore, modifiability is also required if a power sequences client wants to make any of the
4115 offered “optional sequences” a “chosen sequence”.

4116

4117 *4.3.24.3.3.5.2 Definition*
4118 A sequence is called “modifiable” if the sequence’s information element
4119 “sequenceRemoteControllable” is available and set to “true”. Otherwise it is not modifiable.

4120

4121 *4.3.24.3.3.6 Sequence interruptibility*

4122 *4.3.24.3.3.6.1 Definition*
4123 A sequence is called “interruptible” if all of the following conditions are fulfilled:

4124 1. The sequence is modifiable.
4125 2. The information element “alternatives. powerSequence. operatingConstraintsInterrupt” is
4126 available AND at least one of the child elements “isPausable” and “isStoppable” are set to
4127 “true”.

Otherwise a sequence is not interruptible. This also means the absence of the information element "alternatives. powerSequence. operatingConstraintsInterrupt" assigns "isPausable" and "isStoppable" the value "false" implicitly.

The process of interrupting/resuming a sequence may be further constrained by additional data as explained in 4.3.24.3.3.6.2.

4.3.24.3.3.6.2 Process rules

Only "current sequences" and partly "pre-announced sequences" can be interrupted and resumed by the bound power sequences client. However, further conditions apply:

1. A "current sequence" can be paused, resumed or aborted dependent on the interruptibility data (see 4.3.24.3.3.6.1) and the state of the sequence (see 4.3.24.4.5.1.2).
2. A "chosen sequence" of a "pre-announced group" (i.e. a "chosen pre-announced sequence") can be changed as follows:
    a. Its state can be changed from "scheduled" to "scheduledPaused" if the sequence is interruptible and "isPausable" is set to true.
    b. Its state can be changed from "scheduledPaused" to "scheduled".
    c. It can be changed to become an "unavailable sequence", provided that the sequence is interruptible in general and "isStoppable" is set to "true". This requires the modification of the sequence's information element "state" to "invalid" (see 4.3.24.4.5.1.2).

    Remark: This item just discusses "direct" state changes. It does not discuss "indirect" state changes (e.g. from the configuration of another sequence).
3. There can be further constraints which need to be considered as explained below.

The process of interrupting/resuming a power sequences server REQUIRES that the bound power sequences client advises the power sequences server to change its state. Consequently, a bound power sequences server accepts such commands only if it permits the proper interruptibility as stated above. Please note that this process is in general independent from a sequence configuration through the bound power sequences client.

If a sequence is interruptible and further constraints are specified in "alternatives. powerSequence. operatingConstraintsInterrupt" or "alternatives. powerSequence. operatingConstraintsDuration", these constraints SHALL be considered by the bound power sequences client (esp. in order to prevent damage of the power sequences server).

The information element "alternatives. powerSequence. operatingConstraintsResumeImplication" SHALL only be available if "alternatives. powerSequence. operatingConstraintsInterrupt" is available AND the child element "isPausable" is available. It SHALL NOT be considered if "isPausable" is set to "false".

4.3.24.3.3.7 Sequence schedule modification

4.3.24.3.3.7.1 Introduction (informative)

Section 4.3.24.3.3.3 describes “sequence schedule information” itself. This section (4.3.24.3.3.7) describes some rules to set or modify “sequence schedule information”.

4.3.24.3.3.7.2 Definition

A sequence is called to have a “modifiable sequence schedule” if all of the following conditions are fulfilled:

1. The sequence is either a “chosen sequence” or an “optional sequence”.
2. The sequence is modifiable (see 4.3.24.3.3.5).
3. The information element “alternatives. powerSequence. scheduleConstraints” is available.

Otherwise a sequence’s schedule information is not modifiable.

4.3.24.3.3.7.3 Process rules

If a “chosen sequence” has a “modifiable sequence schedule” a bound power sequences client MAY modify the sequence’s “sequence schedule information”.

If an “optional sequence” has a “modifiable sequence schedule” a bound power sequences client MAY assign the sequence a “sequence schedule information”.

4.3.24.3.3.8 Sequences configuration

4.3.24.3.3.8.1 Introduction (informative)

This section discusses the relation and dependency of modifiability aspects as described in section 4.3.24.3.3.5.1. Furthermore it describes how the configuration of one sequence may affect another sequence of the same alternatives group.

4.3.24.3.3.8.2 Definition

A sequence is called “configurable” if all of the following conditions are fulfilled:

1. The sequence has a “modifiable sequence schedule” according to section 4.3.24.3.3.7.
2. The sequence belongs to a “current group” or “pre-announced group”.

Otherwise a sequence is called “not configurable”.

4.3.24.3.3.8.3 Process rules

Throughout this section only “current groups” and “pre-announced groups” are considered. Furthermore, only “chosen sequences” and “optional sequences” of these groups are considered and are called “applicable sequences” within this section. The following rules apply:

1. Applies to “pre-announced groups” only: Each “optional sequence” SHALL be configurable.
2. Applies to “pre-announced groups” only: If a bound power sequences client assigns a configurable “optional sequence” a “sequence schedule information” (see 4.3.24.3.3.7.3) the sequence becomes then a “chosen sequence”. If there was another sequence of the alternatives group a “chosen sequence”, this other sequence becomes an “optional sequence” or (implementation specific) a “past sequence” then.
3. A “current sequence” SHALL NOT become an “optional sequence” through the configuration of another sequence that is an “optional sequence” of the alternatives group. I.e. once a sequence becomes a “current sequence” it remains the “current sequence” until it completes successfully or is aborted finally (if applicable).
4. Each “applicable sequence” that is configurable MAY be interruptible according to section 4.3.24.3.3.6.
5. Each “applicable sequence” that is configurable MAY have configurable slots according to section 4.3.24.3.2.7. If a sequence needs to have configurable slots the sequence itself SHALL be configurable.
6. A configurable sequence MAY become a non-configurable sequence or vice versa at any time. Whether this happens is up to the power sequences server and implementation specific. However, specific process rules may impose restrictions on such changes.

Note: If a “chosen sequence” becomes an “optional sequence” (or even “unavailable sequence”) its information changes. Among others, this can “remove” a sequence’s schedule information (see section 4.3.24.3.3.4).

Please note that the configuration of a sequence through the bound power sequences client is in general independent from a sequence’s interruptibility. However, dependent on the implementation or further process rules it may well happen that the process of sequence configuration leads to a change of the sequence’s interruptibility.

4.3.24.3.4 Overall energy management information

This section describes briefly some “global information” and rules on its use.

The information element “nodeScheduleInformation” contains information on the device’s overall energy management state, i.e. regardless of a specific sequence or slot. Some of its child elements are rather static (constant) while others typically vary over time. However, the following rule applies in every case: The content of “nodeScheduleInformation” SHALL NOT contradict the sequence- or slot-specific information. This also means that the content does not “supersede” sequence- or slot-specific information. This is relevant esp. for the following information:

1. Remote controllability
   A power sequences server is “remote controllable” exactly if it has at least one sequence that is configurable (see 4.3.24.3.3.8.2).
2. Scheduling of single slots vs. multiple slots
   Even if a power sequence permits the configuration of slots it might be that the power sequences server requires individual configuration messages for each slot. I.e. it may not support the configuration of several slots within a single message. This is denoted by the element “supportsSingleSlotSchedulingOnly”.

3. One-time sequence selection
   A power sequences server may offer to select a sequence (i.e. an “optional sequence” becomes a “chosen sequence”). As long as the sequence did not start it might be possible to alter the selection, i.e. make another sequene a “chosen sequence”. However, some device may not permit such multiple re-selections and permit just a one-time selection. The device’s possibility for re-selection is denoted by the element “supportsReselection”.

The information element “nodeScheduleInformation” can contain more than the items mentioned above. However, these are explained in the definition of the element and are not relevant for process considerations.

4.3.24.3.5 Rules for creation of information

This section primarily focuses on rules to be considered by the power sequences server. However, the rules are relevant for a power sequences client as well.

4.3.24.3.5.1 On the use of “alternativesId”

As described in section 4.3.24.3.3.2 each “alternatives” element of the server’s power sequences information contains the power sequences of the same alternatives group. Each of these groups can be identified with the element “alternatives. relation. alternativesId”. Therefore, each alternatives group exposed by a power sequences server SHALL always have a unique value in "alternativesId", i.e. there SHALL never be concurrently valid, duplicate values of “alternativesId”. Furthermore, a power sequences server SHALL NOT reuse values of “alternativesId” immediately after an alternatives group is not present anymore on the power sequences server.

As power sequence IDs are unique across all alternatives groups (see section 4.3.24.3.5.2) the use and evaluation of alternativesId is required just in case of “restricted function exchange” (see [ProtocolSpecification]). This is in particular the case if a new power sequence is notified for an already existing alternatives group.

4.3.24.3.5.2 On the use of (power) “sequenceId”

Each power sequence exposed by a power sequences server SHALL always have a unique value in its power sequence ID (element "sequenceId"), i.e. there SHALL never be concurrently valid, duplicate power sequences IDs. Furthermore, a power sequences server SHALL NOT reuse power sequences IDs immediately after sequences with the according IDs have been completed. As an example: A washing machine should not expose its cycles with a fixed sequence ID of “1” all the time. Instead, it should increase the sequence ID with every new sequence. In general, a power sequence ID is just a simple identifier (to select a specific sequence from a list) and SHALL NOT be interpreted to carry any additional information, such as information about which sequence runs after another judging from ascending or descending IDs.

4.3.24.3.5.3 On the embedding of slots in sequences

Sequences are used together with slots within this specification.

4.3.24.3.5.3.1 Common rules

The following common rules apply:

4. Each sequence SHALL have at least one slot.
5. Each sequence SHALL EITHER have only determined slots OR exactly one undetermined slot (see 4.3.24.3.2.2).
6. Applies only to sequences with at least one determined slot: For at least one slot it SHALL NOT be possible to deactivate it and it SHALL always have a duration greater than 0 seconds. Remark: This requirement has no impact on the potential possibility to “deactivate” the sequence itself.

4.3.24.3.5.3.2 Embedding slots in case of “repetition concept”

Some additional considerations are required in case of the “repetition concept” as briefly explained in section 4.3.24.3.2.10. Again we consider the example where a sequence consists of three different phases “A”, “B”, “C” that are processed two times, resulting in an overall phase cycle { A, B, C, A, B, C }. We assume this is represented with three slots 1, 2, 3 and accordingly the number of executions is set in the element “alternatives. powerSequence. description. repetitionsTotal” to two (please note that “repetition” here means “number of execution”). Then, the number of the current repetition has to be set in the element “alternatives. powerSequence. state. activeRepetitionNumber”.

As just three explicit slot information entities are used in this example (instead of the effectively six phases/slots) this means that rules have to be defined and considered whether a slot’s information belongs to the first repetition (i.e. first execution) only or to every repetition or is updated with each repetition. These rules are defined in the element “alternatives.powerSequence.slotInformation” of message “smartEnergyManagementPsData”. As a rule of thumb, slot information of this message usually belongs to the current repetition unless stated otherwise. This means information for slot number 1 still belongs to the current repetition (in contrast to the next repetition) even if it was already processed and slot number 2 is currently processed. Constraints, on the other hand, usually (but not always) apply to each repetition. In general a configuration of slots shall be considered to apply to every repetition.

4.3.24.3.5.4 Prevent group overlap in time

Pre-announced alternatives groups SHALL NOT overlap in time. No pre-announced alternatives group SHALL overlap in time with a current alternatives group.

This means no schedule or schedule constraints of any sequence of alternatives group “A” shall overlap with any schedule or schedule constraints of any sequence of alternatives group “B”.

Remark: Of course, sequences of the same alternatives group may well overlap in time.

As a consequence, some restrictions SHALL be applied if

1. EITHER one current alternatives group and at least one pre-announced group,

2. OR at least two pre-announced groups

are created:

1. All pre-announced sequences of the alternatives group that is later in time SHALL have a schedule with startTime AND endTime or they SHALL have sequence schedule constraints with earliestStartTime AND latestEndTime.

4.3.24.3.5.5 Permitted sequence types

4.3.24.3.5.5.1 Introduction

The data models, rules (and messages described subsequently) are designed to permit a limited number of different sequence types or processes. These types are briefly described in the following sections. The description primarily makes use of terms defined in previous sections and is extended with specific restrictions or rules where required. This approach shall also help to compare the sequence types.

4.3.24.3.5.5.2 Sequence type "fixed forecast"

This type is designed to notify a sequence where a power sequences client will not be given any kind of control. I.e. this sequence denotes a completely autonomous schedule of the power sequences server.

A power sequence with this sequence type SHALL be initially announced as follows:

1. It SHALL be notified by the power sequences server before its execution as "pre-announced sequence" unless the sequence was created suddenly and begins immediately. If it was created suddenly and begins immediately it SHALL be notified as "current sequence". In either case the sequence SHALL be the only sequence in the alternatives group (which is accordingly a "pre-announced group" or "current group", resp.).
2. It SHALL be notified as "chosen sequence". Additionally, its "alternatives. powerSequence. schedule" child element "endTime" SHALL be set.
   Remark: Of course, slot duration uncertainties can have an impact on the precision of the whole sequence's end time as well.
3. The sequence SHALL NOT be interruptible.
4. The sequence SHALL NOT be configurable.
5. The sequence SHALL have at least one slot.
6. All slots of the sequence SHALL be determined.
7. All slots of the sequence SHALL have an implicit slot schedule (hence are also not configurable).

If the sequence becomes a "current sequence" it SHALL remain non-interruptible and non-configurable.

4.3.24.3.5.5.3 Sequence type "flexible chosen forecast"

This type is designed to notify a sequence where a power sequences client usually will be given some control. I.e. this sequence denotes the preference or current configuration a power sequences server will execute autonomously unless the power sequences client takes its possibilities to modify the schedule or state.

A power sequence with this sequence type SHALL be initially announced as follows:

1. It SHALL be notified by the power sequences server before its execution as "pre-announced sequence". It SHALL only be notified within a "pre-announced group".
2. It SHALL be notified as "chosen sequence". Additionally, its "alternatives. powerSequence. schedule" child element "endTime" MAY be set.
3. The sequence MAY be interruptible.
4. The sequence SHALL be configurable.
5. The sequence SHALL have at least one slot.
6. All slots of the sequence SHALL be determined.
7. Each slot SHALL have EITHER a constrained schedule OR an implicit schedule. The general rules on configurable and non-configurable slots of section 4.3.24.3.2.7 apply.

If the sequence becomes a "current sequence" its interruptibility and configurability MAY be changed by the power sequences server.

4.3.24.3.5.5.4 Sequence type "flexible optional forecast"

This type is similar to the type of section 4.3.24.3.5.5.3 but denotes just an option. I.e. a power sequences server will NOT execute this sequence autonomously. However, a power sequences client can take its possibilities to modify the schedule or state in order to make this sequence a "chosen sequence" (see below also).

A power sequence with this sequence type SHALL be initially announced as follows:

1. It SHALL be notified by the power sequences server before its execution as "pre-announced sequence". It SHALL only be notified within a "pre-announced group".
2. It SHALL be notified as "optional sequence".
3. The sequence MAY be interruptible.
4. The sequence SHALL be configurable.
5. The sequence SHALL have at least one slot.
6. All slots of the sequence SHALL be determined.
7. Each slot SHALL have EITHER a constrained schedule OR an implicit schedule. The general rules on configurable and non-configurable slots of section 4.3.24.3.2.7 apply.

See section 4.3.24.3.3.8 in order to modify the kind of the sequence or its state.

In case this sequence becomes a "chosen sequence" please see section 4.3.24.3.5.5.3, as this change corresponds to the change of the sequence type.

4.3.24.3.5.5.5 Sequence type "optional sequence-based immediate control"

This type is designed for devices that cannot forecast their consumption or production with "sufficient precision" but that can at least announce an optional task with upper or lower bounds (e.g. schedule constraints or min/max values). For this sequence type the client is given at least some kind of control in order to pause and resume this task (considering constraints imposed by the power sequences server). This sequence type is initially optional and requires a proper configuration from the power sequences client in order to start at the configured time. Afterwards the power sequences client can use its options on the interrupt/resumption of the sequence.

A power sequence with this sequence type SHALL be initially announced as follows:

1. It SHALL be notified by the power sequences server before its execution as "pre-announced sequence". It SHALL only be notified within a "pre-announced group".
2. It SHALL be notified as "optional sequence".
3. The sequence SHALL be interruptible.
4. The sequence SHALL be configurable.
5. The sequence SHALL have exactly one slot.
6. The slot of the sequence SHALL be undetermined.
   Remark: This item and item 5 are in accordance with the rules in section 4.3.24.3.5.3.
7. Remark: Due to item 6 the slot itself is not configurable.
8. The child element "endTime" of "alternatives. powerSequence. schedule" SHALL NOT be set unless the sequence becomes a "past sequence". I.e. neither can the power sequences server announce "endTime" in advance nor after the configuration of the schedule nor shall the power sequences client try to configure "endTime".
9. Remark: The absence of any duration prevents a price forecast (see 4.3.24.4.9.1).

See section 4.3.24.3.3.8 in order to modify the kind of the sequence or its state.

If the sequence becomes a "current sequence" it SHALL NOT be configurable any more. Its interruptibility MAY be changed by the power sequences server (temporarily).

Please note: This sequence type typically requires at the power sequences client a strong consideration of constraints imposed by the power sequences server. Furthermore, this sequence type is typically more likely to be aborted at some point of time by the power sequences server autonomously.

4.3.24.3.6 Rules for evaluation

This section primarily focuses on rules to be considered by the power sequences client. It is based on the sequence grouping concept and section 4.3.24.3.5.

A power sequences client should analyse received information on sequences as follows:

1. Order alternatives groups chronologically:
   Received alternatives groups SHALL be classified chronologically according to section 4.3.24.3.3.2. Afterwards the chronological order of current or pre-announced groups shall be derived according to the rules defined in section 4.3.24.3.5.4 to prevent overlapping groups.
2. Within each alternatives group consider rules of section 4.3.24.3.5.

4.3.24.3.7 Conventions

This section describes some further conventions that SHALL apply in every case.

4.3.24.3.7.1 Production vs. generation, sign convention:

The basic PowerSequences class defines the element “positiveEnergyDirection” in order to model whether positive or negative values (of power/energy) denote the production of energy. This element is not used explicitly within the complex class of this specification and is replaced by the following convention instead:

1. Power and energy values less than zero (negative value sign) denote the production of energy.
2. Power and energy values greater than zero (positive value sign) denote the consumption of energy.

This also means an implicit assignment of “positiveEnergyDirection = consume”.

4.3.24.3.7.2 Update of time information elements

All time information that identifies a point in time SHALL be updated continuously with every message communicated (e.g. the startTime of a sequence). Time information that specifies a duration of some slot or sequence that do not change over time SHALL keep the same value (e.g. the defaultDuration of a slot).

4.3.24.3.7.3 Contiguous sequence data

Within a message instance, information on a given power sequence SHALL be contiguous. I.e. information on the power sequence with ID "1" is not interrupted with information on the power sequence with ID "2". The contiguity also means each power sequence SHALL be split into as few data groups as possible (unless stated otherwise by a specific message definition).

4.3.24.3.8 Role concept

If a device is a power sequences server and a power sequences client at the same time, the instances of the power sequences server and the power sequences client SHALL NOT be present on the same entity. There SHALL NOT be more than one instance of a power sequences server on the same entity as well. However, there MAY be multiple instances of power sequences clients on one entity.

4.3.24.3.9 Binding considerations

Power sequences servers SHALL only accept one binding to a power sequences client for each instance of a power sequences server entity. Once the power sequence instance is bound to a client, the power sequences server SHALL reject additional binding requests for this power sequence instance.

If a power sequences server detects that a bound power sequence client is not accessible any more (e.g. since it has been sold to another household), it is a vendor specific implementation to either delete the binding or continue waiting for the bound device. Thus, it might only be possible for device owners to re-bind the power sequences server with a factory reset of the device.

Devices which are interested in data of a power sequences server, but don't configure the power sequences server flexibilities, SHALL only use data subscriptions or explicit read commands without bindings to receive updated information and SHALL never try to create bindings to a power sequences server.

For more details about binding, please consider [ProtocolSpecification] .

4.3.24.3.10 General error handling

4.3.24.3.10.1 Acknowledgement concept

The acknowledgement concept as specified in [ProtocolSpecification] applies. A bound power sequences server and bound power sequences client SHOULD explicitly request acknowledgement messages for message exchanged among each others (provided the message type permits acknowledgement request).

4.3.24.3.10.2 Valid vs. invalid classifiers

Subsequent message specific sections contain sub-sections with the headline "When to send". These sub-sections determine explicitly under which condition a given message shall be sent and also which classifier(s) shall be used. These classifiers are subsequently considered valid for the respective messages and conditions. Other classifiers are considered invalid for the respective messages.

A brief overview of valid classifiers is given in Table 113. Furthermore, sub-sections with the headline "When received" usually mention valid classifiers only.

4.3.24.4 Message purpose specification details

This section defines details of the message purposes that were briefly announced in Table 113 of section 4.3.24.1.

4.3.24.4.1 powerSequences information

4.3.24.4.1.1 General data model description and rules

4.3.24.4.1.1.1 General aspects

With a power sequences information message a power sequences server exposes all power sequences (potentially) relevant for the power sequences client, which may be a combination of:

- Alternative processes to choose from:
  This has already been introduced in section 4.3.24.2.2 as choice of power sequences: A

device may have different operation modes or (more generally speaking) mutually exclusive processes, which it offers to a power sequences client. The client can only choose one of these mutually exclusive processes. These processes SHALL be exposed within one “alternatives” element.

- Single or multiple, sequenced processes:
  A device may have different processes which are executed in a serialized manner, i.e. cannot be parallelized or run concurrently. These processes SHALL be exposed within different “alternatives” elements.

A power sequences server can also expose a combination of these options. As an example, we consider the case of an HVAC system, which knows that it has to run in the morning and in the evening. These processes cannot be run concurrently. However, it has 2 modes of heating: One with high power consumption, but shorter heating cycle and one with a lower power consumption and longer heating cycle. In this case, the HVAC system would expose 2 “alternatives” elements, each containing 2 power sequences. One of these pairs of power sequences would reflect the different power consumption modes during the morning whereas the second pair does the same for the evening.

4.3.24.4.1.1.2 Cancelling (deleting/aborting) offered power sequences

There might be situations in which a power sequences server offers one or more power sequences, but encounters a situation which renders one or more of the offered power sequence(s) invalid. In such situations the affected power sequences SHALL be cancelled/deleted. As an example, a washing machine might offer a power sequence representing a washing cycle, but a few seconds later the user of the washing machine cancels the washing cycle, because he has chosen the wrong washing program. If a power sequences server has offered one or more power sequences which are now invalid, it SHALL send a “powerSequences information” with a notify classifier and set the according power sequence to an “invalid” state. Afterwards it SHALL remove the according power sequence ID and respect the rules on reusing sequence IDs (see section 4.3.24.3.5.2). The deletion of the sequence MAY result in a delete-notification (see section "Restricted function exchange with cmdOptions" in [ProtocolSpec]), but it is not recommended to do so.

In general, a power sequences server SHALL cancel all power sequences that have been announced and possibly also configured, but are no longer valid.

Errors might occur if a power sequences client does not receive the notification(s) about cancelled power sequence(s), e.g. because the power sequences client is currently offline. However, in these situations the power sequences client SHOULD be aware of potentially lost packets and SHOULD apply appropriate mechanisms, such as explicitly requesting the power sequences available.

4.3.24.4.1.1.3 Processes that might be run in parallel

Please note a single instance of a power sequences server SHALL NOT expose processes that (potentially or partly) run at the same time! To give an example: Consider a fridge/freezer combination with independent temperature control. I.e. for both the fridge temperature control as well as for the freezer temperature control a process can be exposed. However, as these processes

are independent from each other (hence can also run in parallel) each process has to be exposed by an individual power sequences server instance. This means there has to be a power sequences server for cooling processes and another power sequences server for freezing processes. As stated in section 4.3.24.3.8, these power sequences server instances SHALL be located on different entities.

4.3.24.4.1.1.4 Power sequences with and without flexibilities

Please note that power sequences might or might not contain flexibilities, i.e. offer time-shifting of a power sequence or slot or offer slots which have durations which can be reduced or extended, i.e. have a configurable duration. A power sequence with schedule flexibilities SHALL contain schedule constraints. A power sequence not containing flexibilities, i.e. a fixed forecast, SHALL NOT contain schedule constraints.

4.3.24.4.1.1.5 Consistency of a power sequence

In case of a sequence with flexible schedule (and determined slots), a sequence's "earliest start time" and "latest end time" SHALL always frame the slots that are part of the sequence. I.e. the sum of durations of the slots in a sequence SHALL always fit into the frame given by these elements. Care has to be taken if slots can be extended regarding their duration, i.e. which have a configurable duration: Extending all durations to their maximum may exceed the time frame permitted by a sequence's "earliest start time" and "latest end time". I.e. such configurations are invalid and shall be rejected by the power sequences server. However, there SHALL at least be one possible configuration of the duration of all slots in a sequence that fits into the frame given by "earliest start time" and "latest end time" of the sequence.

As an example, consider the slots 1, 2 and 3 of a sequence. All of these slots may have a typical duration of 5 minutes, but all of them can be extended so they have a duration of 15 minutes each. This results in a minimum length of 15 minutes and a maximum length of 45 minutes for the whole sequence. However, the power sequences server might be limited to an "earliest start time" and "latest end time", which only provide a duration of 35 minutes for the whole sequence. In this example, the power sequences client would be free to either extend the duration of all slots equally so they fit into the frame, but it might also only extend slot 1 and 2 to their maximum duration. The power sequences client might, however, also choose to not extend the duration of any of the slots at all.

4.3.24.4.1.1.6 Possible state values

The states delivered in a power sequences state can have several, pre-defined values. This Feature Type uses as well the definition of PowerSequenceStateType (see Table 269, section 5.3.19.9.2), but restricts the permitted values. The following values are allowed within the scope of this specification:

- "running": The according sequence is currently executed.
- "paused": The according sequence is paused.
- "scheduled": The according sequence will be executed by the power sequences server at the scheduled time. It will then usually be in the state "running".

- “scheduledPaused”: The according sequence will be executed by the power sequences server at the scheduled time – but it will then usually be in the state “paused”.
- “inactive”: The according sequence has been offered as optional choice, but has not been configured.
- "completed": The according sequence has been completed.
- "invalid": The according sequence has been cancelled. Please consider section 4.3.24.4.1.1.2.

4.3.24.4.1.1.7 Data model description and rules

The complex function “smartEnergyManagementPsData” of the complex class “SmartEnergyManagementPs” is used with the following restrictions and rules:

Element name	**M/O/C (1.4.7)**	**Explanation**
nodeScheduleInformation	M	
nodeScheduleInformation. nodeRemoteControllable	M	If nodeRemoteControllable is equal to "false" no power sequence is remote controllable, independet from the value of the element "sequenceRemoteControllable". If nodeRemoteControllable is set to "true", it depends on the value of the element "sequenceRemoteControllable" if a specific sequence is remote controllable, or not.
nodeScheduleInformation. supportsSingleSlotSchedulingOnly	M	If set to “true” the power sequences server applies a restriction on received “powerSequences configuration” messages: The element “alternatives. powerSequence” SHALL NOT contain more than one child element “powerTimeSlot”. This restriction does not apply if supportsSingleSlotSchedulingOnly is set to “false”.
nodeScheduleInformation. alternativesCount	M	
nodeScheduleInformation. totalSequencesCountMax	M	
nodeScheduleInformation. supportsReselection	M	
alternatives	O	SHALL only be left out if no power sequences are present within its group. Otherwise, it SHALL be present.
alternatives. relation	M	
alternatives. relation. alternativesId	M	See section 4.3.24.3.5.1.
alternatives. powerSequence	M	
alternatives. powerSequence. description	M	

alternatives. powerSequence. description. sequenceId	M	
alternatives. powerSequence. description. description	O	Default rules of section 3.10.1.3.1 SHALL be applied.
alternatives. powerSequence. description. powerUnit	O	Element SHOULD be set. If ommitted, "W" SHALL be assumed as unit for the value implicitly.
alternatives. powerSequence. description. energyUnit	O	Element SHOULD be set. If ommitted, "Wh" SHALL be assumed as unit for the value implicitly.
alternatives. powerSequence. description. valueSource	O	If absent, the source of the forecasted values is undefined.
alternatives. powerSequence. description. taskIdentifier	O	
alternatives. powerSequence. description. repetitionsTotal	O	MAY be present. Absence of the element is equal to a presence with a value of 1 (one). SHALL be absent if the value is 1. If a power sequence executes its sequence of slots more than one time, the element SHALL be present and contain the total number of executions.
alternatives. powerSequence. state	M	
alternatives. powerSequence. state. state	M	Consider esp. section 4.3.24.3.3.1 on restrictions of permitted values from the enumeration!
alternatives. powerSequence. state. activeSlotNumber	O	If state is set to "running" or "paused" this element SHALL contain the currently active slot number. Otherwise it SHALL be omitted.
alternatives. powerSequence. state. elapsedSlotTime	O	If state is set to "running" or "paused" AND the slot is determined (see 4.3.24.3.2.2) this element MAY contain the time the slot has already been in “running” state (this also means the value remains constant during a “paused” state). Otherwise it SHALL be omitted.
alternatives. powerSequence. state. remainingSlotTime	O	If state is set to "running" or "paused" AND the slot is determined (see 4.3.24.3.2.2) this element SHALL contain the time the slot still needs to be in “running” state (this also means the value remains constant during a “paused” state). Otherwise it SHALL be omitted.

alternatives. powerSequence. state. sequenceRemoteControllable	M	Denotes whether the sequence is modifiable (value “true”) or not (value “false”), see 4.3.24.3.3.5. Modifiability is required to configure sequences and slots (see section 4.3.24.4.4). It is also required to change a power sequence state (if supported; see 4.3.24.4.5).
alternatives. powerSequence. state. activeRepetitionNumber	O	SHALL be present if alternatives. powerSequence. description. repetitionsTotal is present and has a value greater than 1. Otherwise, it SHALL be absent.
alternatives. powerSequence. state. remainingPauseTime	O	This element SHALL ONLY be present if the power sequence is interruptible. Otherwise, it SHALL be omitted. In case of the sequence’s interruptibility the following rules apply: If the element is absent this means there is no explicit pause duration restriction for the current slot; a value of 0s denoted the slot does not permit being paused.
alternatives. powerSequence. schedule	O	If the sequence has a schedule, this element with its sub-elements SHALL be present. Otherwise it SHALL be omitted.
alternatives. powerSequence. schedule. startTime	M	
alternatives. powerSequence. schedule. endTime	O	If the power sequence has an endTime, it SHALL be put here. Otherwise the element SHALL be omitted.
alternatives. powerSequence. scheduleConstraints	O	If there are constrains available, this element and its sub-elements SHALL be present. Otherwise it SHALL be omitted.
alternatives. powerSequence. scheduleConstraints. earliestStartTime	M	Only “xs:duration” value types SHALL be used to denote a relative time which relates to “now” as time 0.
alternatives. powerSequence. scheduleConstraints. latestEndTime	M	Only “xs:duration” value types SHALL be used to denote a relative time which relates to “now” as time 0.
alternatives. powerSequence. schedulePreference	O	MAY be present. A power sequences client should analyse this element when it configures the power sequences server. Note: There may be several options available which cannot be fulfilled altogether. This means the options can in general only be considered “as good as it can get”.
alternatives. powerSequence. schedulePreference. greenest	O	MAY be present. Absence of this element is equal to the presence with value “false”.
alternatives. powerSequence. schedulePreference. cheapest	O	MAY be present. Absence of this element is equal to the presence with value “false”.

alternatives. powerSequence. operatingConstraintsInterrupt	O	If the sequence may be paused or stopped by the power sequences client, this element and the required sub-elements SHALL be present. Otherwise it SHALL be omitted.
alternatives. powerSequence. operatingConstraintsInterrupt. isPausable	O	If the sequence is pausable by the bound power sequences client, this element SHALL be present and set to true. Otherwise it SHALL be omitted.
alternatives. powerSequence. operatingConstraintsInterrupt. isStoppable	O	If the sequence is stoppable by the bound power sequences client, this element SHALL be present and set to true. Otherwise it SHALL be omitted.
alternatives. powerSequence. operatingConstraintsInterrupt. maxCyclesPerDay	O	
alternatives. powerSequence. operatingConstraintsDuration	O	The element SHALL ONLY be present if at least one of its child elements is present. Otherwise it SHALL be omitted.
alternatives. powerSequence. operatingConstraintsDuration. activeDurationMin	O	
alternatives. powerSequence. operatingConstraintsDuration. activeDurationMax	O	
alternatives. powerSequence. operatingConstraintsDuration. pauseDurationMin	O	
alternatives. powerSequence. operatingConstraintsDuration. pauseDurationMax	O	
alternatives. powerSequence. operatingConstraintsDuration. activeDurationSumMin	O	
alternatives. powerSequence. operatingConstraintsDuration. activeDurationSumMax	O	
alternatives. powerSequence. operatingConstraintsResumeI mplication	O	The element SHALL ONLY be present if at least one of its child elements is present. Otherwise it SHALL be omitted.

alternatives. powerSequence. operatingConstraintsResumeImplication. resumeEnergyEstimated	O	
alternatives. powerSequence. operatingConstraintsResumeImplication. energyUnit	O	For electric energy, this element SHALL be omitted and Wh SHALL be assumed as unit for the value implicitly. For any other energy type, the unit SHALL be stated explicitly.
alternatives. powerSequence. operatingConstraintsResumeImplication. resumeCostEstimated	O	
alternatives. powerSequence. operatingConstraintsResumeImplication. currency	O	
alternatives. powerSequence. powerTimeSlot	M	
alternatives. powerSequence. powerTimeSlot. schedule	M	
alternatives. powerSequence. powerTimeSlot. schedule. slotNumber	M	
alternatives. powerSequence. powerTimeSlot. schedule. timePeriod	O	SHALL ONLY be present if at least one of its child elements is present.
alternatives. powerSequence. powerTimeSlot. schedule. timePeriod. startTime	O	
alternatives. powerSequence. powerTimeSlot. schedule. timePeriod. endTime	O	The following equation SHALL apply: endTime - startTime = defaultDuration.
alternatives. powerSequence. powerTimeSlot. schedule. defaultDuration	O	SHALL be present in case of “determined slot” and state the duration of the slot. If the slot has a configurable length, this element SHALL reflect the currently configured length.
alternatives. powerSequence. powerTimeSlot. schedule. durationUncertainty	O	May only be used if alternatives. powerSequence. powerTimeSlot. schedule. defaultDuration is set. Otherwise it SHALL be omitted.

alternatives. powerSequence. powerTimeSlot. schedule. slotActivated	O	If the slot is optional this element SHALL be present and reflect the current status of the slot. If the slot is not optional, this element SHALL be absent.
alternatives. powerSequence. powerTimeSlot. schedule. description	O	Default rules of section 3.10.1.3.1 SHALL be applied.
alternatives. powerSequence. powerTimeSlot. valueList	M	
alternatives. powerSequence. powerTimeSlot. valueList. value	M (1..6)	The following restriction to the list size applies: 1..6 occurrences of “value” are permitted. If more than just one “value” in “valueList” is present, each “value” in “valueList” SHALL state a different valueType than each other “value” in “valueList” of this slot. See section 4.3.24.3.2.5 for details about allowed values.
alternatives. powerSequence. powerTimeSlot. valueList. value. valueType	M	
alternatives. powerSequence. powerTimeSlot. valueList. value. value	M	
alternatives. powerSequence. powerTimeSlot. scheduleConstraints	O	If a slot has information about constraints for a time slot, this element and the required sub-elements SHALL be present. Otherwise AND for fixed forecasts (i.e. sequences without any flexibility), this element SHALL be omitted. Note: In case of a power sequence with repetition of its sequence of slots, this element often applies to the first execution only! Details are explained in the description of each child element.
alternatives. powerSequence. powerTimeSlot. scheduleConstraints. earliestStartTime	O	Only “xs:duration” value types SHALL be used to denote a relative time which relates to “now” as time 0. Note: This element applies to the first execution of the slot number only!
alternatives. powerSequence. powerTimeSlot. scheduleConstraints. latestEndTime	O	Only “xs:duration” value types SHALL be used to denote a relative time which relates to “now” as time 0.

		Note: This element applies to the first execution of the slot number only!
alternatives. powerSequence. powerTimeSlot. scheduleConstraints. minDuration	O	If the slot has a configurable duration, this element SHALL be present and denote the minimum supported configuration. Please also see section 4.3.24.3.2.7. Note: This element applies to every execution (repetition) of the slot number.
alternatives. powerSequence. powerTimeSlot. scheduleConstraints. maxDuration	O	If the slot has a configurable duration, this element SHALL be present and denote the maximum supported configuration. Please also see section 4.3.24.3.2.7. Note: This element applies to every execution (repetition) of the slot number.
alternatives. powerSequence. powerTimeSlot. scheduleConstraints. optionalSlot	O	SHALL be present and set to “true” if the slot can be omitted. Otherwise the element SHALL be omitted or set to “false”. I.e. the value “false” SHALL be assumed implicitly if the element is omitted. Note: This element applies to every execution (repetition) of the slot number.

Table 114: Description of complex class function “smartEnergyManagementPsData” for powerSequences information

4.3.24.4.1.2 When received

If a power sequences server receives this message, it SHALL consider this as erroneous receipt.

4.3.24.4.1.3 When to send

A power sequences server SHALL produce this message with a “notify” classifier and send it to the subscribed nodes if the data for one of its power sequences on this server instance changes.

A power sequences server SHALL produce a proper message with a “reply” classifier upon a received “read” request according to the rules described in section 4.3.24.4.2. This reply may then be a complete or restricted function exchange.

4.3.24.4.1.4 Error considerations

See section 4.3.24.4.1.2.

4.3.24.4.2 powerSequences information request

4.3.24.4.2.1 General data model description and rules

A power sequences information request is a request for the power sequences information of a power sequences server.

A power sequences server SHALL answer to power sequences information requests, even if most of the power sequences communication relies on event based (unsolicited) transmissions.

The request to get power sequences information is sent with “read” classifier and an “empty” complex class function “smartEnergyManagementPsData”.

Optionally, the request MAY be modelled for “restricted function exchange” using a dedicated selectors. This version of the specification RECOMMENDS to support the request for information of a specific power sequence as shown in Table 115.

Element name	**M/O/C (1.4.7)**	**Explanation**
smartEnergyManagementPsDataSelectors	O	SHALL ONLY be present if a child element is required. Otherwise it SHALL be absent.
smartEnergyManagementPsDataSelectors . alternativesRelation	-	Not considered in this recommendation.
smartEnergyManagementPsDataSelectors . alternativesRelation. alternativesId	-	Not considered in this recommendation.
smartEnergyManagementPsDataSelectors . powerSequenceDescription	M	SHALL ONLY be present if the following child element is required. Otherwise it SHALL be absent.
smartEnergyManagementPsDataSelectors . powerSequenceDescription. sequenceId	M	SHALL be present and set to the sequenceId of the power sequence whose information is requested.
smartEnergyManagementPsDataSelectors . powerTimeSlotSchedule	-	Not considered in this recommendation.
smartEnergyManagementPsDataSelectors . powerTimeSlotSchedule. slotNumber	-	Not considered in this recommendation.

Table 115: Dedicated "selectors" of “smartEnergyManagementPsData” to request a specific powerSequences information.

4.3.24.4.2.2 When received

If a power sequences server receives a power sequences information request with a “read” classifier, it SHALL answer with a proper “power sequence information”.

If the request is modelled for “restricted function exchange” as shown in Table 115, a reply with “smartEnergyManagementPsData” (as described in section 4.3.24.4.1) is modified as follows: The response SHALL only contain information of the power sequence with the same sequenceId value of the “selectors” content.

4.3.24.4.2.3 When to send

A power sequences server SHALL never produce this message.

4.3.24.4.2.4 Error considerations

If the power sequences server gets a power sequence request containing an invalid sequenceId (a sequenceId which does not exist at the time of the request on this instance of the power sequences server), the power sequences server SHALL consider this as erroneous receipt.

4.3.24.4.3 powerSequences configuration request call

4.3.24.4.3.1 General data model description and rules

This function can be sent from a power sequences server to its bound power sequences client if it wants one or "all" of its power sequences to be configured. It is intended as "trigger" for the bound power sequences client that may then initiate a power sequences configuration to configure the power sequences server's power sequences (see 4.3.24.4.4).

The server's message uses the complex function "smartEnergyManagementPsConfigurationRequestCall" of the complex class "SmartEnergyManagementPs" with the following restrictions and rules:

Element name	**M/O/C (1.4.7)**	**Explanation**
scheduleConfigurationRequest	M	
scheduleConfigurationRequest . sequenceId	O	If present, denotes the sequence ID of the power sequences server which SHOULD be configured. More precisely, it denotes the current power sequence preference of a single alternatives group. If "sequenceId" is not present, "all" power sequences of the power sequences server SHOULD be configured (more precisely: from each alternatives group a configuration of a proper power sequence should be done, if feasible).

Table 116: Description of complex class function "smartEnergyManagementPsConfigurationRequestCall" for powerSequences configuration request call

4.3.24.4.3.2 When received

If a power sequences server receives this message, it SHALL consider this as erroneous receipt.

4.3.24.4.3.3 When to send

A power sequences server SHALL only send this message with a "call" classifier to its bound power sequences client. It SHALL never send this message to clients that are only subscribed.

Once a power sequences server has a bound power sequences client and at least one of the power sequences contains flexibilities, it MAY send a power sequences configuration request call. If none of

the power sequences contains flexibilities, it SHALL NOT send a power sequences configuration request call.

A power sequences server MAY send this message containing a “sequenceId” to explicitly trigger the configuration of a specific power sequence or any of the (permissive) sequences of the same alternatives group. However, this SHALL NOT be done for fixed forecasts.

A power sequences server MAY also omit the “sequenceId” in the message. This denotes the request to get a configuration for each alternatives group except for those with fixed forecast.

4.3.24.4.3.4 Error considerations

If a power sequences server encounters a failure with its transmission of the power sequences configuration request call (e.g. explicitly from a received error message, or from any internal transmission API), the power sequences server SHOULD consider that the bound power sequences client may currently be not reachable or able to process the information, hence the client will not send a configuration. The absence of a power sequences configuration from the client SHALL NOT be considered as a problem, because the power sequences client may need more time for the calculation or needs to collect some further information beforehand.

If the bound power sequences client has not sent a power sequences configuration within 15 minutes, the power sequences server MAY consider to re-announce its power sequences and do a power sequences configuration request call or run as it would do in stand-alone mode, if it lacks time to wait for a configuration.

However, in which way a power sequences server handles errors is up to the manufacturer. For a good user experience, a power sequences server SHOULD, however, not behave in a different way than it would do in stand-alone mode. A washing machine SHOULD consider starting a cycle with a configuration defined on its own, while a battery storage MAY consider to idle or execute a maintenance cycle, e.g.

4.3.24.4.4 powerSequences configuration

4.3.24.4.4.1 General data model description and rules

The power sequences configuration is for use with a “write” classifier together with proper restriction information from the power sequences client to a bound power sequences server only.

A power sequences server may hold more than one power sequence with configurability at a time. However, a power sequences client SHALL only configure one power sequence within one command, i.e. configuring one power sequence after the other.

If the power sequences server offered multiple power sequences which were mutual alternatives, a configuration of one of these power sequences SHALL be interpreted as a choice for this specific power sequence. A consecutive configuration of another of the alternative power sequences SHALL be interpreted as a superseding choice between the mutual alternative power sequences.

There are 3 possibilities for configuring a power sequence:

- Configuration option A: By configuring the start time of a whole sequence
  - If a power sequences server uses this method, it SHALL start the sequence at the given time and follow its own preference of the slot durations.
  - MAY only be used by the power sequences client to configure the bound power sequences server if the durations of all slots are already set (by the server as its own preference). Otherwise the client SHALL NOT use this configuration method.
- Configuration option B: By configuring the end time of a whole sequence
  - If a power sequences server uses this method, it SHALL calculate the start time of the sequence by subtracting the expected durations of all slots (including all repetitions, if any) from the configured end time. Then, the server SHALL start the sequence at the calculated time and follow its own preference of the slot durations.
    Please note that the sum of expected durations SHOULD consider slot duration uncertainties in order to finish not later than the configured endTime.
  - MAY only be used by the power sequences client to configure the bound power sequences server if the durations of all slots are already set (by the server as its own preference). Otherwise the client SHALL NOT use this configuration method.
- Configuration option C: By configuring all configurable slots of a sequence
  - If a power sequences server uses this method, it SHALL set the durations of all slots according to the given configuration, if all constraints are meeting the server's requirements.
  - If a power sequences client wants to use this method in its configuration message, it SHALL retain the order of the slot numbers it received from the power sequences server message with power sequences information. Additionally, at least the start time or the end time of the sequence SHALL be configured (since only durations are included within the slots).
  - Please also note that this is the only method to re-configure a power-sequence that has already begun (i.e. at least the first slot has been started), provided that this power sequence still offers flexibilities.
  - Configurable slots are:
    - Slots whose duration can be changed
    - Slots that are marked as optional (and hence can be set active or inactive)
    - Slots with a combination of the above stated configurabilities

Note: This section is just about the configuration of a sequence, not about a "state change": Apart from a configuration a power sequences server's active power sequence may also be modified by a state change (provided that this functionality is enabled). This is described in 4.3.24.4.5.

The configuration message is modelled with a specific "write" operation to the power sequences server's instance of "smartEnergyManagementPsData" (i.e. the server's powerSequences information as described in section 4.3.24.4.1). The message SHALL be composed as follows:

1. The classifier SHALL be set to "write".
2. The destination of the message SHALL be the power sequences server's instance of the complex function "smartEnergyManagementPsData".
3. The message SHALL be modelled for "restricted function exchange" (this also means the "payload" element "cmd.function" SHALL be set properly; in this case it SHALL be "smartEnergyManagementPsData").

4. The “cmdControl” SHALL contain “partial”.
5. A (curtailed) function smartEnergyManagementPsData” SHALL be given, containing only the data to modify. The content is defined by Table 117.

Element name	**M/O/C (1.4.7)**	**Explanation**
alternatives	M (1..1)	The following restriction to the list size applies: Exactly one occurrence of “alternatives” is permitted.
alternatives. powerSequence	M (1..1)	The following restriction to the list size applies: Exactly one occurrence of “powerSequence” is permitted within its parent element.
alternatives. powerSequence. description	M	
alternatives. powerSequence. description. sequenceId	M	States the sequenceId for this configuration.
alternatives. powerSequence. schedule	O	Its presence depends on its child elements.
alternatives. powerSequence. schedule. startTime	See right, (*1).	If used, SHALL state the relative start time of the whole power sequence. If used with configuration option A, elements alternatives. powerSequence. schedule. endTime and element alternatives. powerSequence. powerTimeSlot shall be omitted, since the whole power sequence is configured. (*1) Element presence: For configuration option A: M B: NV C: If endTime is used: O; Otherwise: M
alternatives. powerSequence. schedule. endTime	See right, (*1).	If used, SHALL state the relative end time of the whole power sequence. If used with configuration option B, elements alternatives. powerSequence. schedule. startTime and alternatives.

		powerSequence. powerTimeSlot SHALL be omitted, since the whole power sequence is configured. (*1) Element presence: For configuration option A: NV B: M C: If startTime is used: O; Otherwise: M
alternatives. powerSequence. powerTimeSlot	See right, (*1).	List of power time slot schedule configuration data. (*1) Element presence: For configuration option A: NV B: NV C: M
alternatives. powerSequence. powerTimeSlot. schedule	See right, (*1).	The power time slot schedule configuration of a slot. (*1) Element presence: For configuration option A: NV B: NV C: M
alternatives. powerSequence. powerTimeSlot. schedule. slotNumber	M	States the slot number for which this configuration applies.
alternatives. powerSequence. powerTimeSlot. schedule. defaultDuration	O	SHALL be present if the slot has a configurable length AND the slot duration shall be changed to this new value.

alternatives. powerSequence. powerTimeSlot. schedule. slotActivated	O	If the optionality of the slot is configurable and its activation status shall be changed to the new value, this element SHALL be present and denote whether the slot shall be used (slotActivated = “true”) or not (slotActivated = “false”). Otherwise is SHALL be absent (i.e. the slot is mandatory).

Table 117: Description of “restricted write” operation for powerSequences configuration.

4.3.24.4.4.2 When received

A power sequences server SHALL only process power sequences configuration data received with a “write” classifier and proper restriction elements. Otherwise, a power sequences server SHALL consider this as erroneous receipt.

A power sequences server SHALL only accept power sequences configuration data that fit (with an "acceptable tolerance" as explained below) to the constraints of the according sequenceId. A power sequences server SHALL only accept power sequences configuration that configure all flexibilities that are available (durations, etc.) offered in one “write” operation for one power sequence.

If a power sequences server receives a power sequences configuration data with a configuration this power sequence cannot fulfil, it SHALL consider this as erroneous receipt and – if an acknowledgement message is to be sent – respond with a negative acknowledgement message with error number “7” (command rejected).

On "acceptable tolerance": The transmission of a configuration may be delayed "a bit too much" due to network problems, e.g. In this case the time configuration may violate the constraints given by the server "a bit". In such a case, the server SHOULD tolerate and adjust the received configuration silently as long as the violation is negligible. In general a tolerance of 30 seconds SHOULD be applied on time constraints.

Note: It may happen that a received power sequences configuration is valid according to the rules stated above but leads to no change at all of the sequence schedule (e.g. because received start/end times are identical to the schedule the server already announced). This SHALL NOT be considered an error.

4.3.24.4.4.3 When to send

A power sequences server SHALL never produce this message.

4.3.24.4.4.4 Error considerations

If the power sequences server gets a power sequence configuration containing an invalid sequenceId (a sequenceId which does not exist on this instance of the power sequences server), the power sequences server SHALL consider this as erroneous receipt.

See section 4.3.24.4.3.4 for more error considerations.

4.3.24.4.5 powerSequences state change

4.3.24.4.5.1 General data model description and rules

4.3.24.4.5.1.1 General aspects

The normal procedure for a power sequences client to control a bound power sequences server's power sequences is to initiate a powerSequences configuration as described in section 4.3.24.4.4. In some cases, a more direct way to control power sequences is needed. For this the “powerSequences state change” can be used. It enables a bound power sequences client to request a change of the state of a power sequences server's power sequence to "run", "pause" or "invalid" (i.e. terminate), dependent on the current state of the power sequence and some flags (see section 4.3.24.4.5.1.2 for details about the rules). This is based upon the definitions of section 4.3.24.3.3.6, among others.

Please note that state changes may occur from other situations as well. Among others, the state of a currently “chosen sequence” changes if another sequence is configured to become a “chosen sequence”. In this case, the previously “chosen sequence” may get the state “inactive” (e.g.). However, this is an “indirect” state change which is caused by (server) internal mechanisms according to this specification’s rules. Such “indirect” state changes are NOT subject of this section.

4.3.24.4.5.1.2 Rules for changing the state

The state of a power sequences server's power sequence MAY be changed by the bound power sequence client if the following aspects are considered:

1. If the power sequence is in state scheduled AND the flag isPausable is set to true AND the flag sequenceRemoteControllable is set to true: The client then MAY request for the state change:
    - scheduledPaused
2. If the power sequence is in state scheduledPaused AND the flag isPausable is set to true AND the flag sequenceRemoteControllable is set to true: The client then MAY request for the state change:
    - scheduled
3. If the power sequence is in state running AND the flag isPausable is set to true AND the flag sequenceRemoteControllable is set to true: The client then MAY request for the state change:
    - paused
4. If the conditions in item 3 are met and additionally the flag isStoppable is set to true, the client MAY request for the state changes:
    - paused
    - invalid
5. If the power sequence is in state paused AND the flag isPausable is set to true AND the flag sequenceRemoteControllable is set to true, the client MAY request for the state change:
    - running
6. If the conditions in item 5 are met AND additionally the flag isStoppable is set to true, the client MAY request for the state changes:
    - running

- invalid

7. If the power sequence is in state running, paused, scheduled, scheduledPaused or inactive AND the flag isStoppable is set to true, the client MAY request for the state change:
   - invalid

In all other cases, a (direct) state change SHALL NOT be requested by the bound power sequences client.

4.3.24.4.5.1.3 Data model description and rules

The "powerSequences state change" message is modelled with a specific "write" operation to the power sequences server's instance of "smartEnergyManagementPsData" (i.e. the server's powerSequences information as described in section 4.3.24.4.1). The message SHALL be composed as follows:

1. The classifier SHALL be set to "write".
2. The destination of the message SHALL be the power sequences server's instance of the complex function "smartEnergyManagementPsData".
3. The message SHALL be modelled for "restricted function exchange" (this also means the "payload" element "cmd.function" SHALL be set properly; in this case it SHALL be "smartEnergyManagementPsData").
4. The "cmdControl" SHALL contain "partial".
5. A (curtailed) function "smartEnergyManagementPsData" SHALL be given, containing the data to modify. The content is defined by Table 118.

Element name	**M/O/C (1.4.7)**	**Explanation**
alternatives	M (1..1)	The following restriction to the list size applies: Exactly one occurrence of "alternatives" is permitted.
alternatives. powerSequence	M (1..1)	The following restriction to the list size applies: Exactly one occurrence of "powerSequence" is permitted within its parent element.
alternatives. powerSequence. description	M	
alternatives. powerSequence. description. sequenceId	M	States the sequenceId for this state change.
alternatives. powerSequence. state	M	
alternatives. powerSequence. state. state	M	This element SHALL always be present and denotes the state change. See section 4.3.24.4.5.1.2 for values permitted in this state change.

Table 118: Description of "restricted write" operation for powerSequences state change.

4.3.24.4.5.2 When received

If a power sequences server receives a powerSequences state change from its bound power sequences client, it SHALL verify whether this state change is allowed within the rules denoted in 4.3.24.4.5.1.2. If it is not allowed, the power sequences server SHALL consider this as erroneous receipt.

4.3.24.4.5.3 When to send

A power sequences server SHALL never generate this message.

4.3.24.4.5.4 Error considerations

See section 4.3.24.4.5.2.

4.3.24.4.6 powerSequences price information

A power sequences server MAY provide information on the cost of one or more of its power sequences. The support of this information is optional.

The complex function “smartEnergyManagementPsPriceData” of the complex class “SmartEnergyManagementPs” is used with the following restrictions and rules:

Element name	**M/O/C (1.4.7)**	**Explanation**
price	O	SHALL ONLY be present if proper child elements are present.
price. sequenceId	M	
price. price	M	
price. currency	M	

Table 119: Description of smartEnergyManagementPsPriceData for powerSequences price information.

4.3.24.4.6.1 When received

If a power sequences server receives this message with a “notify” or “reply” classifier, it SHALL consider this as erroneous receipt. For “write” classifier see section 4.3.24.4.8.

4.3.24.4.6.2 When to send

A power sequences server SHALL produce this message with a “notify” classifier and send it to subscribed nodes if the price information for one of its power sequences on this server instance changes.

A power sequences server SHALL produce a proper message with a “reply” classifier upon a received “read” request according to the rules described in section 4.3.24.4.7.2. This reply may then be a complete or restricted function exchange.

4.3.24.4.6.3 Error considerations

See section 4.3.24.4.7.2.

4.3.24.4.7 powerSequences price information request

4.3.24.4.7.1 General data model description and rules

A power sequences price information request is a request for the power sequences price information of a power sequences server.

A power sequences server SHALL answer to power sequences price information requests if the power sequences server support power sequences price information.

The request to get power sequences price information is sent with “read” classifier and an “empty” complex class function “smartEnergyManagementPsPriceData”.

Optionally, the request MAY be modelled for “restricted function exchange” using a dedicated selectors. This version of the specification RECOMMENDS to support the request for price information of a specific power sequence as shown in Table 120.

Element name	**M/O/C (1.4.7)**	**Explanation**
smartEnergyManagement PsPriceDataSelectors	O	SHALL ONLY be present if a child element is required. Otherwise it SHALL be absent.
smartEnergyManagement PsPriceDataSelectors. price	M	SHALL ONLY be present if the following child element is required. Otherwise it SHALL be absent.
smartEnergyManagement PsPriceDataSelectors. price. sequenceId	M	SHALL be present and set to the sequenceId of the power sequence whose price information is requested.

Table 120: Dedicated "selectors" of “smartEnergyManagementPsPriceData” to request a specific powerSequences price information.

4.3.24.4.7.2 When received

If a power sequences server receives a power sequences price information request with a “read” classifier, it SHALL answer with a proper “power sequence price information”.

If the request is modelled for “restricted function exchange” as shown in Table 120, a reply with “smartEnergyManagementPsPriceData” (as described in section 4.3.24.4.6) is modified as follows: The response SHALL only contain price information of the power sequence with the same sequenceId value of the “selectors” content.

4.3.24.4.7.3 When to send

A power sequences server SHALL never produce this message.

4.3.24.4.7.4 Error considerations

If the power sequences server gets a power sequence request with a sequenceId where no price information is available, the power sequences server SHALL consider this as erroneous receipt.

4.3.24.4.8 powerSequences price information change

4.3.24.4.8.1 General data model description and rules

4.3.24.4.8.1.1 General aspects

A power sequences price information change is usually only initiated upon a power sequences price calculation request call. I.e. once a powerSequences server asked a bound powerSequences client for a price calculation (see section 4.3.24.4.9), the server awaits receiving a powerSequences price information change from the client. This information should contain the price for the requested sequence. Of course, the powerSequences client is then obliged to consider the server's information on the current sequence schedule (smartEnergyManagementPsData).

Further rules apply:

A power sequences price information change can only be considered to reflect a whole sequence's price if the proper power sequence information is complete. Once a sequence started, a power sequences server may remove past slots from the power sequences information, resulting in incomplete sequence information. As a consequence, the request for the price calculation and the execution and delivery of the price calculation SHOULD occur before the power sequence starts as otherwise a bound power sequences client may not have sufficient information to calculate a price for the whole sequence.

Once the price for a whole sequence was calculated and transmitted successfully, it SHOULD also be possible to request and execute a price calculation after the sequence started. As mentioned above, this is only feasible if the power sequences server keeps past slots in its sequence information.

4.3.24.4.8.1.2 Data model description

The "powerSequences price information change" message is modelled with a specific "write" operation to the power sequences server's instance of "smartEnergyManagementPsPriceData" (i.e. the server's powerSequences price information as described in section 4.3.24.4.6). The message SHALL be composed as follows:

1. The classifier SHALL be set to "write".
2. The destination of the message SHALL be the power sequences server's instance of the complex function "smartEnergyManagementPsPriceData".

3. The message SHALL be modelled for “restricted function exchange” (this also means the “payload” element “cmd.function” SHALL be set properly; in this case it SHALL be “smartEnergyManagementPsPriceData”).
4. The “cmdControl” SHALL contain “partial”.
5. A (curtailed) function “smartEnergyManagementPsPriceData” SHALL be given, containing only the data to modify. The content is defined by Table 121.

Element name	**M/O/C (1.4.7)**	**Explanation**
price	M (1..1)	The following restriction to the list size applies: Exactly one occurrence of “price” is permitted.
price. sequenceId	M	The sequenceId whose price shall be modified.
price. price	M	The new price.
price. currency	M	The currency of the price.

Table 121: Description of “restricted write” operation for a power sequence price modification.

4.3.24.4.8.2 When received

If a power sequences server receives a power sequences price information change from a device that is NOT its bound power sequences client, it MAY consider this as erroneous receipt.

A power sequences server SHOULD ONLY accept a power sequences price information change if it was received from its bound power sequences client. Afterwards, further rules apply:

If a power sequences server receives a power sequences price information change as valid message upon its previous power sequence price calculation request, it SHALL evaluate the message accordingly. Afterwards, it SHOULD NOT accept subsequently received power sequences price information changes. Such “additional responses” may denote a price update information. However, instead of evaluating these “additional responses” the power sequences server SHOULD consider this as erroneous receipt, and it MAY initiate a new power sequences price calculation request.

If a power sequences server receives a power sequences price information change without having issued a power sequences price calculation request call before, it SHOULD consider this as erroneous receipt.

If a power sequences server receives a power sequences price information change containing a power sequenceId it has not asked for, it SHALL consider this as erroneous receipt.

Remark (informative): If a power sequences server receives a valid power sequences price information change, the use of this information is manufacturer specific. Examples of use are displaying it to a customer on a device display.

4.3.24.4.8.3 When to send

A power sequences server SHALL never generate this message.

4.3.24.4.8.4 Error considerations

See section 4.3.24.4.8.2.

4.3.24.4.9 powerSequences price calculation request call

4.3.24.4.9.1 General data model description and rules

A power sequences price calculation request call is a request from a power sequences server to a bound power sequences client, in order to ask the power sequences client to calculate a price for a certain sequence ID of the power sequences server. The answer is delivered by the power sequences client using the "power sequences price calculation response call" message. The sequence needs to have schedule information (see 4.3.24.3.3.3) and all slots need to be determined to enable the client to calculate a price for it. Furthermore, the request SHOULD be sent "in time" in order to receive the price calculation before the sequence starts. See section 4.3.24.4.8.1 for details!

The server's message uses the complex function "smartEnergyManagementPsPriceCalculationRequestCall" of the complex class "SmartEnergyManagementPs" with the following restrictions and rules:

Element name	**M/O/C (1.4.7)**	**Explanation**
priceCalculationRequest	M	
priceCalculationRequest. sequenceId	M	

Table 122: Description of complex class function "smartEnergyManagementPsPriceCalculationRequestCall" for powerSequences price calculation request call

4.3.24.4.9.2 When received

If a power sequences server receives this message, it SHALL consider this as erroneous receipt.

4.3.24.4.9.3 When to send

A power sequences server SHALL produce this message with a "call" classifier if it is interested in the price for a configured sequence it offers, i.e. a sequence which has a valid schedule (including information about all slot's durations as stated in 4.3.24.4.9.1). It will get a calculation response based on its offered sequence and the according schedule, provided that the bound client provides this service and has sufficient information. Please consider also rules described in section 4.3.24.4.8.1!

Note: A power sequences server SHALL never change the configuration for a power sequence that has already been configured by its bound power sequences client, just to calculate different prices.

4.3.24.4.9.4 Error considerations

See section 4.3.24.4.9.2.

Please note that calculating prices is an optional feature.

Additionally, if a device sends a power sequences price calculation request call to its bound power sequences client and the binding partner returns a negative acknowledgement message, this might happen due to the fact that the power sequences client temporarily cannot calculate prices, e.g. because the client has not yet been updated with valid prices for the whole period of the sequence. Thus, a power sequences server SHOULD be aware that it might encounter power sequences clients which currently cannot calculate prices. For this case, the server SHALL be able to cope with errors returned to price calculation requests it issues. How exactly it handles these errors is up to the manufacturer.

4.3.25 SupplyCondition

4.3.25.1 Dependencies to other Feature Types

None.

4.3.25.2 Introduction

Practice shows that the supply of electricity, water, etc. can be temporarily impaired. Some possible causes are announced maintenance work at the supply line, supply network overload, or depletion of a local energy resource (battery, e.g.). The Supply Condition concept permits indication of the supply condition by a household's SPINE data models.

The concept is less intended for any automatisms. It is rather intended for indications to users in order to prepare or apply measures in time.

4.3.25.3 Rules

None.

4.3.25.4 supplyConditionListData

4.3.25.4.1 Description

This function can be used to model the "events", i.e. the information on a specific supply condition. These events can, e.g., be sent from a DSO to a household. In this case, the household would be the data owner and the DSO would send (as a client) write commands to this resource.

4.3.25.4.2 Rules

None.

5038 *4.3.25.4.3 Element rules*

Element	Rule	Description
conditionId	SHALL be set as PRIMARY IDENTIFIER.	Identifier of the condition.
timestamp	SHALL be set as SUB IDENTIFIER to the timestamp of the creation of the message.	The timestamp of the event.
eventType	SHALL be set. Value rules of Table 124 SHALL be applied.	The kind of the event.
originator	SHOULD be set. Value rules of Table 125 SHALL be applied.	The originator (in terms of instance, not in terms of address) of the information.
thresholdId	MAY be set as FOREIGN IDENTIFIER to the according thresholdId, if related to a threshold.	The identifier of this event's threshold (if available).
thresholdPercentage	Reached percentage of the threshold value. Only if *thresholdId* is set, *thresholdPercentage* MAY be set, too.	Relation of current situation to (configured) threshold (if available).
relevantPeriod	None.	Esp. the announcement of maintenance can make use of this element in order to tell when a maintenance is planned.
description	Default rules of section 3.10.1.3.1 SHALL be applied.	Descriptive information on the event.
gridCondition	None.	Indicates the condition of the electricity grid.

5039 *Table 123: supplyConditionListData Element rules*

5040 Element "**eventType**" value rules:

Value	Rule	Description
thesholdExceeded	None.	The related threshold was exceeded (may be good or bad, see definition of the corresponding *thresholdType* element in section 5.3.26.4.2 for details). This means that the actual value is greater than the threshold.
fallenBelowThreshold	None.	The related threshold was fallen below (may be good or bad, see definition of the corresponding *thresholdType* element in section 5.3.26.4.2 for details). This means that the actual value is lower than the threshold.
supplyInterrupt	None.	The supply of the corresponding commodity was interrupted.
releaseOfLimitations	None.	"Back to normal", no limitations or threshold violations.

otherProblem	None.	Some problem with the supply without further details about the problem.

Table 124: Element "eventType" value rules

Element "**originator**" value rules:

Value	Rule	Description
externDSO	None.	The originator of the current condition "event" is the DSO (distribution system operator).
externSupplier	None.	The originator of the current condition "event" is the energy supplier.
internalLimit	None.	Some internal limit at the premises generated the current condition "event".
internalService	None.	Due to an internal service at the premises, the current condition "event" was generated.
internalUser	None.	A user interaction is the source of the current condition "event".

Table 125: Element "originator" value rules

Element "**gridCondition**" value rules:

Value	Rule	Description
consumptionRed	None.	The electricity grid condition for consumption from the grid is "red".
consumptionYellow	None.	The electricity grid condition for consumption from the grid is "yellow".
good	None.	The electricity grid condition is "good".
productionYellow	None.	The electricity grid condition for feeding into the grid is "yellow".
productionRed	None.	The electricity grid condition for feeding into the grid is "red".

Table 126: Element "gridCondition" value rules

4.3.25.5 supplyConditionDescriptionListData

4.3.25.5.1 Description

The more static information about a supply condition is modelled within this function.

4.3.25.5.2 Rules

None.

4.3.25.5.3 Element rules

Element	Rule	Description
conditionId	SHALL be set as PRIMARY IDENTIFIER.	Identifier of the condition.
commodityType	SHOULD be set.	The type of the commodity handled by this condition is stated here.
positiveEnergyDirection	SHOULD be set. The value "consume" expresses that positive values of energy or power denote	The *positiveEnergyDirection* states whether energy consumption or production will be counted as positive value.

	energy consumption by the premises. The value "produce" expresses that positive values of energy or power denote energy production by the premises.	
label	Default rules of section 3.10.1.2.1 SHALL be applied.	A short label of the condition.
description	Default rules of section 3.10.1.3.1 SHALL be applied.	Descriptive information on the condition.

Table 127: supplyConditionDescriptionListData Element rules

4.3.25.6 supplyConditionThresholdRelationListData

4.3.25.6.1 Description

Multiple thresholds can be associated to a condition. This leads to the relation

- 1 conditionId -> 0..m thresholdId

The supplyConditionThresholdRelationListData function can be used to model concrete relations.

4.3.25.6.2 Rules

None.

4.3.25.6.3 Element rules

Element	Rule	Description
conditionId	SHALL be set as PRIMARY IDENTIFIER.	Identifier of the condition.
thresholdId (list)	Multiple tresholdIds of thresholds linked to this condition MAY be provided.	Related threshold identifier(s).

Table 128: supplyConditionThresholdRelationListData Element rules

4.3.25.7 Descriptive information and further definitions

None.

4.3.26 TariffInformation

Reserved for future use.

4.3.27 TaskManagement

4.3.27.1 Dependencies to other Feature Types

- DirectControl

- HVAC
- LoadControl
- PowerSequences
- SmartEnergyManagementPs

4.3.27.2 Introduction

The TaskManagement Class lists all "tasks" that are currently relevant on an entity. This enables one to have a quick look on the tasks of related classes, but not for their modification. Further information has to be gathered with the corresponding classes supporting the specific functionality.

In this version of the specification, tasks can be "jobs" like in DirectControl, PowerSequences / SmartEnergyManagementPs, etc. (see the related function groups below).

Remark: Future versions of this specification may add other kinds of tasks (i.e. tasks that are not covered by “jobs”).

4.3.27.3 Rules

None.

4.3.27.4 taskManagementJobListData

4.3.27.4.1 Description

All jobs that are available on the entity are listed here, together with their state and (if available) the elapsed and remaining time.

4.3.27.4.2 Rules

None.

4.3.27.4.3 Element rules

Element	Rule	Description
jobId	SHALL be set as PRIMARY IDENTIFIER.	The *jobId* is used for indentifying different job information collected by this class.
timestamp	SHOULD be set to the timestamp of the creation of the message.	Point in time, this information was generated.
jobState	SHALL be set. Value rules of Table 50 SHALL be applied. Value rules of Table 70 SHALL be applied. Value rules of section 3.10.1.5.1 SHALL be applied.	Current state of the job.

elapsedTime	If set, this element SHALL contain the duration the job was already active.	The duration, this job was already active (normally, pause times are not counted).
remainingTime	If set, this element SHALL contain the duration the job still needs to be active.	The duration, this job will most likely be active before it ends.

Table 129: taskManagementJobListData Element rules

4.3.27.5 taskManagementJobRelationListData

4.3.27.5.1 Description

To identify where a job originates from, the *taskManagementJobRelationListData* function is used. If the related class allows more than one job the corresponding identifier(s) are denoted.

4.3.27.5.2 Rules

If this function is used, each list entry SHALL contain exactly one relation to a job. This means that it is not allowed to set more than one of the following Elements within one list entry:

- directControlRelated
- hvacRelated
- loadControlReleated
- powerSequencesRelated
- smartEnergyManagementPsRelated

4.3.27.5.3 Element rules

Element	Rule	Description
jobId	SHALL be set as PRIMARY IDENTIFIER.	The *jobId* is used for indentifying different job information collected by this class.
directControlRelated	None.	Flag that indicates that the job is related to DirectControl.
hvacRelated	None.	Flag that indicates that the job is related to HVAC.
hvacRelated. overrunId	If the job is related to an overrun, this element SHALL be set to the FOREIGN IDENTIFIER of the corresponding overrun.	Identifier for the related overrun.
loadControlReleated	None.	Flag that indicates that the job is related to LoadControl.
loadControlReleated. eventId	If the job is related to an event, this element SHALL be set to the FOREIGN IDENTIFIER of the corresponding event.	Identifier for the related load control event.
powerSequencesRelated	None.	Flag that indicates that the job is related to PowerSequences.
powerSequencesRelated. sequenceId	If the job is related to a power sequence, this element SHALL be	Identifier for the related power sequence.

	set to the FOREIGN IDENTIFIER of the corresponding power sequence.	
smartEnergyManagementPs Related	None.	Flag that indicates that the job is related to SmartEnergyManagementPs.
smartEnergyManagementPs Related. sequenceId	If the job is related to a SmartEnergyManagementPs sequence, this element SHALL be set to the FOREIGN IDENTIFIER of the corresponding sequence.	Identifier for the related power sequence.

Table 130: taskManagementJobRelationListData Element rules

4.3.27.6 taskManagementJobDescriptionListData

4.3.27.6.1 Description

The more static information about the jobs are communicated with this function.

4.3.27.6.2 Rules

None.

4.3.27.6.3 Element rules

Element	Rule	Description
jobId	SHALL be set as PRIMARY IDENTIFIER.	The *jobId* is used for indentifying different job information collected by this class.
jobSource	SHOULD be set. Value rules of Table 132 SHALL be applied.	Specifies what kind of reason caused this job.
label	Default rules of section 3.10.1.2.1 SHALL be applied.	A short label of the job.
description	Default rules of section 3.10.1.3.1 SHALL be applied.	Descriptive information on the job.

Table 131: taskManagementJobDescriptionListData Element rules

Element "**jobSource**" value rules:

Value	Rule	Description
internalMechanism	None.	The job was started due to some device internal mechanism (e.g. a regular cleaning cycle).
userInteraction	None.	The user activated some functionality directly via the UI of the device.
externalConfiguration	None.	The job started, because it was triggered from some external device (e.g. an energy manager).

Table 132: Element "jobSource" value rules

4.3.27.7 taskManagementOverviewData

4.3.27.7.1 Description

A general overview is given with this function. It specifies whether a remote controlling is permitted at that moment and if currently jobs are active.

4.3.27.7.2 Rules

None.

4.3.27.7.3 Element rules

Element	Rule	Description
remoteControllable	SHOULD be set. If set to "false", write operations on features linked via *taskManagementJobRelationListData* to this feature SHALL be declined. If absent, a default value of "true" SHALL be applied.	States if remote controlling of the device or entity is permitted.
jobsActive	None.	Denotes whether jobs are active on the device or entity or not.

Table 133: taskManagementOverviewData Element rules

4.3.27.8 Descriptive information and further definitions

The TaskManagement Feature Type enables the brief listing of all tasks that are currently relevant for the feature's entity. A special case is the implementation of the Feature Type *TaskManagement* on an entity with the entity type *DeviceInformation*: In this case the feature covers tasks of all entities that are available on this device.

4.3.28 Threshold

4.3.28.1 Dependencies to other Feature Types

None.

4.3.28.2 Introduction

Thresholds are helpful in several kind of use cases. E.g. a device or application can be configured to do some specific behaviour if a measured value is above or below a configured threshold. Some measurands need to stay in a certain value range, to ensure proper and safe working of devices. Automatic alarms after reaching a threshold help reacting on the occurrence immediately.

4.3.28.3 Rules

None.

5164

5165 ***4.3.28.4 thresholdListData***

5166 *4.3.28.4.1 Description*

5167 Each threshold (identified by its *thresholdId*) has a configured value. This function is used to read or
5168 write the actual value.

5169

5170 *4.3.28.4.2 Rules*

5171 None.

5172

5173 *4.3.28.4.3 Element rules*

Element	Rule	Description
thresholdId	SHALL be set as PRIMARY IDENTIFIER.	Identifier for the threshold.
thresholdValue	SHALL be set.	Value that should not be fallen below or exceeded (depends on the *thresholdType*, see section 5.3.26.4.2).

5174 *Table 134: thresholdListData Element rules*

5175

5176 ***4.3.28.5 thresholdConstraintsListData***

5177 *4.3.28.5.1 Description*

5178 The value range from which a threshold value configuration may be choosen from may be restricted
5179 by a minimum and a maximum value. Between these, not every value, but only ones with a specific
5180 gap, may be used, specified with the step size.

5181

5182 *4.3.28.5.2 Rules*

5183 If thresholds are writeable, this function SHOULD be supported, otherwise it MAY be supported.

5184

5185 *4.3.28.5.3 Element rules*

Element	Rule	Description
thresholdId	SHALL be set as PRIMARY IDENTIFIER.	Identifier for the threshold.
thresholdRangeMin	If set, the threshold value SHALL NOT be set to a value below this minimum value.	Minumum value that may be used to configure the element *thresholdValue* (see section 5.3.26.2.2).
thresholdRangeMax	If set, the threshold value SHALL NOT be set to a value above this maximum value.	Maximum value that may be used to configure the element *thresholdValue* (see section 5.3.26.2.2).
thresholdStepSize	The threshold value SHALL ONLY contain multiples of thresholdStepSize.	Step size for the configuration of the element *thresholdValue* (see section 5.3.26.2.2).

5186 *Table 135: thresholdConstraintsListData Element rules*

5187

5188 ***4.3.28.6 thresholdDescriptionListData***

5189 *4.3.28.6.1 Description*

5190 The rather static information about thresholds is defined within this function.

5191

5192 *4.3.28.6.2 Rules*

5193 None.

5194

5195 *4.3.28.6.3 Element rules*

Element	Rule	Description
thresholdId	SHALL be set as PRIMARY IDENTIFIER.	Identifier for the threshold.
thresholdType	SHALL be set.	The type of the treshold.
unit	SHALL be set, if there could be ambiguity how to interpret the threshold without the unit.	The unit in which the value of the threshold is denoted.
scopeType	None.	Specifies a more detailed meaning of the threshold.
label	Default rules of section 3.10.1.2.1 SHALL be applied.	A short label of the threshold.
description	Default rules of section 3.10.1.3.1 SHALL be applied.	Descriptive information on the threshold.

5196 *Table 136: thresholdDescriptionListData Element rules*

5197 Element "**thresholdType**" value rules:

Value	Rule	Description
goodAbove	None.	A positive reaction will follow after the value is greater than the threshold.
badAbove	None.	A negative reaction will follow after the value is greater than the threshold.
goodBelow	None.	A positive reaction will follow after the value is lower than or equal to the threshold.
badBelow	None.	A negative reaction will follow after the value is lower than or equal to the threshold.
minValueThreshold	None.	If the value is lower than this minimum threshold, some warning will be activated.
maxValueThreshold	None.	If the value is greater than this maximum threshold, some warning will be activated.
minValueThresholdExtreme	None.	If the value is lower than this "extreme minimum" threshold, some important warning will be activated.
maxValueThresholdExtreme	None.	If the value is greater than this "extreme maximum" threshold, some important warning will be activated.
sagThreshold	None.	If this sag threshold is underrun, some warning will be activated. Commonly used for electricity voltage sags.

swellThreshold	None.	If this swell threshold is exceeded, some warning will be activated. Commonly used for electricity voltage swells.

Table 137: Element "thresholdType" value rules

Note: All threshold rules SHALL be understood in a mathematical way.

4.3.28.7 Descriptive information and further definitions

None.

4.3.29 TimeInformation

4.3.29.1 Dependencies to other Feature Types

None.

4.3.29.2 Introduction

The exchange of time information is important for most smart grid and home automation systems. Please note the underlying revision of the SPINE data model does not specify a sophisticated synchronisation mechanism that includes a given precision. This is due to the complex of problems appearing with communication delays in the different technologies and hardware platforms. However, an application could measure all relevant delays and take care of a real synchronisation in the network using the *TimeInformation* class.

Independent of any synchronisation aspect the "owner concept" applies here as well: The "owner" of any given data (a measured value of a sensor, e.g.) applies its own SPINE time to its own data. This is relevant for the case of a substantial difference between the SPINE time of two nodes.

In order to avoid problems from such differences a typical sequence could be that a new device sends a *timeDistributorEnquiryCall* (with the *isTimeDistributor* element set to true) to all devices in the SPINE network. All time distributors would send a *timeDistributorData* instance as notification to the device. The new device would pick the time distributor with the highest priority and send a read command to the *timeInformationData* function of this distributor. The time distributor would reply with a *timeInformationData* containing its current time.

4.3.29.3 Rules

None.

4.3.29.4 timeInformationData

4.3.29.4.1 Description

The *timeInformationData* command is used for exchanging the SPINE time.

5232 *4.3.29.4.2 Rules*
5233 None.

5234

5235 *4.3.29.4.3 Element rules*

Element	Rule	Description
utc	SHALL be set if *isTimeDistributor* is set "true". Otherwise it SHOULD be set.	The current time in UTC.
utcOffset	None.	The configured offset to UTC (e.g. PT2H [= +2 hours] for Central European Summer Time (CEST)).
dayOfWeek	None.	To model the actual day of week (Monday to Sunday) this element is used. It refers to the local time. If there is no utcOffset given, it is assumed, that the local time is equivalent to UTC-time.
calendarWeek	None.	A value between 1 and 53 stating the current calendar week number. This element refers to the local time. If there is no utcOffset given, it is assumed, that the local time is equivalent to UTC-time.

5236 *Table 138: timeInformationData Element rules*

5237

5238 ***4.3.29.5 timeDistributorData***

5239 *4.3.29.5.1 Description*
5240 One or some devices in a SPINE network should act as time distributor. The information thereof and
5241 the dedicated priority can be modelled with the *timeDistributorData* command.

5242

5243 *4.3.29.5.2 Rules*
5244 None.

5245

5246 *4.3.29.5.3 Element rules*

Element	Rule	Description
isTimeDistributor	If set to "true" the device SHALL be able to act as time distributor. Otherwise it SHALL be omitted or set to "false".	Denotes whether a device can act as a time distributor in the SPINE network or not.
distributorPriority	If *isTimeDistributor* is set to "false", this element SHOULD be absent. The time distributor with the highest priority SHOULD be used for time synchronization. There MAY be further time distributors with lower priority to have redundant time distributors that can be used as soon	A (configured) priority of the time distributor.

	as the time distributor with the higher priority can currently not be used. Only values greater than zero SHALL be used. A value of "1" SHALL be interpreted as highest priority. The priority decreases with increasing value of distributorPriority.	

Table 139: timeDistributorData Element rules

4.3.29.6 timePrecisionData

4.3.29.6.1 Description

Information regarding the precision of the time of a device can be modelled using *timePrecisionData*.

4.3.29.6.2 Rules

The element “isSynchronized” SHALL be used with regards to the time distributor with the highest priority. Time should not differ more than 10 seconds from the time retrieved from a commonly used NTP server in the internet (e.g. ptbtime1.ptb.de). If a device has a *clockDrift* element set to 1, it should at least get a sync every 10 days. After that time, it can not be sure to have a correct time information and therefore sets "isSynchronised" to false.

4.3.29.6.3 Element rules

Element	Rule	Description
isSynchronised	SHOULD be set. If omitted, isSynchronized SHALL be interpreted as "false".	This is a Boolean value, stating whether the device's SPINE time information is synchronized with the SPINE network or not, meaning that it could receive the SPINE time from a time distributor and has still a “sufficiently precise time”.
lastSyncAt	MAY be used to communicate when the last synchronization occurred.	This is the timestamp of the last synchronisation with a time distributor, or, in case of a time distributor itself, the timestamp of the last official date and time information (e.g. from an NTP server).
clockDrift	MAY be used to communicate the clock drift.	The drift of the internal clock (in seconds per day) is indicated by this element.

Table 140: timePrecisionData Element rules

4.3.29.7 timeDistributorEnquiryCall

4.3.29.7.1 Description

This function can be used for finding all time distributors in the network. Therefore, it can be sent to all devices in the network that have their *isTimeDistributor* element in the function *timeDistributor* (see section 5.3.27.3) set to true.

5268

5269 *4.3.29.7.2 Rules*
5270 Devices that do not support the function *timeDistributor* SHOULD NOT be used as destination of this
5271 call.

5272

5273 *4.3.29.7.3 Element rules*
5274 None.

5275

5276 ***4.3.29.8 Descriptive information and further definitions***
5277 None.

5278

5279 **4.3.30 TimeSeries**

5280 ***4.3.30.1 Dependencies to other Feature Types***
5281 None.

5282

5283 ***4.3.30.2 Introduction***
5284 TimeSeries is a versatile class that allows to express any kind of time dependent values as a time
5285 series of values. It can be used to express electricity related values like power constraints or power
5286 consumption over time, but also other values that are not electricity related like temperature or
5287 humidity over time.

5288

5289 ***4.3.30.3 Rules***
5290 None.

5291

5292 ***4.3.30.4 timeSeriesListData***

5293 *4.3.30.4.1 Description*
5294 The timeSeriesListData function is used to describe the value of a timeSeries over time.

5295

5296 *4.3.30.4.2 Rules*
5297 None.

5298

5299 *4.3.30.4.3 Element rules*

Element	Rule	Description

timeSeriesId	SHALL be set as PRIMARY IDENTIFIER.	Allows the identification of a timeSeries. Allows linking of the different functions to the same timeSeries. Allows linking of the timeSeries within other Features that are placed in the same entity.
timePeriod	If set, neither timeSeriesSlot.duration nor timeSeriesSlot.timePeriod SHALL define times out of the bound of the overall timeSeries timePeriod. Only relative times SHALL be used (xs:duration). Relative times SHALL be interpreted relative to “now”.	Time period of the complete timeSeries.
timeSeriesSlot (list)	A timeSeries SHOULD have at least one timeSeriesSlot.	Allows to define slots of a timeSeries.
timeSeriesSlot. timeSeriesSlotId	SHALL be set as SUB IDENTIFIER, if timeSeriesData contains more than one timeSeriesSlot entry. In case of more than one timeSeriesSlot entry, timeSeriesSlotId SHALL increase matching with the chronological order of the timeSeriesSlots. If timeSeriesSlotId is omitted it SHALL be interpreted as zero.	Allows identification of a timeSeriesSlot within a timeSeries.
timeSeriesSlot .timePeriod	timeSeriesSlot .timePeriod .startTime SHALL be used in slots that have a time gap to the previous slot. timeSeriesSlot .timePeriod .startTime SHALL be used in the first slot if no startTime is provided in the timePeriod element with the overall time period of the timeSeries. Only relative times SHALL be used (xs:duration). Relative times SHALL be interpreted relative to “now”.	Allows to define a timePeriod of a timeSeriesSlot and to model time gaps inbetween slots.
timeSeriesSlot. duration	SHALL be set. SHALL only contain values greater than zero seconds. If timeSeriesSlot. timePeriod. startTime ist set, the duration starts from there. If timeSeriesSlot. timePeriod. startTime is not set in the first timeSeriesSlot (slot with the lowest timeSeriesSlotId), the timePeriod. startTime of the overall timeSeries SHALL be set and the duration	Allows to define a duration of a timeSeriesSlot.

	of the first timeSeriesSlot starts from there. If timeSeriesSlot. timePeriod. startTime is not set in a slot that is not the first slot, the duration starts after the previous slot is finished.	
timeSeriesSlot. recurenceInformation	Only relative times SHALL be used (xs:duration) within reccurenceInformation. Relative times SHALL be interpreted relative to “now”.	If the time series slot is a recurring one, the corresponding information can be found within this element and its sub-elements.
timeSeriesSlot. value	A positive value SHALL relate to consumption and a negative value SHALL relate to production of the server. If minValue is set the content of value SHALL be greater. If maxValue is set the content of value SHALL be smaller.	Defines the expected value during a timeSeriesSlot.
timeSeriesSlot. minValue	A positive value SHALL relate to consumption and a negative value SHALL relate to production of the server. If value or maxValue is set the content of minValue SHALL be smaller.	Defines a lower value limit
timeSeriesSlot. maxValue	A positive value SHALL relate to consumption and a negative value SHALL relate to production of the server. If value or minValue is set the content of maxValue SHALL be greater.	Defines a upper value limit

Table 141: timeSeriesListData Element rules

4.3.30.5 timeSeriesDescriptionListData

4.3.30.5.1 Description

The timeSeriesDescriptionListData function is used to describe a timeSeries.

4.3.30.5.2 Rules

This is the first data that SHOULD initially be read of a timeSeries. If this data matches with the data that the client is expecting, the other functions of timeSeries become relevant.

4.3.30.5.3 Element rules

Element	Rule	Description
timeSeriesId	SHALL be set as PRIMARY IDENTIFIER.	Allows the identification of a timeSeries. Allows linking of the different functions to the same

		timeSeries. Allows linking of the timeSeries within other Features that are placed in the same entity.
timeSeriesType	Value rules of Table 143 SHALL be applied.	Allows to define which timeSeries type is supported.
timeSeriesWriteable	If set to "true", this timeSeriesData SHALL be writeable. If set to false or omitted, this timeSeriesData SHALL NOT be writeable.	Allows to define if the timeSeries is writeable.
updateRequired	With updateRequired the server can request an update of writeable or changeable data related to the same PRIMARY IDENTIFIER from a client. The server SHALL ensure that only one responsible client is able to update the related data. To request an update the server SHALL set updateRequired to “true”. Note: In this case, the server expects the responsible client to update the writeable or changeable data related to the same PRIMARY IDENTIFIER. However, also if updateRequired is set to “false” a server SHOULD in general allow updates of the data from the responsible client. The server SHALL set the updateRequired back to “false”, as soon as “timeSeriesDescriptionListData” was updated successfully (if writeable or changeable) OR the update of the other writeable or changeable data related to the same PRIMARY IDENTIFIER was successful. Note: The client does not need to stop an ongoing update process (e.g. if multiple functions are written), when updateRequired is set back to “false”. The server MAY choose to withdraw the update request at any time by setting updateRequired back to “false”.	Request a write to timeSeriesData from the bound client.
measurementId	MAY be set as FOREIGN IDENTIFIER to link data of other Features to the timeSeries.	Allows to link data of other Features to a timeSeries, e.g. Measurement or ElectricalConnection data
currency	If set, the currency SHALL be applied to the value in timeSeriesData	Allows to describe the currency of the value, if the value is a price.

unit	SHALL be set if the corresponding value has a unit. If set, the unit SHALL be applied to the value in timeSeriesData	Allows to describe the unit of the value, if the value has a unit.
label	Default rules of section 3.10.1.2.1 SHALL be applied.	Allows to define user friendly name for the timeSeries.
description	Default rules of section 3.10.1.3.1 SHALL be applied.	Allows to define a user friendly description for the timeSeries.
scopeType	None.	Specifies a more detailed meaning of the time series.

Table 142: timeSeriesDescriptionListData Elements

Element "**timeSeriesType**" value rules:

Value	Rule	Description
plan	"plan" SHALL be used to model how a certain value will change over time. value as well as minValue and maxValue MAY be used.	"plan" is used to model how a certain value changes over time.
singleDemand	"singleDemand" SHALL be used to model a single demand with a simple timeSeries that only SHALL use one timeSeriesSlot. value as well as minValue and maxValue MAY be used.	"singleDemand" is used to model a single demand with a simple timeSeries (e.g. this can be used to model a simple energy demand).
constraints	"constaints" SHALL be used to model certain value constraints or limits over time. For this only min- and/or maxValue SHALL be used.	"constaints" is used to model certain value constraints over time with a min- and/or maxValue curve.

Table 143: Element "timeSeriesType" value rules

4.3.30.6 timeSeriesConstraintsListData

4.3.30.6.1 Description

Allows to define constraints for the timeSeries, e.g. what is the maximum amount of slots allowed or similar. This function can be especially valuable if timeSeriesData is writeable but the server has certain constraints. The client can than match its own constraints with the server constraints to form a valid write.

4.3.30.6.2 Rules

None.

4.3.30.6.3 Element rules

Element	Rule	Description
timeSeriesId	SHALL be set as PRIMARY IDENTIFIER.	Allows the identification of a timeSeries. Allows

		linking of the different functions to the same timeSeries. Allows linking of the timeSeries within other Features.
slotCountMin	If set and a corresponding timeSeriesData exists, it SHALL NOT have less identifieable slots.	The minimum amount of slots for a specific time series is stated in this element.
slotCountMax	If set, the corresponding timeSeriesData SHALL NOT have more slots.	The maximum amount of slots for a specific time series is stated in this element. Recurring slots only count as one.
slotDurationMin	If set, the corresponding timeSeriesData SHALL NOT have slots with shorter duration.	If the slots of this time series have a minimum duration, it is denoted here.
slotDurationMax	If set, the corresponding timeSeriesData SHALL NOT have slots with longer duration.	If the slots of this time series have a maximum duration, it is denoted here.
slotDurationStepSize	If set, the duration of a slot SHALL be a multiple of slotDurationStepSize and also slotDurationMin and slotDurationMax SHALL be multiples of slotDurationStepSize.	Slots can have a specific duration step size. This element holds the corresponding value.
earliestTimeSeriesStartTime	If set the start time of a timeSeries SHALL be later than or equal to earliestTimeSeriesStartTime.	This is the earliest start time the corresponding time series is allowed to start.
latestTimeSeriesEndTime	If set, the end time of a timeSeries SHALL be earlier than or equal to latest TimeSeriesEndTime.	This is the latest end time the corresponding times series is allowed to end.
slotValueMin	The value, maxValue or minValue of a slot SHALL be equal to or higher than slotValueMin.	The values within the slots of the corresponding time series are not allowed to be lower than this value.
slotValueMax	The value, maxValue or minValue of a slot SHALL be equal to or lower than slotValueMax.	The values within the slots of the corresponding time series are not allowed to be higher than this value.
slotValueStepSize	If slotValueStepSize is set and slotValueMin as well as slotValueMax is not set, the elements value, maxValue and minValue of a slot SHALL be a multiple of slotValueStepSize. If slotValueStepSize is set and slotValueMin is also set, the elements value, maxValue and minValue of a slot SHALL be	The minimum step size between two different values within the slots of the corresponding time series.

	slotValueMin plus a multiple of slotValueStepSize. If slotValueStepSize is set and slotValueMax is also set, the elements value, maxValue and minValue of a slot SHALL be slotValueMax minus a multiple of slotValueStepSize.	

Table 144: timeSeriesConstraintsListData Element rules

4.3.31 TimeTable

4.3.31.1 Dependencies to other Feature Types

None.

4.3.31.2 Introduction

All time dependent functionality like setpoint-configuration, etc. makes use of this elementary class. It provides a list of time tables, each containing some number of time slots. Each slot has a start and end time. The times can either be absolute (*dateTime*) or recurring (*daysOfWeek* plus *time* and, if required, the recurring interval step elements). The count of slots and the start and end times can be configurable or static (device specific). Other classes that make use of time tables only reference to the *timeTableId*, not to specific time slots.

4.3.31.3 Rules

None.

4.3.31.4 timeTableListData

4.3.31.4.1 Description

The list of time tables and time slots together with the start and end times are modelled in this function.

4.3.31.4.2 Rules

None.

4.3.31.4.3 Element rules

Element	Rule	Description
timeTableId	SHALL be set as PRIMARY IDENTIFIER.	The *timeTableId* is used for indentifying specific table information in the different functions and for relations in other classes like *Setpoint*.

timeSlotId	SHALL be set as SUB IDENTIFIER.	Each slot of a time table has its own slot ID.
recurrenceInformation	None.	Information about the recurrence of a slot.
startTime	None.	Start time of one slot.
endTime	None.	End time of one slot.

Table 145: timeTableListData Element rules

4.3.31.5 timeTableConstraintsListData

4.3.31.5.1 Description

If a time table has some constraints, they are denoted with the help of this function (e.g. the minimum and maximum count of slots (may differ between the time tables)).

4.3.31.5.2 Rules

None.

4.3.31.5.3 Element rules

Element	Rule	Description
timeTableId	SHALL be set as PRIMARY IDENTIFIER.	The *timeTableId* is used for indentifying specific table information in the different functions and for relations in other classes like *Setpoint*.
slotCountMin	If set, the time table SHALL NOT have less slots.	The minimum amount of slots for a specific time table is stated in this element. Recurring slots only count as one.
slotCountMax	If set, the time table SHALL NOT have more slots.	The maximum amount of slots for a specific time table is stated in this element. Recurring slots only count as one.
slotDurationMin	If set, the corresponding time table SHALL NOT have slots with lower duration.	If the slots of this time table have a minimum duration, it is denoted here.
slotDurationMax	If set, the corresponding time table SHALL NOT have slots with higher duration.	If the slots of this time table have a maximum duration, it is denoted here.
slotDurationStepSize	If set, the duration of a slot SHALL be a multiple of slotDurationStepSize.	Slots can have a specific duration step size. This element holds the corresponding value.
slotShiftStepSize	If set, the start time of a slot SHALL be firstSlotBeginsAt + a multiple of slotShiftStepSize.	Slots can have a specific shift step size. This element holds the corresponding value.
firstSlotBeginsAt	If set, the start time of the first slot SHALL be firstSlotBeginsAt.	Especially time tables with fixed slot durations probably need this information to specify, where in time the slots are located (i.e. fixed 15 minute slots can start either at

		00:00:00 (default) or e.g. at 00:10:00). This value only defines, when the time pattern (defined by the element *slotDurationStepSize*) begins on a day.

Table 146: timeTableConstraintsListData Element rules

4.3.31.6 timeTableDescriptionListData

4.3.31.6.1 Description

The more static information about the time tables are described in the *timeTableDescriptionData* function.

4.3.31.6.2 Rules

None.

4.3.31.6.3 Element rules

Element	Rule	Description
timeTableId	SHALL be set as PRIMARY IDENTIFIER.	The *timeTableId* is used for indentifying specific table information in the different functions and for relations in other classes like *Setpoint*.
timeSlotCountChangeable	If set to "true", a client SHALL be allowed to add new slots to or remove slots from the timeTable. Otherwise the element SHALL be omitted or set to "false". The timings of added slots SHALL NOT overlap with existing slots.	States whether the amount of slots is changeable by a client for a specific time table or not.
timeSlotTimesChangeable	If set to "true", a client SHALL be able to change timings of existing slots. Otherwise it SHALL be omitted or set to "false".	States whether the times of the slots of a specific time table are changeable by a client or not.
timeSlotTimeMode	Value rules of Table 148 SHALL be applied.	If either *absolute* or *recurring* time slots or *both* modes are supported is stated in this element.
label	Default rules of section 3.10.1.2.1 SHALL be applied.	A short label of the time table.
description	Default rules of section 3.10.1.3.1 SHALL be applied.	Descriptive information on the time table.

Table 147: timeTableDescriptionListData Element rules

Element "**timeSlotTimeMode**" value rules:

Value	Rule	Description
absolute	None.	Only absolute times are allowed for the time table.
recurring	None.	Only recurring times are allowed for the time table.
both	None.	Absolute and recurring times are allowed for the time table.

Table 148: Element "timeSlotTimeMode" value rules

4.3.31.7 Descriptive information and further definitions

None.

5 Class descriptions

5.1 Introduction

In this chapter, all Classes defined by the EEBUS Initiative e.V. are described. It should be seen as equivalent to the XSD definitions. Everything that can be found there is described within the sections of this chapter. Descriptions and rules for the Functions and Elements are only described within the Feature Types as they define a more specific meaning for a Class (see section 2.1.2.3).

Please refer to section 2.2 for details on the concepts of Standard Classes and Complex Classes.

5.2 Complex Classes

5.2.1 IncentiveTable

5.2.1.1 Introduction

The IncentiveTable data model allows to define tiers with different incentives and boundaries. In the case the incentive is price based, it also could be called a price table. However, there can also be other incentives, CO_2 Emission for example. Within the boundaries of a tier the incentive of the tier apply. This allows to distribute e.g. different types of power to different power consumers. E.g. a photovoltaic system produces more power than can be consumed and feed into the electricity grid (surplus power), in this case the power is for free and can be distributed from the CEM to consumers that have the flexibility to increase their consumption. When all surplus power is consumed the next tier would be the electricity grid feed-in power, which in this example would be cheaper than the electricity grid power. In most cases the electricity grid power would be more expensive and therefore would be the last tier. In the given example we would have three tiers, 1st surplus power on top of that the 2nd electricity grid feed in power and beyond that the 3rd electricity grid power. However, the electricity grid power can also get cheaper, e.g. in case the electricity grid has an excess of power available an incentive to flexible consumers could be given.

Even though power was used in this example, the boundaries can also be defined by other limiters.

The incentiveTable has three different functions:

1. incentiveTable: holds the actual values of the boundaries and incentives
2. incentiveTableDescription: holds data that describes the values of incentiveTable
3. incentiveTableConstraint: allows a server to define certain constraints regarding the incentiveTable, this is especially important if the IncentiveTable is writeable

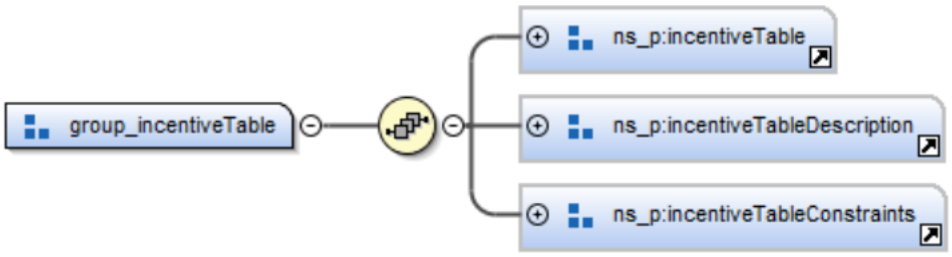


Figure 40: IncentiveTable function-group overview

The tariffId is used to identify the data of different IncentiveTables within each function.

5.2.1.2 *incentiveTableData*

5.2.1.2.1 General

The incentiveTable within the incentiveTableData function holds the values of the IncentiveTable. This includes the boundary values as well as the incentive values for each tier.

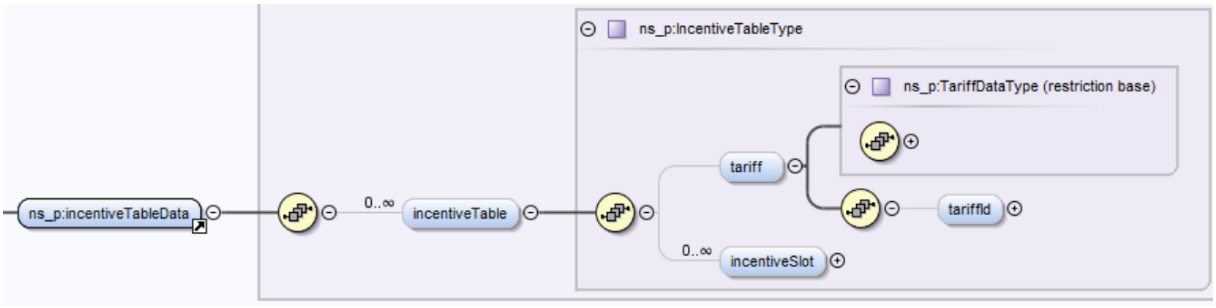


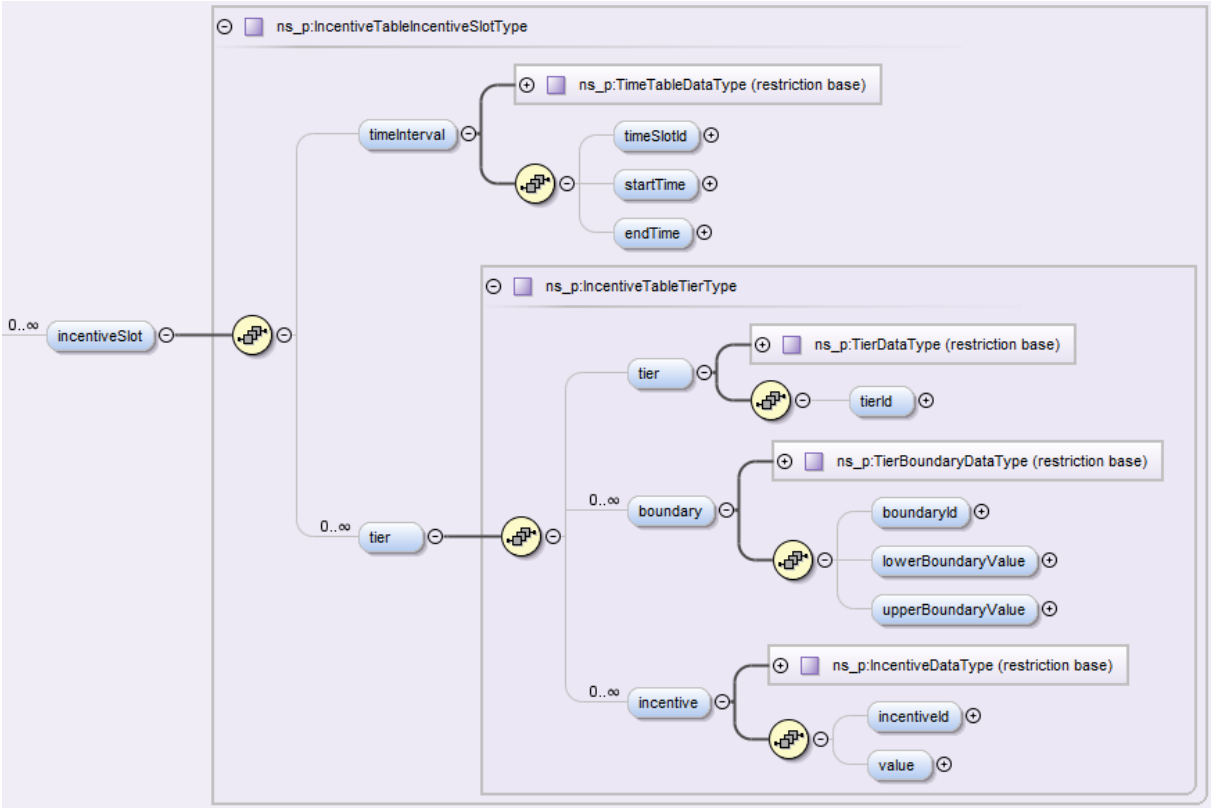
Figure 41: incentiveTableData function overview

5.2.1.2.2 Detailed description of elements

Element	Type	Description
incentiveTable (list)		
incentiveTable. tariff		
incentiveTable. tariff. tariffId	*Identifier "TariffIdType" (see* Table 339*).*	Allows the identification of a incentiveTable. Allows linking of the data in the different functions of IncentiveTable. Allows linking of the incentiveTables within other features that are placed in the same entity.
incentiveTable. incentiveSlot (list)	*Complex type "IncentiveTableIncentiveSlotType". See* Table 150*.*	An incentiveTable can include different incentiveSlots to describe changes of boundaries or incentives over time. Please refer to the detailed description of *IncentiveTableIncentiveSlotType elements* in Table 150.

Table 149: incentiveTable detailed description of elements

IncentiveTableIncentiveSlotType:



5426

5427 *Figure 42: IncentiveTableIncentiveSlotType overview*

Element	Type	Description
timeInterval		Time period of the incentiveSlot.
timeInterval. timeSlotId	*Identifier "TimeSlotIdType" (see* Table 339*).*	Allows identification of an incentiveSlot within an incentiveTable.
timeInterval. startTime		
timeInterval. startTime. dateTime	xs:dateTime (W3C standard type)	Absolute start time.
timeInterval. startTime. relative	xs:duration (W3C standard type)	Relative start time containing a value relative to now. Positive values are in the future, negative ones are in the past and zero seconds represents "now".
timeInterval. endTime		
timeInterval. endTime. dateTime	xs:dateTime (W3C standard type)	Absolute end time.
timeInterval. endTime. relative	xs:duration (W3C standard type)	Relative end time containing a value relative to now. Positive values are in the future, negative ones are in the past and zero seconds represents "now".
tier (list)		
tier. tier		

tier. tier. tierId	*Identifier "TierIdType" (see* Table 339*).*	Allows identification of a tier within a incentiveTable.
tier. boundary (list)		
tier. boundary. boundaryId	*Identifier "TierBoundaryIdType" (see* Table 339*).*	Allows identification of a boundary within a tier.
tier. boundary. lowerBoundaryValue	*Common data type "ScaledNumberType". See section 3.10.1.8.*	This represents the lower end value range boundary.
tier. boundary. upperBoundaryValue	*Common data type "ScaledNumberType". See section 3.10.1.8.*	This represents the upper end value range boundary.
tier. incentive (list)		
tier. incentive. incentiveId	*Identifier "IncentiveIdType" (see* Table 339*).*	Allows identification of an incentive within a tier.
tier. incentive. value	*Common data type "ScaledNumberType". See section 3.10.1.8.*	The value of the incentive.

Table 150: IncentiveTableIncentiveSlotType detailed description of elements

5.2.1.2.3 Available selectors

- tariff. tariffId

5.2.1.3 incentiveTableDescriptionData

5.2.1.3.1 General

The incentiveTableDescription within the incentiveTableDescriptionData function holds the description of the IncentiveTable. This includes the tier, boundary and incentive type as well as further elements.

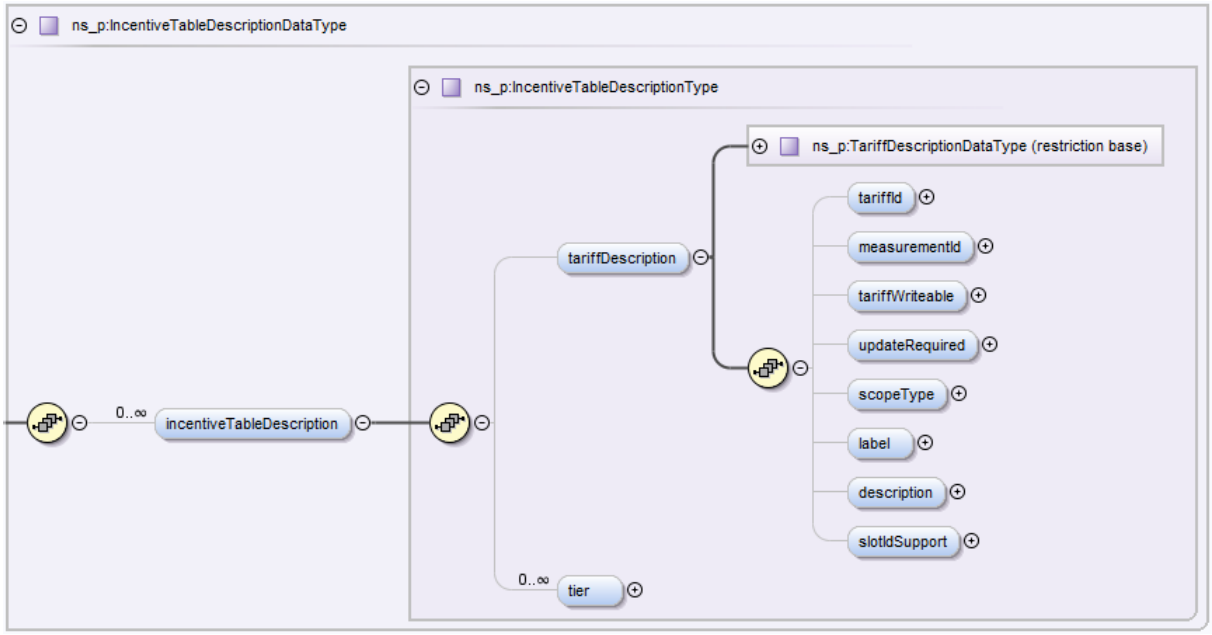


Figure 43: incentiveTableDescriptionData function overview

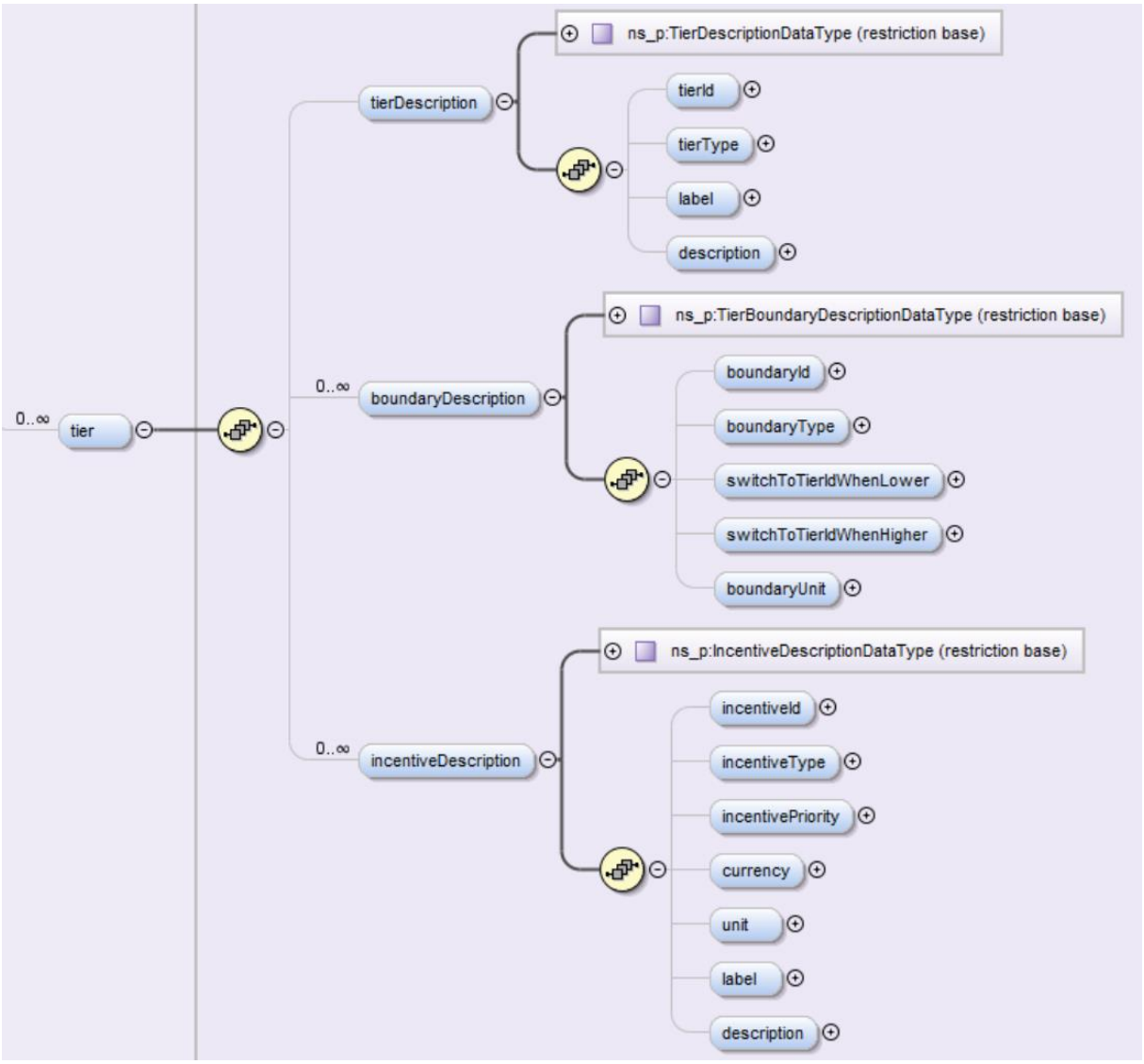
5.2.1.3.2 Detailed description of elements

Element	Type	Description
incentiveTableDescription (list)		
incentiveTableDescription. tariffDescription		
incentiveTableDescription. tariffDescription. tariffId	*Identifier "TariffIdType" (see* Table 339*).*	Allows the identification of a incentiveTable. Allows linking of the data in the different functions of IncentiveTable. Allows linking of the incentiveTables within other features that are placed in the same entity.
incentiveTableDescription. tariffDescription. measurementId	*Identifier "MeasurementIdType" (see* Table 339*).*	If a measurand is linked to the incentive table, its identifier is denoted here.
incentiveTableDescription. tariffDescription. tariffWriteable	xs:boolean (W3C standard type)	Whether the incentive table is writeable by a client or not can be denoted in this element.
incentiveTableDescription. tariffDescription. updateRequired	xs:boolean (W3C standard type)	Allows to request an update from the bound client.
incentiveTableDescription. tariffDescription. scopeType	*Common data type "ScopeTypeType". See section 3.10.1.21.*	A scopeType for the incentive table can be stated here.
incentiveTableDescription. tariffDescription. label	*Common data type "LabelType". See section 3.10.1.2.*	A short label on the incentive table.

incentiveTableDescription. tariffDescription. description	*Common data type "DescriptionType". See section 3.10.1.3.*	A description for the incentive table.
incentiveTableDescription. tariffDescription. slotIdSupport	xs:boolean (W3C standard type)	Indicates support of timeSlotId within the corresponding tariff in the Function incentiveTableData.
tier (list)	*Complex type "IncentiveTableDescriptionTierType". See* Table 152.	Allows to describe tiers and the incentives and boundaries that are used by the tier. Please refer to the detailed description of *IncentiveTableDescriptionTierType elements.*

5442 *Table 151: incentiveTableDescriptionData function detailed description of elements*

5443 IncentiveTableDescriptionTierType:



5444

5445 *Figure 44: IncentiveTableDescriptionTierType overview*

Element	Type	Description
tierDescription		

tierDescription. tierId	*Identifier "TierIdType" (see* Table 339*).*	Allows identification of a tier within a incentiveTable.
tierDescription. tierType	*Union " TierTypeType":* *- Enum (see* Table 153*):* TierTypeEnumType - EnumExtendType (see section 3.10.1.5)	Allows to define the type of the tier
tierDescription. label	*Common data type "LabelType". See section 3.10.1.2.*	Allows to define user friendly name for the tier.
tierDescription. description	*Common data type "DescriptionType". See section 3.10.1.3.*	Allows to define a user friendly description for the tier.
boundaryDescription (list)		
boundaryDescription. boundaryId	*Identifier "TierBoundaryIdType" (see* Table 339*).*	Allows identification of a boundary within a tier.
boundaryDescription. boundaryType	*Union "TierBoundaryTypeType":* *- Enum (see* Table 154*):* TierBoundaryTypeEnumType - EnumExtendType (see section 3.10.1.5)	Allows to define the type of the boundary.
boundaryDescription. switchToTierIdWhenLower	*Identifier "TierIdType" (see* Table 339*).*	Link to the tier that will become active if the lowerBoundaryValue is underrun.
boundaryDescription. switchToTierIdWhenHigher	*Identifier "TierIdType" (see* Table 339*).*	Link to the tier that will become active if the upperBoundaryValue is exceeded.
boundaryDescription. boundaryUnit	*Common data type "UnitOfMeasurementType". See section 3.10.1.17.*	Allows to define the unit of the boundary values.
incentiveDescription (list)		
incentiveDescription. incentiveId	*Identifier "IncentiveIdType" (see* Table 339*).*	Allows identification of an incentive within a tier.
incentiveDescription. incentiveType	*Union "IncentiveTypeType":* *- Enum (see* Table 155*):* IncentiveTypeEnumType - EnumExtendType (see section 3.10.1.5)	Allows to define the type of the incentive.
incentiveDescription. incentivePriority	*Simple type "IncentivePriorityType" (restriction of xs:unsignedInt)*	Allows to define the priority of the incentive. E.g. the priority that the customer has

		assigned for this incentive.
incentiveDescription. currency	*Common data type "CurrencyType". See section 3.10.1.19.*	Monetary incentives may need to describe a currency.
incentiveDescription. unit	*Common data type "UnitOfMeasurementType". See section 3.10.1.17.*	Allows definition of a unit, if the incentive has a unit.
incentiveDescription. label	*Common data type "LabelType". See section 3.10.1.2.*	Allows to express a user friendly label for the incentive.
incentiveDescription. description	*Common data type "DescriptionType". See section 3.10.1.3.*	Allows to express a user friendly description for the incentive.

5446 *Table 152: IncentiveTableDescriptionTierType detailed description of elements*

5447 Enumeration **TierTypeEnumType**:

Value	Rule	Description
fixedCost	The tier has a cost incentive that SHALL NOT vary over time	The fixed cost have to be payed, independent from the consumed energy.
dynamicCost	The tier has a cost incentive that MAY vary over time	The dynamic cost has a (linear) dependency to the consumed energy.

5448 *Table 153: Enumeration TierTypeEnumType*

5449 Enumeration **TierBoundaryTypeEnumType**:

Value	Rule	Description
powerBoundary	The boundary SHALL be an electrical power. E.g. if the surplus power is consumed, more expensive power has to be used.	Provides boundaries for power.
energyBoundary	The boundary SHALL be an electrical energy. E.g. if the stored self produced energy in a battery is consumed, more expensive energy has to be used.	Provides boundaries for energy.
countBoundary	The boundary SHALL be a count. E.g. after a certain count the price of a good may be reduced.	Provides boundaries for counts.

5450 *Table 154: Enumeration TierBoundaryTypeEnumType*

5451 Enumeration **IncentiveTypeEnumType**:

Value	Rule	Description
absoluteCost	None.	Absolute costs with a currency and a unit. The total costs can be calculated.
relativeCost	None.	Relative costs that cannot be interpreted as some monetary costs.

renewableEnergyPercentage	None.	Percentage of renewable energy of the total obtained energy. This is an ecological incentive that allows optimizing consumption of renewable energy instead of costs.
co2Emission	None.	CO2 emission related to the obtained energy. This is an ecological incentive that allows optimizing the CO2 emission instead of costs.

Table 155: Enumeration IncentiveTypeEnumType

5.2.1.3.3 Available selectors

- tariffDescription. tariffId
- tariffDescription. scopeType

5.2.1.4 incentiveTableConstraintsData

5.2.1.4.1 General

The incentiveTableConstraints within the incentiveTableConstraintsData function holds additional constraints of the IncentiveTable, regarding how much tiers, boundaries, incetives and slots are allowed.

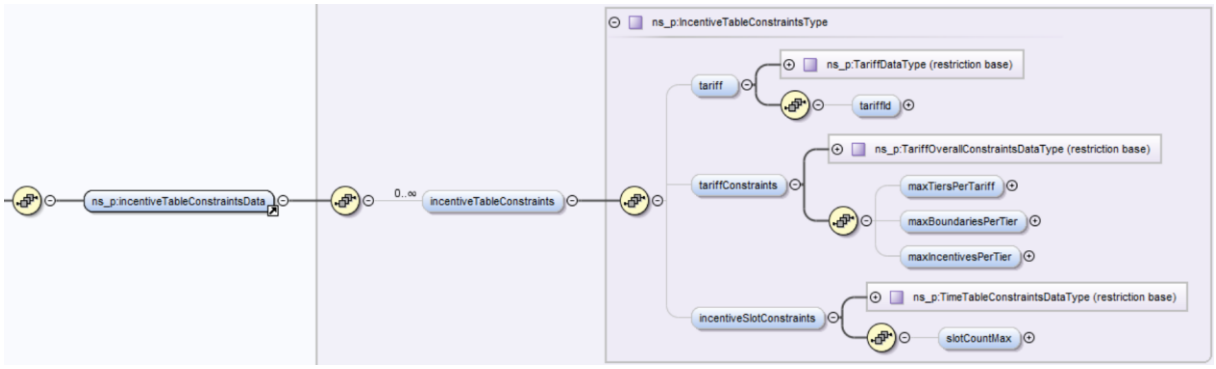


Figure 45: incentiveTableConstraintsData overview

5.2.1.4.2 Detailed description of elements

Element	Type	Description
incentiveTableConstraints (list)		
incentiveTableConstraints. tariff		
incentiveTableConstraints. tariff. tariffId	*Identifier "TariffIdType" (see* Table 339*).*	Allows the identification of a incentiveTable. Allows linking of the different functions within the IncentiveTable Feature. Allows linking of the

		incentiveTable within other Features that are placed in the same entity.
incentiveTableConstraints. tariffConstraints		
incentiveTableConstraints. tariffConstraints. maxTiersPerTariff	*Simple type "TierCountType" (restriction of "TierIdType" (see* Table 339*)).*	Allows the server to define a maximum tier count for this IncentiveTable. This is especially important if a client wants to write an IncentiveTable.
incentiveTableConstraints. tariffConstraints. maxBoundariesPerTier	*Simple type "TierBoundaryCountType" (restriction of "TierBoundaryIdType" (see* Table 339*)).*	Allows the server to define a maximum boundaries count for tiers. This is especially important if a client wants to write an IncentiveTable.
incentiveTableConstraints. tariffConstraints. maxIncentivesPerTier	*Simple type "IncentiveCountType" (restriction of "IncentiveIdType" (see* Table 339*)).*	Allows the server to define a maximum incentive count for tiers. This is especially important if a client wants to write an IncentiveTable.
incentiveTableConstraints. incentiveSlotConstraints		
incentiveTableConstraints. incentiveSlotConstraints. slotCountMax	*Simple type "TimeSlotCountType" (restriction of "TimeSlotIdType", see* Table 328*).*	Allows the server to define a maximum slot count for this IncentiveTable. This is especially important if a client wants to write an IncentiveTable.

Table 156: incentiveTableConstraints detailed description of elements

5.2.1.4.3 Available selectors

- tariff. tariffId

5.2.2 NodeManagement

See document [ProtocolSpecification], chapter "Functional commissioning" for details.

5.2.3 SmartEnergyManagementPs

5.2.3.1 Introduction

5.2.3.1.1 Scope of this complex class

This complex class provides solutions for “smart energy management” use cases that are typical for domestic homes: On the one hand, energy providers often offer their energy with a price that varies over time (e.g. consumption at night is cheaper than at noon). The availability of renewable energies from an own photovoltaics also varies over time. On the other hand, the residents certainly need to run their white goods (do one’s laundry etc.) and other machines, but may have the opportunity to

shift these tasks in order to reduce the overall cost. Such kind of optimisation is hard to achieve manually in order to gain a real benefit. As a consequence, some kind of automation is required.

This complex class provides a solution that can best be expressed with the definition of two roles (i.e. device types) for these roles, and the information exchanged between them. The following roles are assumed:

1. A “smart appliance”:
   This is kind of device is usually an energy consuming or producing device, i.e. a device whose activity can be worth to be scheduled for a specific time in order to reduce the overall cost. Typical devices are whitegoods, photovoltaics, or battery storages. Such devices need to be “smart” as they have to be able to provide some information on their activity (esp. ahead of time) and (if possible) the capability to shift their activity upon appropriate requests from a role that is described next.
2. An energy management system:
   A smart appliance usually cannot have the information required to perform an overall optimization of the cost and possibility for energy consumption and production. Instead, this is the responsibility of an energy management system. Such a system can gather activity information from several appliances of the domestic home. It may even be able to fetch information from an energy provider to get details of the tariff in order to execute up-to-date price calculations. This collection of information can then be used to adjust the activities of the connected appliances.

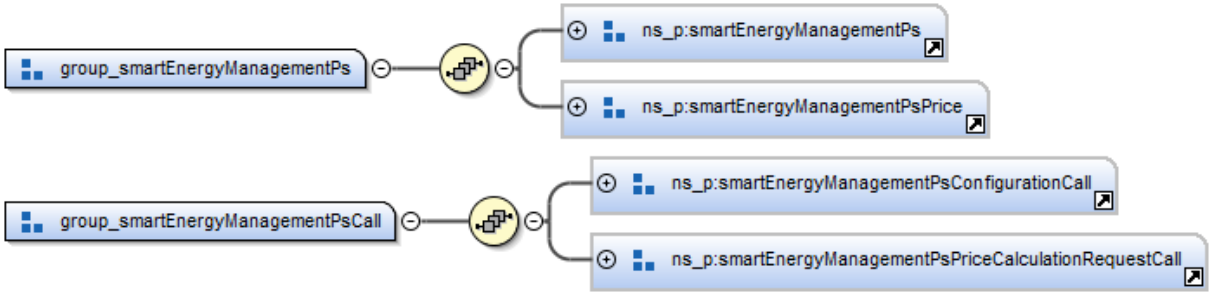


Figure 46: SmartEnergyManagementPs function-group overview

5.2.3.1.2 Re-use of standard classes: PowerSequences etc.

This complex class is mainly based upon definitions and concepts of the class PowerSequences (see section 5.3.19) which are used to express power or energy over time. In fact, this complex class primarily groups definitions from this and some other classes together into “useful collections”. Among others, this helps to avoid many single messages and gives the corresponding data parts a more “direct” relation to each other. This way, the overall structure of the complex class basically still represents power sequences. Therefore, throughout this complex class the term “power sequences” is still used. Section 5.2.3.1.3 briefly introduces the power sequences concept of this complex class.

5.2.3.1.3 Theory of operation (informative)

The power sequences structures of this complex class model the energy consumption / generation / storage as a mathematical graph and additional information, such as constraints for time based shifts.

Throughout this complex class, two terms - from the inherited PowerSequences class - are of major interest: “sequence” and “slot”. A sequence can be used to describe a “task”. It is the most “coarse” view in this class, i.e. a sequence represents all single steps of a whole task. The single steps are represented by slots. Slots are the most “fine” view in this class with regards to the time resolution of a task.

Each sequence and slot can be associated with an identifier or number, respectively. The slot numbers of two sequences should be considered independent from each other. I.e. slot number 7 of sequence 1 describes a different slot than slot number 7 of sequence 2. This basically means a slot is only “uniquely” identified in combination with a sequence ID. As formal notation this leads to the following relation:

- 1 sequence has 1..m slots
- 1 slot belongs to exactly 1 sequence
- 1 pair of a sequenceId and slotNumber uniquely identifies a slot

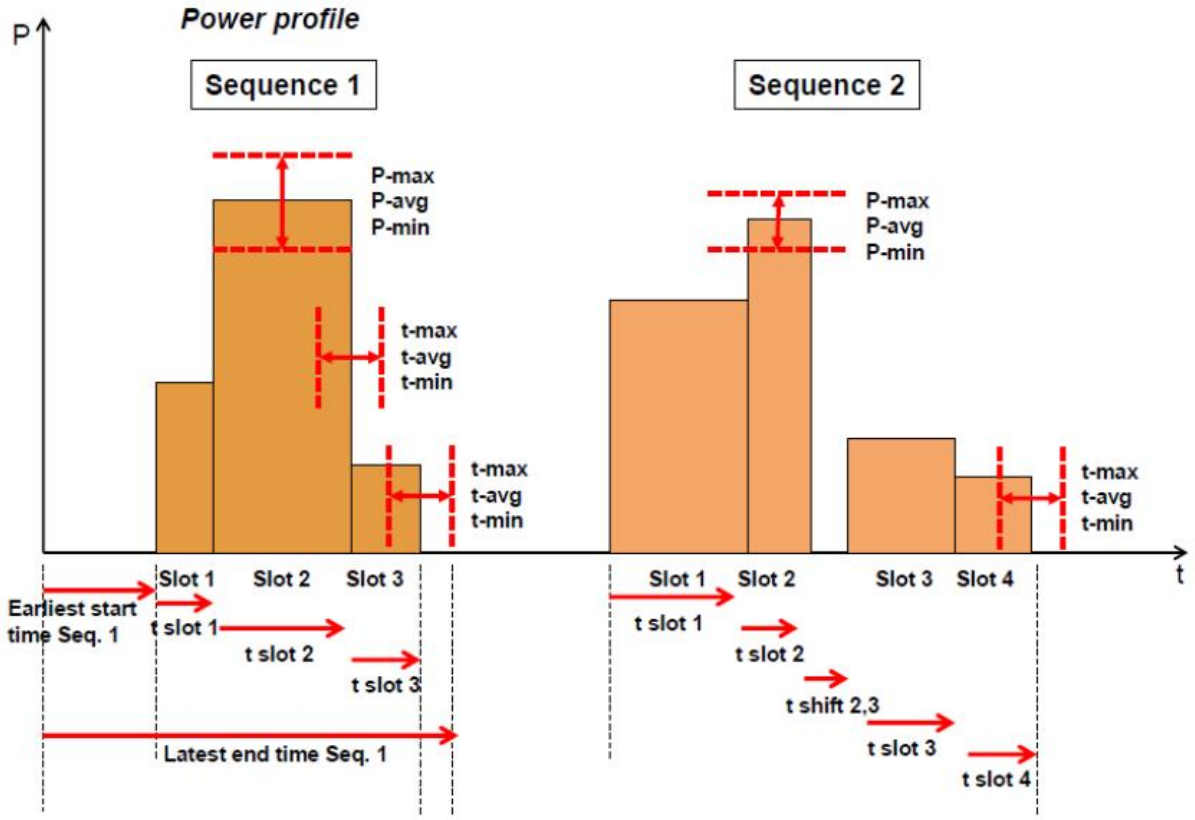


Figure 47: Concept of a power sequence

Each slot is, among other things, characterized by a time it applies for and a list of valid values, which apply for this time, such as average energy consumption, maximum energy consumption and others.

In addition to value properties, a slot and a sequence can be described by their constraints. Constraints state whether a sequence and/or its single slots can be shifted in time. Additionally, if this is the case, the constraints state the boundaries for time shifts or duration configurations. This contributes to the definition of a power sequence's flexibility.

Some time-related information on a slot shall be considered:

- startTime: The currently scheduled start time of the slot.
- endTime: The currently scheduled end time of the slot.
- defaultDuration: The currently scheduled duration of this slot.
- durationUncertainty: The duration may have some uncertainty. The actual duration then may be in the range from defaultDuration - durationUncertainty to defaultDuration + durationUncertainty, e.g.
- earliestStartTime: The earliest time a slot could start. This determines the "lowest" boundary to shift a slot.
- latestEndTime: The latest time a slot has to be finished. This determines the "highest" boundary to shift a slot.

Within a power sequence, there may be mandatory and optional slots. An example for an optional slot is a pause/intentional gap between 2 process steps of a device, where the pause might either take place or can be skipped.

Within a sequence, the model permits in theory to have a "time gap" between two consecutive slots. However, for some reason it is usually better to express such a gap (or "pause") as regular slot with an energy value of "0".

Each sequence and slot can be described using textual information, e.g. for displaying it on a smartphone or an in home display.

A power sequences server can also request a price calculation for a sequence, e.g. to display a price on a device display to the user.

In general, a so-called power sequences server (see section 5.2.3.1.4) can offer one or more power sequences. A power sequence can contain flexibility or can instead be used to forecast fixed consumption or generation behaviour without any flexibility. The granularity, i.e. the number of slots and value types provided for the slots a device models per power sequence, is a choice which is up to the manufacturer.

A power sequences server can offer multiple power sequences for different purposes:

1. The power sequences can model choices a so-called power sequences client (see 5.2.3.1.4) can take.
   As an example, consider a freezer which can run in normal mode or in deep freezing mode (which means increased power consumption, but the time frame for the power consumption is shorter). This choice can be offered to the power sequences client. In this complex class the power sequences of a choice are combined in a group called "alternatives" (see below).
2. The power sequences can model behaviour over time.

As an example, consider an HVAC system which knows that it has to run in the morning and in the evening. A power sequences server can publish 2 different power sequences, one for each time frame.

3. The power sequences can model mixtures of choices and behaviour over time (1 and 2).

The general concept of power sequences is applicable to many kinds of devices. The above mentioned “alternatives” groups are based on the same concept as the “alternatives” concept of the PowerSequences class (see section 5.3.19.1.3). Within this complex class an “alternatives” group is modelled in the root element “alternatives” of function smartEnergyManagementPsData.

Summarizing, there are several kinds of data contributing to the extent of a power sequences server's flexibility:

1. Several flags denoting whether the power sequences server permits configuration from a bound power sequences client.
2. The number of power sequences of an "alternatives" group (i.e. whether a choice from multiple concurrent power sequences can be taken).
3. The possibility for shifting a sequence or its slots (see above for explanation on constraints).

5.2.3.1.3.1 Power sequences application behaviour guidance (informative)

In general, the SmartEnergyManagementPs complex class permits a lot of different use cases or applications to be realised. Thus, it is rather difficult to give a complete overview for all applications. Instead, this section gives a guideline, which covers the main aspects that are typical for most applications.

A power sequence in general is one of the following:

- An energy- or power-forecast for a process without any flexibilities (fixed forecast)
- An energy- or power-forecast that can be shifted, chosen or configured by a power sequences client

Anyhow, the basic application guideline is common for both types of power sequences.

We assume a power sequences server and a power sequences client established a communication channel. At some point in time, a process is going to be started on the device which contains the power sequences server. The power sequences server thus exposes at least one power sequence for this process. Depending on the kind of action the device wants to start, the power sequences server exposes fixed forecast(s), i.e. power sequence(s) without any flexibilities, power sequences with flexibilities, which might also contain alternative choices, or a mixture of both.

Once a power sequences client receives one or more power sequences from a server, it should start the evaluation of the received data. The first thing to determine is, whether the power sequences server has exposed power sequence(s) which allow for configuration, or whether they are fixed forecasts.

Independent of the type of the power sequence(s), the power sequences server should notify changes of the power sequence(s) (configuration and state changes, e.g.) according to the current situations.

If the power sequences contain flexibilities and/or alternatives, the power sequences server can send a power sequences configuration request call. Once the power sequence client receives this message, it can evaluate whether it will configure the according power sequences server or ignore the call (temporarily or permanently). If the power sequences client does not configure the power sequences server, the power sequences server follows the configuration it has initially exposed, i.e. its own preference. If the power sequences client wants to configure the server it can achieve this by writing a power sequences configuration to the power sequence server. If the power sequences server receives a configuration, it first checks whether the configuration is valid. If the received configuration is valid, i.e. meets the constraints the power sequences server has laid out, it takes over the configuration.

During runtime, the power sequences client and server may re-negotiate the configuration multiple times, even if the according power sequence is already running (if the power sequences server supports it). However, every time the data on either the power sequence information, configuration or state changes, the power sequences server should update the subscribed nodes. Of course, this should also be done at the completion of a power sequence.

If new processes are going to be considered, these mechanisms are then repeated.

5.2.3.1.3.2 Pausing and resuming power sequences (informative)

Some devices may permit to pause and resume specific power sequences. Such sequences can be marked as pausable (via function smartEnergyManagementPsData, “alternatives. powerSequence. operatingConstraintsInterrupt. isPausable” = "true", eg.). The sequence needs to be remotely controllable, too (which may be expressed by “alternatives. powerSequence. state. sequenceRemoteControllable” = "true" in function smartEnergyManagementPsData, e.g.), when a pause command shall be executed. Of course, the device itself needs to permit power sequences configuration in general (which may be expressed by “nodeScheduleInformation. nodeRemoteControllable” = "true" in function smartEnergyManagementPsData, e.g.). The power sequences client then may pause a running sequence and resume it at a later time (provided that the constraints are met).

Please note that a device may resume its actions on its own, to meet the configured constraints. It then SHALL inform all subscribed clients about the changed operation.

5.2.3.1.4 Role concept

The SPINE Protocol Specification defines the terms “server” and “client” which apply here as well. With regards to this Smart Energy Management specification the terms also get a specific semantic which is briefly described in this section.

A device offering time based shifts for its energy consumption/generation/storage or fixed forecasts or mode switches (between consumption/generation/storage) SHALL be called a power sequences server.

A device providing configuration for the flexibilities another device offers and/or understanding fixed forecasts SHALL be called a power sequences client.

Applications using this complex class may well benefit from binding or subscription mechanisms. Therefore, the following terms are defined:

- A power sequences client which has 1 or more bindings to power sequences servers is called a “bound power sequences client”.
- A power sequences client which does not have a binding to a power sequences server but a subscription instead is called a “subscribed power sequences client”.

5.2.3.2 Function specifications

5.2.3.2.1 smartEnergyManagementPsData

5.2.3.2.1.1 General

This function is modelled to keep all “operational” power sequences information.

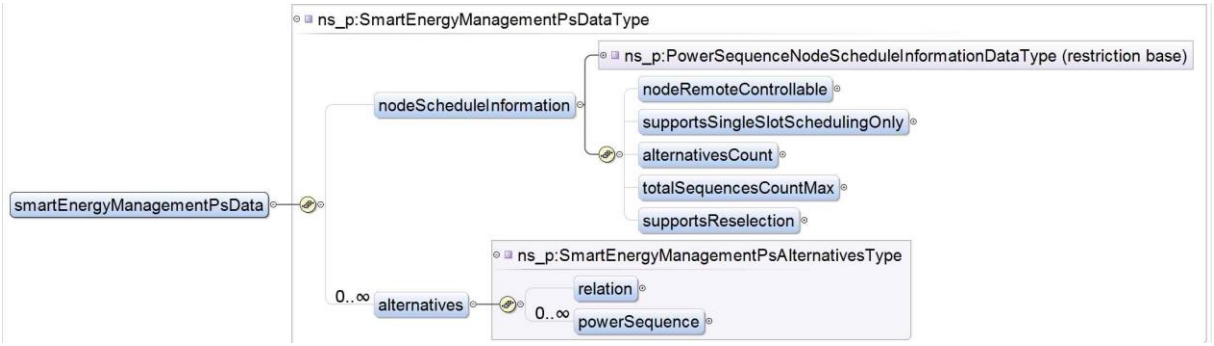


Figure 48: smartEnergyManagementPsData function overview, part 1

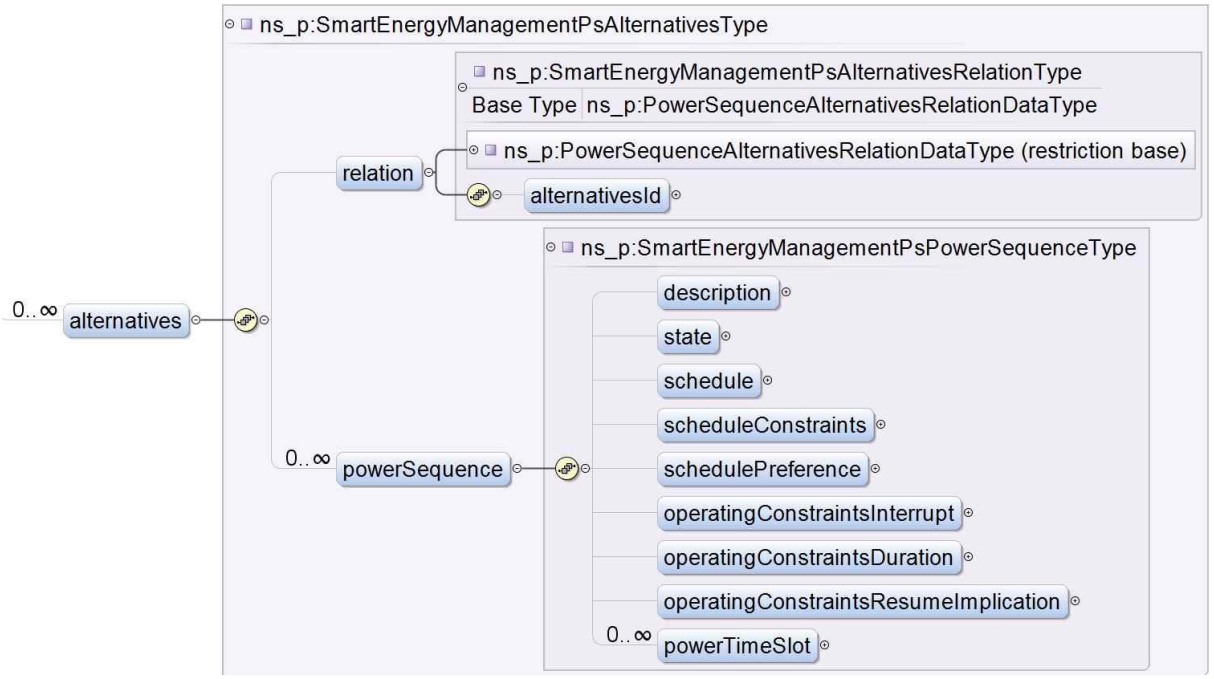


Figure 49: smartEnergyManagementPsData function overview, part 2

Figure 50: smartEnergyManagementPsData function overview, part 3

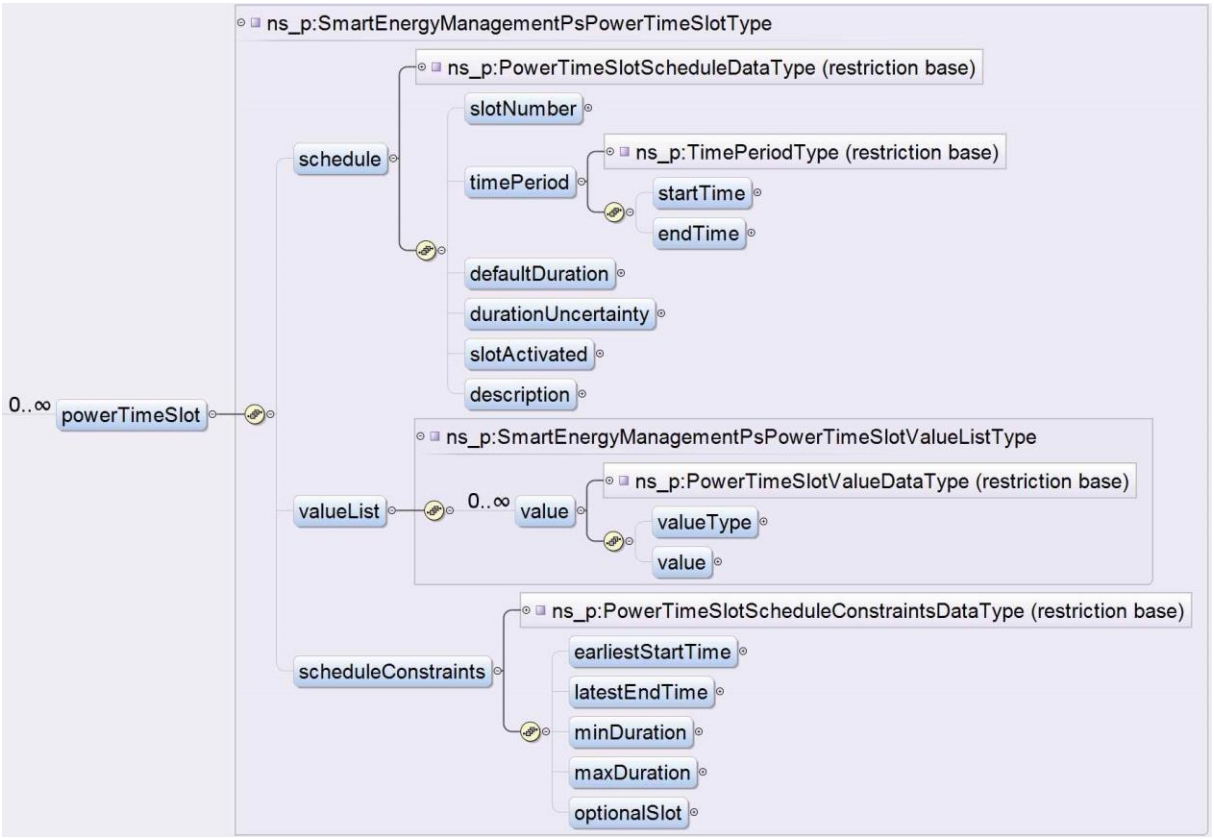


Figure 51: smartEnergyManagementPsData function overview, part 4

5.2.3.2.1.2 Detailed description of elements

The complex function “smartEnergyManagementPsData” of the complex class “SmartEnergyManagementPs” is described in the following table:

Element name	Data type	Explanation
nodeScheduleInformation		Contains “global”/summary information (independent from a specific sequence). Note: This element is derived from the PowerSequences class function “powerSequenceNodeScheduleInformationData”.
nodeScheduleInformation. nodeRemoteControllable	xs:boolean	Indicates whether the power sequences server (e.g., appliance) is configured for remote control (e.g., by an energy management system).This refers to the selection chosen by the user on the remote control feature of the device. If the value is "false", the remote energy management is disabled, otherwise it is enabled.
nodeScheduleInformation. supportsSingleSlotSchedulingOnly	xs:boolean	If set to “true” the power sequences server does NOT permit the modification of more than one slot per configuration command received from a client. I.e. in

		this case a client needs to submit individual configuration commands for each slot.
nodeScheduleInformation. alternativesCount	xs:unsignedInt	The number of “alternatives” groups. I.e. the number of occurrences of the element “alternatives”.
nodeScheduleInformation. totalSequencesCountMax	xs:unsignedInt	The total number of sequences supported by the power sequences server. I.e. the sum of all power sequences across all “alternatives” group in this message (i.e. also regardless of the sequences’ states).
nodeScheduleInformation. supportsReselection	xs:boolean	If set to “true” the power sequences server does not restrict the number of sequence re-selections by a power sequences client. I.e. within a given “alternatives” group the power sequences client may first choose one sequence, alter the selection by configuring another sequence later on, then alter the selection again, etc. (provided the process rules and data still permit configuration). If supportsReselection is set to “false”, on the other hand, the power sequences server permits the power sequences client to select a sequence of an “alternatives” group only one time.
alternatives (list)		Each occurrence of “alternatives” announces a new group of mutually alternative sequences.
alternatives. relation		Note: This element is derived from the PowerSequences class function “powerSequenceAlternativesRelationData”.
alternatives. relation. alternativesId	*Identifier "AlternativesIdType" (see* Table 339*).*	The endpoint-wide unique identifier of the “alternatives” group.
alternatives. powerSequence	List of data group (1..unbounded)	Each occurrence of “powerSequence” announces a new group of power sequences data.
alternatives. powerSequence. description		General descriptions on the power sequence. Note: This element is derived from the PowerSequences class function “powerSequenceDescriptionData”.
alternatives. powerSequence. description. sequenceId	*Identifier "PowerSequenceId*	The endpoint-wide unique sequence identifier.

	Type" (see Table 339*).*	
alternatives. powerSequence. description. description	xs:string (1..60 characters)	If used, should state a textual description for this power sequence.
alternatives. powerSequence. description. powerUnit	*Common data type "UnitOfMeasurementType". See section 3.10.1.17.*	The unit that applies to all power related values of this function.
alternatives. powerSequence. description. energyUnit	*Common data type "UnitOfMeasurementType". See section 3.10.1.17.*	The unit that applies to all energy related values of this function.
alternatives. powerSequence. description. valueSource	*Union "*MeasurementValueSourceType*":* - *Enum (see* Table 232*):* MeasurementValueSourceEnumType - EnumExtendType (see section 3.10.1.5)	The source (origin / foundation) of the forecasted values for this power sequence, such as “calculatedValue”. Remark: This element shall express the reliability of the forecast.
alternatives. powerSequence. description. taskIdentifier	xs:unsignedInt	This element is used if a server wants to uniquely identify reoccurring types of sequences. As an example: Specific types of washing cycles with specific parameters should have the same taskIdentifier every time they are offered using power sequences. Note: This element should not be confused with “taskId” of the class TaskManagement.
alternatives. powerSequence. description. repetitionsTotal	xs:unsignedInt	If a power sequence executes its sequence of slots more than one time, the element contains the total number of executions.
alternatives. powerSequence. state		Information about the state of the power sequence. Note: This element is derived from the PowerSequences class function “powerSequenceStateData”.

alternatives. powerSequence. state. state	*Union "*PowerSequenceStateType*":* - *Enum (see* Table 269*):* PowerSequenceStateEnumType - EnumExtendType (see section 3.10.1.5)	The current state of the sequence is stated in this element.
alternatives. powerSequence. state. activeSlotNumber	*Identifier "PowerTimeSlotNumberType" (see* Table 339*).*	Contains the currently active slot number.
alternatives. powerSequence. state. elapsedSlotTime	xs:duration	This element contains the time the slot has already been in “running” state (this also means the value remains constant during a “paused” state).
alternatives. powerSequence. state. remainingSlotTime	xs:duration	This element contains the time the slot still needs to be in “running” state (this also means the value remains constant during a “paused” state).
alternatives. powerSequence. state. sequenceRemoteControllable	xs:boolean	Denotes whether the sequence can be configured by a client (value “true”) or not (value “false”).
alternatives. powerSequence. state. activeRepetitionNumber	xs:unsignedInt	activeRepetitionNumber denotes the number of the current execution of the sequence of slots.
alternatives. powerSequence. state. remainingPauseTime	xs:duration	Denotes the duration the current slot (element activeSlotNumber) permits being paused.
alternatives. powerSequence. schedule		Information about the schedule of the power sequence. Note: This element is derived from the PowerSequences class function “powerSequenceScheduleData”.
alternatives. powerSequence. schedule. startTime	xs:duration	The startTime of the power sequence.
alternatives. powerSequence. schedule. endTime	xs:duration	The endTime of the power sequence.
alternatives. powerSequence. scheduleConstraints		Information about the schedule constraints of the power sequence.

		Note: This element is derived from the PowerSequences class function “powerSequenceScheduleConstraintsData”.
alternatives. powerSequence. scheduleConstraints. earliestStartTime	xs:duration	The earliest possible start time for this whole power sequence.
alternatives. powerSequence. scheduleConstraints. latestEndTime	xs:duration	The latest possible end time for this whole power sequence.
alternatives. powerSequence. schedulePreference		Contains information on the power sequences server’s preference for its scheduling. Note: This element is derived from the PowerSequences class function “powerSequenceSchedulePreferenceData”.
alternatives. powerSequence. schedulePreference. greenest	xs:boolean	If present and set to “true”: The power sequences client should try to optimise the configuration towards the minimum usage of NON-renewableenergy.
alternatives. powerSequence. schedulePreference. cheapest	xs:boolean	If present and set to “true”: The power sequences client should try to apply a configuration that minimises the user’s energy bill for this power sequence.
alternatives. powerSequence. operatingConstraintsInterrupt		Information about the interruptibility of the power sequence. Note: This element is derived from the OperatingConstraints class function “operatingConstraintsInterruptData”.
alternatives. powerSequence. operatingConstraintsInterrupt. isPausable	xs:boolean	If the sequence is pausable by the bound power sequences client, this element is set to true.
alternatives. powerSequence. operatingConstraintsInterrupt. isStoppable	xs:boolean	If the sequence is stoppable by the bound power sequences client, this element is set to true.
alternatives. powerSequence. operatingConstraintsInterrupt. maxCyclesPerDay	xs:unsignedInt	States the maximum amount of starts the device allows per day.

alternatives. powerSequence. operatingConstraintsDuration		Constraints about minimum or maximum durations of active- or pause-phases. Note: This element is derived from the OperatingConstraints class function “operatingConstraintsDurationData”.
alternatives. powerSequence. operatingConstraintsDuration. activeDurationMin	xs:duration	This is the minimum duration the sequence has to run without interruption.
alternatives. powerSequence. operatingConstraintsDuration. activeDurationMax	xs:duration	This is the maximum duration the sequence can run without interruption.
alternatives. powerSequence. operatingConstraintsDuration. pauseDurationMin	xs:duration	This is the minimum duration the sequence has to pause after the end of an activity.
alternatives. powerSequence. operatingConstraintsDuration. pauseDurationMax	xs:duration	This is the maximum duration the sequence can pause after the end of an activity.
alternatives. powerSequence. operatingConstraintsDuration. activeDurationSumMin	xs:duration	This is the minimum duration the sequence has to run in total (summation of all active times).
alternatives. powerSequence. operatingConstraintsDuration. activeDurationSumMax	xs:duration	This is the maximum duration the sequence can run in total (summation of all active times).
alternatives. powerSequence. operatingConstraintsResumeImplication		If resuming an active phase after a pause has implications on energy or price costs, this element and the required sub-elements contain the proper information. Note: This element is derived from the OperatingConstraints class function “operatingConstraintsResumeImplicationData”.
alternatives. powerSequence. operatingConstraintsResumeImplication. resumeEnergyEstimated	*Common data type "ScaledNumberType". See section 3.10.1.8.*	Additional energy the device will consume before resuming its normal operation (after a pause). This is only an estimated value which will not be added to the value stated in any slot value information.

alternatives. powerSequence. operatingConstraintsResumeImplication. energyUnit	*Common data type "UnitOfMeasurementType". See section 3.10.1.17.*	Unit for the energy stated in the element resumeEnergyEstimated.
alternatives. powerSequence. operatingConstraintsResumeImplication. resumeCostEstimated	*Common data type "ScaledNumberType". See section 3.10.1.8.*	Additional costs for the resumption of a device to its normal operation (after a pause).
alternatives. powerSequence. operatingConstraintsResumeImplication. currency	*Common data type "CurrencyType". See section 3.10.1.19.*	Currency for the price stated in the element resumeCostEstimated.
alternatives. powerSequence. powerTimeSlot	List of data group (1..unbounded)	Each occurrence of “powerTimeSlot” announces a new “slot information group”.
alternatives. powerSequence. powerTimeSlot. schedule		Information about the schedule of a slot. Note: This element is derived from the PowerSequences class function “powerTimeSlotScheduleData”.
alternatives. powerSequence. powerTimeSlot. schedule. slotNumber	*Identifier "PowerTimeSlotNumberType" (see* Table 339*).*	The powerSequenceId-wide unique slot identifier.
alternatives. powerSequence. powerTimeSlot. schedule. timePeriod		
alternatives. powerSequence. powerTimeSlot. schedule. timePeriod. startTime	xs:duration	The start time of the slot.
alternatives. powerSequence. powerTimeSlot. schedule. timePeriod. endTime	xs:duration	The end time of the slot.
alternatives. powerSequence. powerTimeSlot. schedule. defaultDuration	xs:duration	The duration of the slot.
alternatives. powerSequence. powerTimeSlot. schedule. durationUncertainty	xs:duration	The uncertainty of the duration given in the element alternatives. powerSequence. powerTimeSlot. schedule. defaultDuration.

alternatives. powerSequence. powerTimeSlot. schedule. slotActivated	xs:boolean	If the slot is optional, i.e. “alternatives. powerSequence. powerTimeSlot. scheduleConstraints. optionalSlot” is set to “true”, this element reflects the current status of the slot (true = the slot will be executed, false = the slot will not be executed).
alternatives. powerSequence. powerTimeSlot. schedule. description	*Common data type "DescriptionType". See section 3.10.1.3.* (1..60 characters)	A textual description for this slot.
alternatives. powerSequence. powerTimeSlot. valueList		Parent element of subsequent list.
alternatives. powerSequence. powerTimeSlot. valueList. value	List of “alternatives. powerSequence. powerTimeSlot. valueList. value” (0..unbounded)	Note: This element is derived from the PowerSequences class function “powerTimeSlotValueData”.
alternatives. powerSequence. powerTimeSlot. valueList. value. valueType	*Union* "PowerTimeSlotValueTypeType*":* *- Enum (see* Table 263*):* PowerTimeSlotValueTypeEnumType - EnumExtendType (see section 3.10.1.5)	The value type of this value.
alternatives. powerSequence. powerTimeSlot. valueList. value. value	*Common data type "ScaledNumberType". See section 3.10.1.8.*	A value according to the type expressed in alternatives. powerSequence. powerTimeSlot. valueList. value. valueType. The value’s unit is stated in alternatives. powerSequence. description. powerUnit or alternatives. powerSequence. description. energyUnit.
alternatives. powerSequence. powerTimeSlot. scheduleConstraints		Information about constraints for a time slot. Note: This element is derived from the PowerSequences class function “powerTimeSlotScheduleConstraintsData”.

alternatives. powerSequence. powerTimeSlot. scheduleConstraints. earliestStartTime	xs:duration	The earliest start time for this slot.
alternatives. powerSequence. powerTimeSlot. scheduleConstraints. latestEndTime	xs:duration	The latest possible end time for this slot.
alternatives. powerSequence. powerTimeSlot. scheduleConstraints. minDuration	xs:duration	If the slot has a configurable duration, this element denotes the minimum supported configuration.
alternatives. powerSequence. powerTimeSlot. scheduleConstraints. maxDuration	xs:duration	If the slot has a configurable duration, this element denotes the maximum supported configuration.
alternatives. powerSequence. powerTimeSlot. scheduleConstraints. optionalSlot	xs:boolean	SHALL be present and set to a value of “true” denotes the slot can be omitted.

Table 157: Description of complex class function “smartEnergyManagementPsData”

5.2.3.2.1.3 Identifiers

- alternatives. powerSequence. description. sequenceId
- alternatives. powerSequence. powerTimeSlot. schedule. slotNumber

Please note: "alternatives. relation. alternativesId" is NOT an identifier as it does not open a further dimension in the data room. Instead it is kind of a group for one or more sequences.

5.2.3.2.1.4 Available selectors

The selectors definition for function “smartEnergyManagementPsData” is given in the following table:

Element name	**Data range**	**Explanation**
smartEnergyManagementPsDataSelectors		
smartEnergyManagementPsDataSelectors. alternativesRelation		Note: This element is derived from the PowerSequences class selectors definition “powerSequenceAlternativesRelationListDataSelectors”.

smartEnergyManagementPsDataSelectors. alternativesRelation. alternativesId	*Identifier "AlternativesIdType" (see* Table 339*).*	Selector item: The proper alternatives group is selected.
smartEnergyManagementPsDataSelectors. powerSequenceDescription		Note: This element is derived from the PowerSequences class selectors definition “powerSequenceDescriptionListDataSelectors”.
smartEnergyManagementPsDataSelectors. powerSequenceDescription. sequenceId	*Identifier "PowerSequenceIdType" (see* Table 339*).*	Selector item: The proper power sequence is selected.
smartEnergyManagementPsDataSelectors. powerTimeSlotSchedule		Note: This element is derived from the PowerSequences class selectors definition “powerTimeSlotScheduleListDataSelectors”.
smartEnergyManagementPsDataSelectors. powerTimeSlotSchedule. slotNumber	*Identifier "PowerTimeSlotNumberType" (see* Table 339*).*	Selector item: The proper slot is selected.
smartEnergyManagementPsDataSelectors. powerTimeSlotValue		Note: This element is derived from the PowerSequences class selectors definition “powerTimeSlotValueListDataSelectors”.
smartEnergyManagementPsDataSelectors. powerTimeSlotValue. valueType	*Union "*PowerTimeSlotValueTypeType*":* *- Enum (see* Table 263*):* PowerTimeSlotValueTypeEnumType - EnumExtendType (see section 3.10.1.5)	Selector item: The kind of the value ("power", "powerMin", "energy", ...).

Table 158: Dedicated "selectors" of “smartEnergyManagementPsData”.

5.2.3.2.2 smartEnergyManagementPsConfigurationRequestCall

5.2.3.2.2.1 General

A power sequences server may want to explicitly inform (or remind) a power sequences client if it wants one or “all” of its power sequences to be configured. The server can express this by sending a power sequences configuration request call to the client.

Figure 52: smartEnergyManagementPsConfigurationRequestCall function overview

5698 *5.2.3.2.2.2 Detailed description of elements*

5699 The complex function “smartEnergyManagementPsConfigurationRequestCall” of the complex class
5700 “SmartEnergyManagementPs” is described in the following table:

Element name	Data range	Explanation
scheduleConfigurationRequest		Note: This element is derived from the PowerSequences class function “powerSequenceScheduleConfigurationRequestCall”.
scheduleConfigurationRequest.sequenceId	*Identifier "PowerSequenceIdType" (see* Table 339*).*	States the sequence ID of the power sequences server which it prefers for configuration.

5701 *Table 159: Description of complex class function “smartEnergyManagementPsConfigurationRequestCall”*

5702

5703 *5.2.3.2.2.3 Identifiers*

5704 None.

5705

5706 *5.2.3.2.2.4 Available selectors*

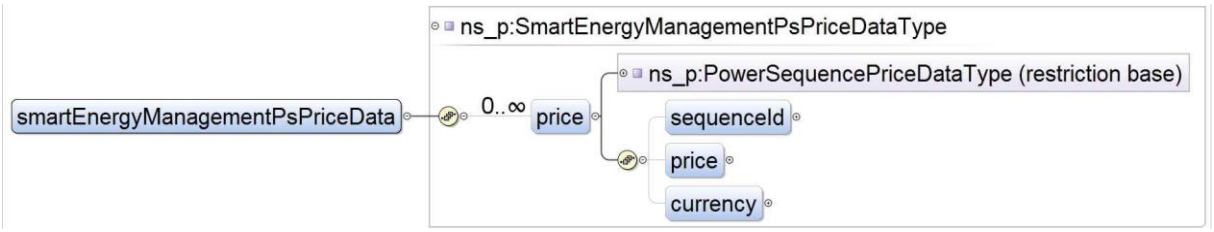
5707 None.

5708

5709 *5.2.3.2.3 smartEnergyManagementPsPriceData*

5710 *5.2.3.2.3.1 General*

5711 A power sequence may have a price, i.e. information on its cost. The complex function
5712 “smartEnergyManagementPsPriceData” of the complex class “SmartEnergyManagementPs” can be
5713 used to describe this information.



5714

5715 *Figure 53: smartEnergyManagementPsPriceData function overview*

5716

5717 *5.2.3.2.3.2 Detailed description of elements*

5718 The complex function “smartEnergyManagementPsPriceData” of the complex class
5719 “SmartEnergyManagementPs” is described in the following table:

Element name	Data range	Explanation
price	List of price information (0..unbounded)	Each occurrence of this element contains information on one power sequence's cost. Note: This element is derived from the PowerSequences class function "powerSequencePriceData".
price. sequenceId	*Identifier "PowerSequenceIdType" (see* Table 339*).*	The sequenceId the calculated price applies for.
price. price	*Common data type "ScaledNumberType". See section 3.10.1.8.*	The price (value) which has been calculated.
price. currency	*Common data type "CurrencyType". See section 3.10.1.19.*	The currency of the price.

Table 160: Description of smartEnergyManagementPsPriceData.

5.2.3.2.3.3 Identifiers

- sequenceId

5.2.3.2.3.4 Available selectors

The selectors definition for function "smartEnergyManagementPsPriceData" is given in the following table:

Element name	Data range	Explanation
smartEnergyManagement PsPriceDataSelectors		
smartEnergyManagement PsPriceDataSelectors. price		Note: This element is derived from the PowerSequences class selectors definition "powerSequencePriceListDataSelectors".
smartEnergyManagement PsPriceDataSelectors. price. sequenceId	*Identifier "PowerSequenceIdType" (see* Table 339*).*	Selector item: The price information of the proper power sequence is selected.

Table 161: Dedicated "selectors" of "smartEnergyManagementPsPriceData".

5.2.3.2.4 smartEnergyManagementPsPriceCalculationRequestCall

5.2.3.2.4.1 General

Similar to the function “powerSequencePriceCalculationRequestCall” of class PowerSequences, the complex function “smartEnergyManagementPsPriceCalculationRequestCall” permits to model a trigger for an external price calculation. I.e. the function is designed for the case that a power sequences server cannot calculate a price for one of its sequences on its own, but knows a power sequences client that is capable to do this. Then, the power sequences server may send a powerSequencePriceCalculationRequestCall with proper data to the client in order to trigger the price calculation at the client. As a result, the client may then write the price to the server. The calculated price can for example be shown on the power sequences server’s device display or stored inside the device to build device internal historical information.

Figure 54: smartEnergyManagementPsPriceCalculationRequestCall function overview

5.2.3.2.4.2 Detailed description of elements

The complex function “smartEnergyManagementPsPriceCalculationRequestCall” of the complex class “SmartEnergyManagementPs” is described in the following table:

Element name	Data range	Explanation
priceCalculationRequest		Note: This element is derived from the PowerSequences class function “powerSequencePriceCalculationRequestCall”.
priceCalculationRequest.sequenceId	*Identifier "PowerSequenceIdType" (see* Table 339*).*	The sequenceId a power sequences server wants to have a price calculated for.

Table 162: Description of complex class function “smartEnergyManagementPsPriceCalculationRequestCall”

5.2.3.2.4.3 Identifiers

None.

5.2.3.2.4.4 Available selectors

None.

5.3 Standard Classes

5.3.1 ActuatorLevel

5.3.1.1 Introduction

Independent from the *ActuatorSwitch* class, the *ActuatorLevel* class enables a user or application to model *LEVEL* commands (*start*, *up*, *percentageAbsolute*, *relative*, ...). This can be used to dim a light, set the speed of an electric motor, etc. Its simple functions can easily be mapped to other technologies.

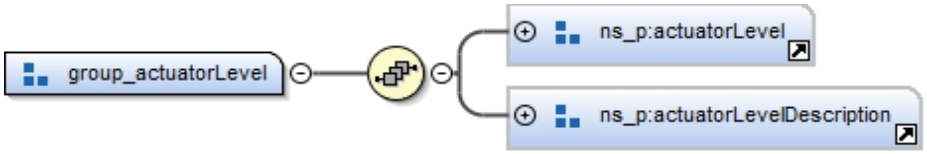


Figure 55: ActuatorLevel function-group overview

5.3.1.2 actuatorLevelData

5.3.1.2.1 General

This function is used for setting the desired operation mode of the node or getting the actual mode respectively.

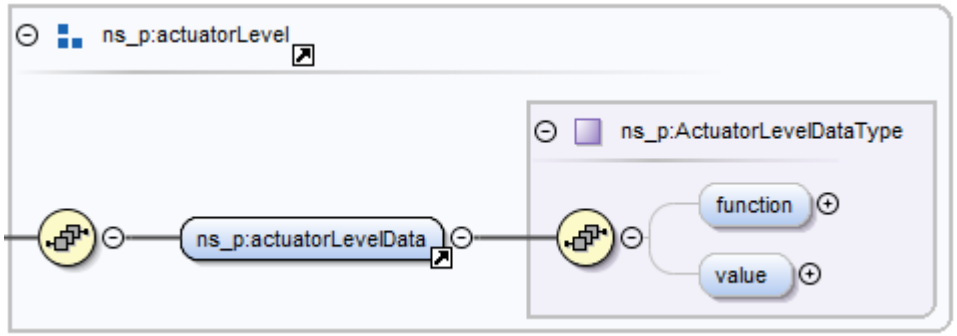


Figure 56: actuatorLevelData function overview

5.3.1.2.2 Detailed description of elements

Element	Type	Description
function	*Union "*ActuatorLevelFctType*":* *- Enum (see* Table 164*):* ActuatorLevelFctEnumType - EnumExtendType (see section 3.10.1.5)	The function contains the mode.
value	*Common data type "ScaledNumberType". See section 3.10.1.8.*	The value is only needed for some functions. E.g. setting *function* to *percentageAbsolute* just makes sense together with a proper value.

Table 163: actuatorLevelData function detailed description of elements

Enumeration **ActuatorLevelFctEnumType**:

Value	Description
start	Actuator starts.
up	Actuator moves up.
down	Actuator moves down.
stop	Actuator stops.
percentageAbsolute	The percentage position / value of the actuator (e.g. 25 %).
percentageRelative	Changes the current value by some percentage value (e.g. move plus 10 percent).
absolute	The absolute position / value of the actuator (e.g. 0.91 m).
relative	Changes the current value by some value (e.g. move plus 0.23 m).

Table 164: Enumeration ActuatorLevelFctEnumType

5.3.1.2.3 Available selectors

None.

5.3.1.2.4 Examples

5.3.1.2.4.1 Write

To dim a light (50 percent brightness), one could send the following XML together with a write classifier:

```
<actuatorLevelData>
    <function>percentageAbsolute</function>
    <value>
        <number>5</number>
        <scale>1</scale>
    </value>
</actuatorLevelData>
```

5.3.1.3 actuatorLevelDescriptionData

5.3.1.3.1 General

The non-changing descriptive data of the *actuatorLevel* class can contain a default unit, among others. This is useful for some *absolute* commands.

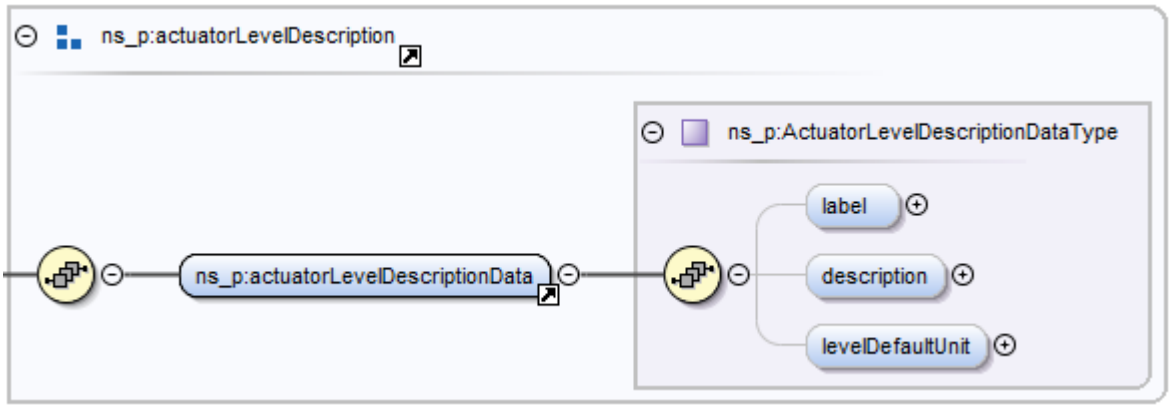


Figure 57: actuatorLevelDescriptionData function overview

5798

5799 *5.3.1.3.2 Detailed description of elements*

Element	Type	Description
label	*Common data type "LabelType". See section 3.10.1.2.*	A short label of the actuator.
description	*Common data type "DescriptionType". See section 3.10.1.3.*	Descriptive information on the actuator.
levelDefaultUnit	*Common data type "UnitOfMeasurementType". See section 3.10.1.17.*	If an actuator can specify a unit for its level-controlling, it can be denoted in this element.

5800 *Table 165: actuatorLevelDescriptionData function detailed description of elements*

5801

5802 *5.3.1.3.3 Available selectors*

5803 None.

5804

5805 *5.3.1.3.4 Examples*

5806 *5.3.1.3.4.1 Read-reply*

5807 To query the description of an actuator level, one could send a read command to the server. The
5808 reply could be e.g.:

```
5809 <actuatorLevelDescriptionData>
5810     <label>ActuatorLevel TestDevice</label>
5811     <description>Device for testing SPINE compliance.</description>
5812     <levelDefaultUnit>W</levelDefaultUnit>
5813 </actuatorLevelDescriptionData>
```

5814

5815 **5.3.2 ActuatorSwitch**

5816 ***5.3.2.1 Introduction***

5817 Basic on/off operations on a simple actuator can be modelled with the *ActuatorSwitch* class.
5818 Whether the function turns a device itself *on* or *off*, or whether it switches a specific feature,
5819 depends on the implementation. E.g. one could model the super freeze program of a freezer using
5820 *ActuatorSwitch* class. An *on* command would then activate the super freeze program and an *off*
5821 command would deactivate it. This example shall just give an idea how *ActuatorSwitch* can be used
5822 for more purposes than only turning devices on and off.

5823 Please note a device's different features may be modelled with separate implementations of this
5824 class, each associated to a different node address. Continuing with the example above, a "defrost"
5825 function may be modelled on another node address with *ActuatorSwitch*.

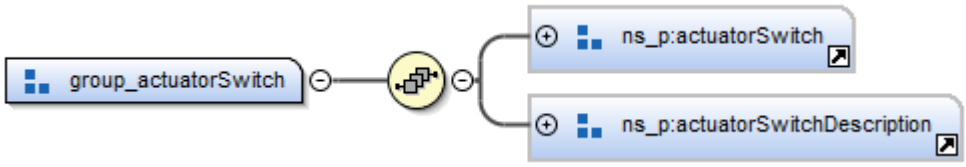


Figure 58: ActuatorSwitch function-group overview

5.3.2.2 actuatorSwitchData

5.3.2.2.1 General

The *actuatorSwitchData* function models the functionality described in section 5.3.2.1.

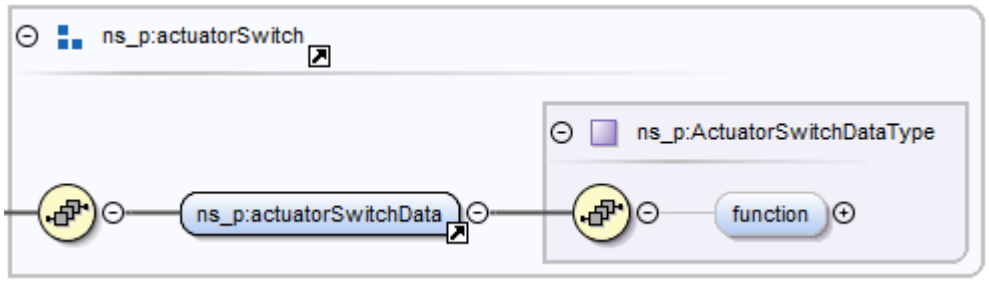


Figure 59: actuatorSwitchData function overview

5.3.2.2.2 Detailed description of elements

Element	Type	Description
function	*Union* "ActuatorSwitchFctType"*:* *- Enum (see* Table 167*):* ActuatorSwitchFctEnumType - EnumExtendType (see section 3.10.1.5)	The only element in this function to model an *on*, *off* or *toggle* command.

Table 166: actuatorSwitchData function detailed description of elements

Enumeration **ActuatorSwitchFctEnumType**:

Value	Description
on	Turn on.
off	Turn off.
toggle	Toggle between on and off.

Table 167: Enumeration ActuatorSwitchFctEnumType

5.3.2.2.3 Available selectors

None.

5.3.2.2.4 Examples

5.3.2.2.4.1 Write

This example could turn on a device upon receipt with a write classifier:


```
<actuatorSwitchData>
    <function>on</function>
</actuatorSwitchData>
```

5.3.2.3 actuatorSwitchDescriptionData

5.3.2.3.1 General

The (typically) non-changing information on the actuator is modelled with this function.

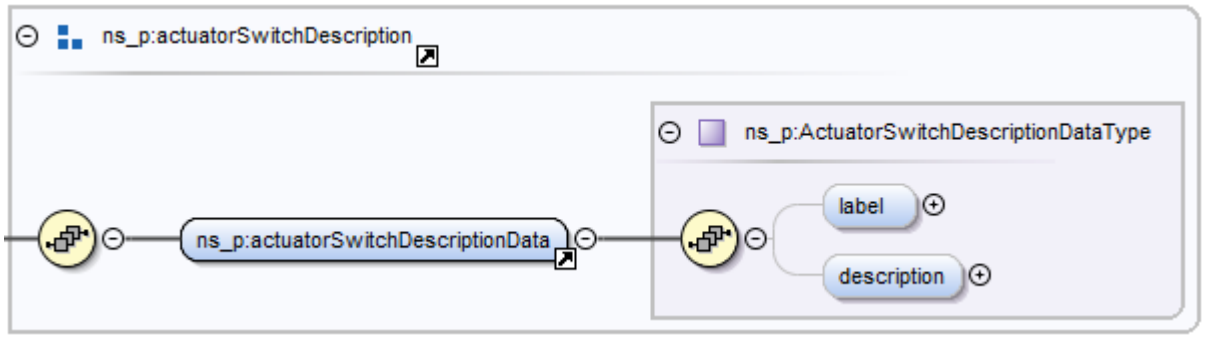


Figure 60: actuatorSwitchDescription function overview

5.3.2.3.2 Detailed description of elements

Element	Type	Description
label	*Common data type "LabelType". See section 3.10.1.2.*	A short label of the actuator.
description	*Common data type "DescriptionType". See section 3.10.1.3.*	Descriptive information on the actuator.

Table 168: actuatorSwitchDescription function detailed description of elements

5.3.2.3.3 Available selectors

None.

5.3.2.3.4 Examples

5.3.2.3.4.1 Read-reply

One would like to know the contents of the label and description of a device. Therefore, a standard read-command is sent to the server. Then a proper reply could be:

```
<actuatorSwitchDescriptionData>
    <label>SmartPlug_1</label>
    <description>SmartPlug Test Installation 01.04.2012</description>
</actuatorSwitchDescriptionData>
```


5.3.3 Alarm

5.3.3.1 Introduction

There are several reasons for generating an alarm. This might be due to exceeding (in a positive or negative way) a threshold (modelled with the Threshold class, see section 5.3.26), or because of some other occurrence that provokes the generating of an alarm (modelled in a later version of this specification). The Alarm class provides the needed data model.

Figure 61: Alarm function-group overview

5.3.3.2 alarmListData

5.3.3.2.1 General

This function models all the needed data for an alarm. Depending on the type of alarm, different elements are used (see Feature Type "Alarm" for more details; section 4.3.3). It can be used to notify an alarm or as a reply to a read command.

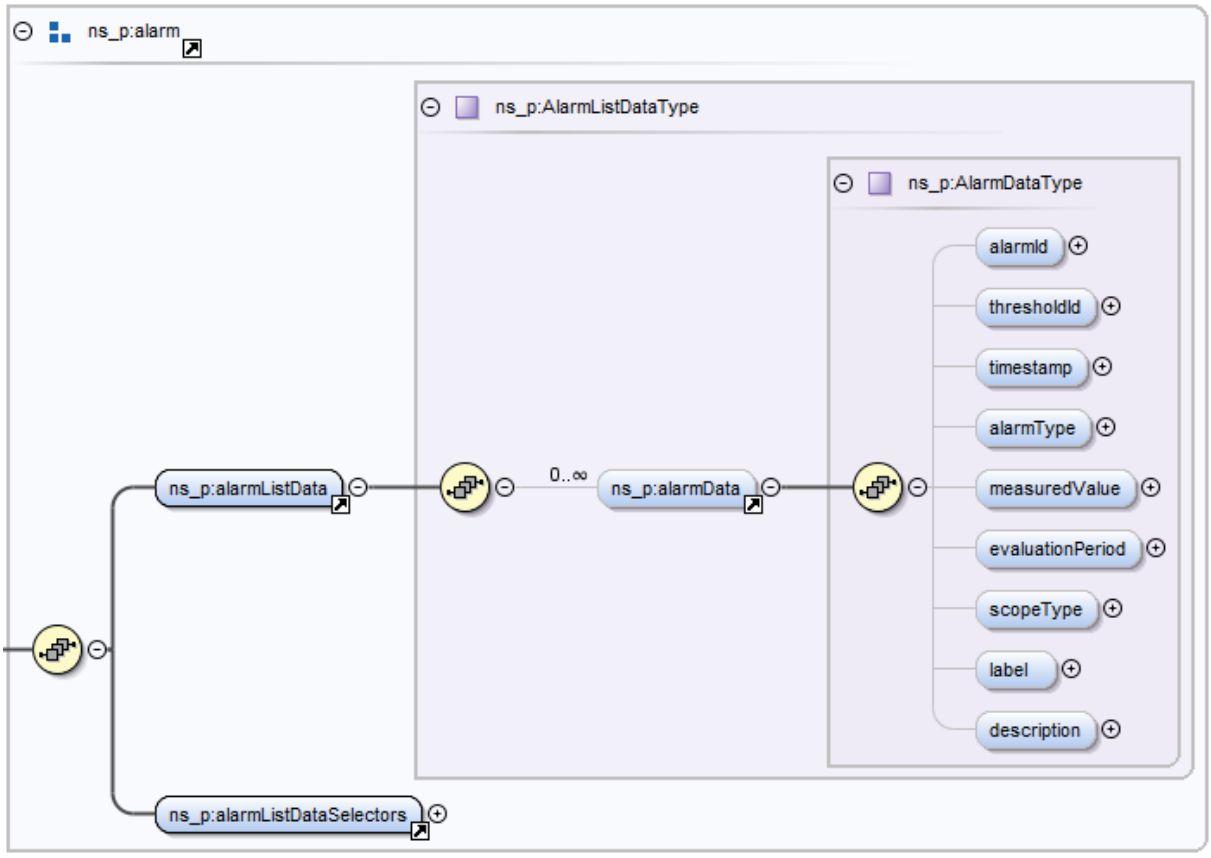


Figure 62: alarmListData function overview

5.3.3.2.2 Detailed description of elements

Element	Type	Description

alarmId	*Identifier "AlarmIdType" (see* Table 339*).*	Identifier of the alarm.
thresholdId	*Identifier "ThresholdIdType" (see* Table 339*).*	If the alarm relates to a threshold, the corresponding ID can be stated here.
timestamp	*Common data type "AbsoluteOrRelativeTimeType". See section 3.10.2.3.*	The time of creation of the alarm.
alarmType	*Union "*AlarmTypeType*":* *- Enum (see Table 170):* AlarmTypeEnumType - EnumExtendType (see section 3.10.1.5)	The type of alarm is given here.
measuredValue	*Common data type "ScaledNumberType". See section 3.10.1.8.*	The value that was measured which exceeded the threshold is denoted in this element.
evaluationPeriod	*Common data type "TimePeriodType". See section 3.10.2.1.*	If a time period can be stated in which
scopeType	*Common data type "ScopeTypeType". See section 3.10.1.21.*	Specifies a more detailed meaning of the alarm (e.g. exceeding of a threshold).
label	*Common data type "LabelType". See section 3.10.1.2.*	A short label of the alarm.
description	*Common data type "DescriptionType". See section 3.10.1.3.*	Descriptive information on the alarm.

Table 169: alarmListData function detailed description of elements

Enumeration **AlarmTypeEnumType**:

Value	Description
alarmCancelled	A previously generated alarm is cancelled (e.g. because the value is in a "normal" range again).
underThreshold	The value is under a related threshold.
overThreshold	The value is over a related threshold.

Table 170: Enumeration AlarmTypeEnumType

5.3.3.2.3 Available selectors

- alarmId
- scopeType

5.3.3.2.4 Examples

5.3.3.2.4.1 Notify

When a threshold is reached, the server would send a notification to all subscribed clients, with a content like the following (the “filter/partial” part to announce the “restricted function exchange is simplified by “...” just to improve the readability”):

. . .


```
<alarmListData>
    <alarmData>
        <alarmId>4</alarmId>
        <thresholdId>2</thresholdId>
        <timestamp>2015-12-01T18:27:32.4Z</timestamp>
        <alarmType>overValue</alarmType>
        <measuredValue>
            <number>711</number>
            <scale>-1</scale>
        </measuredValue>
        <label>Temperature reached</label>
        <description>Desired tea temperature reached by water
boiler.</description>
    </alarmData>
</alarmListData>
```

5.3.4 Bill

5.3.4.1 Introduction

In many cases bill will be writeable. To allow the client to perform a single write, billData includes also a lot of descriptive elements. Additionally this concept avoids that data within billDescriptionData is changed by client and server. Therefore if a bill is written by a client, only billData is written by a client, while the other billConstraints and billDescription are never writeable.

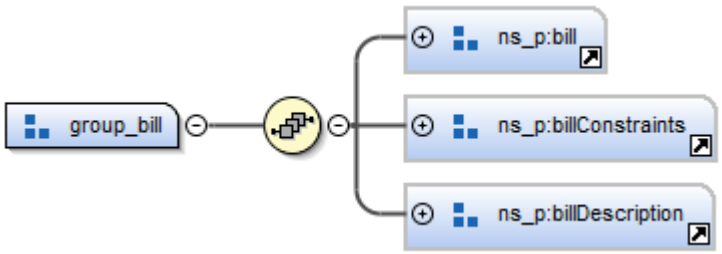


Figure 63: Bill function-group overview

5.3.4.2 billListData

5.3.4.2.1 General

billData holds the bill data, including descriptive elements like currency, unit and billType. This is necessary so the client can write a bill with a single write.

A billData has a "total" element, including total values and corresponding costs in its sub-values. Additionally positions of the total data can be provided, to describe the composition of the total data. To link total costs and values in a position the valueId and costId are used within a billData. The billId allows to identify the overall bill in other functions of bill as well as outside of the bill resource.

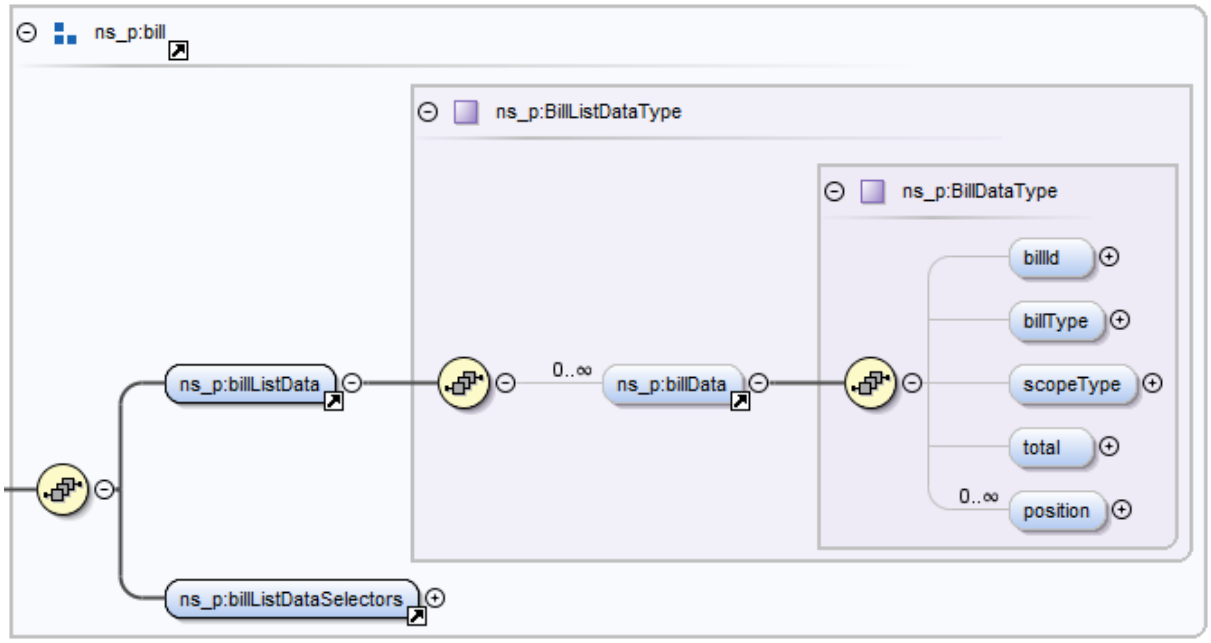


Figure 64: billListData function overview

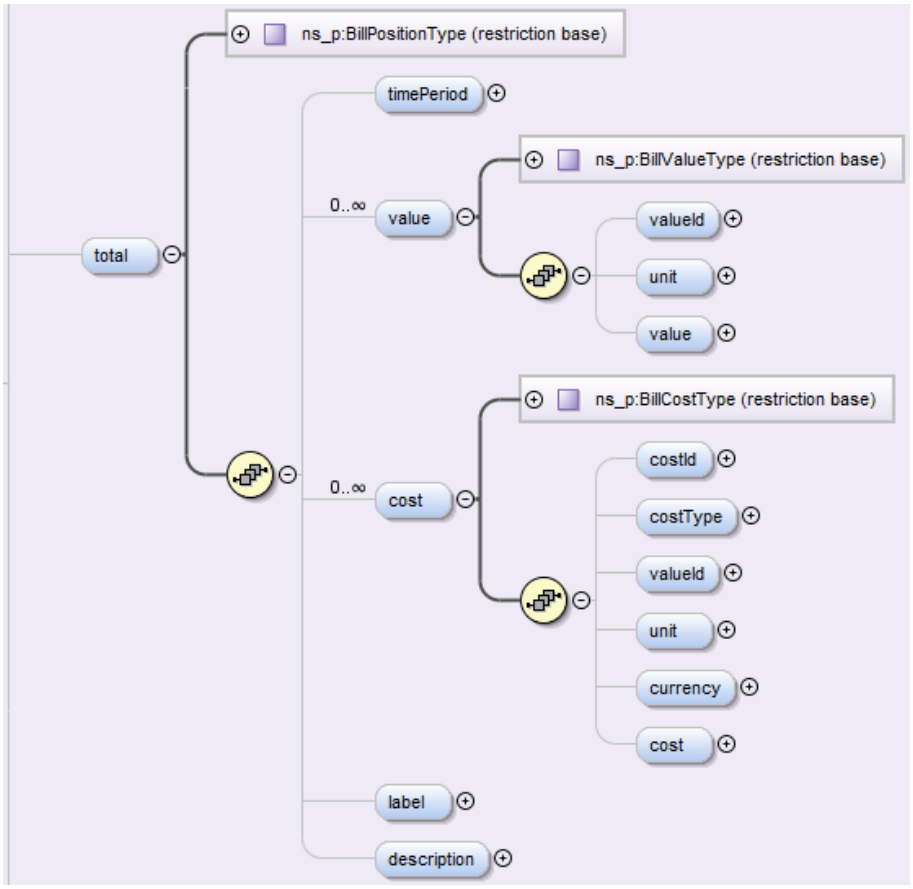
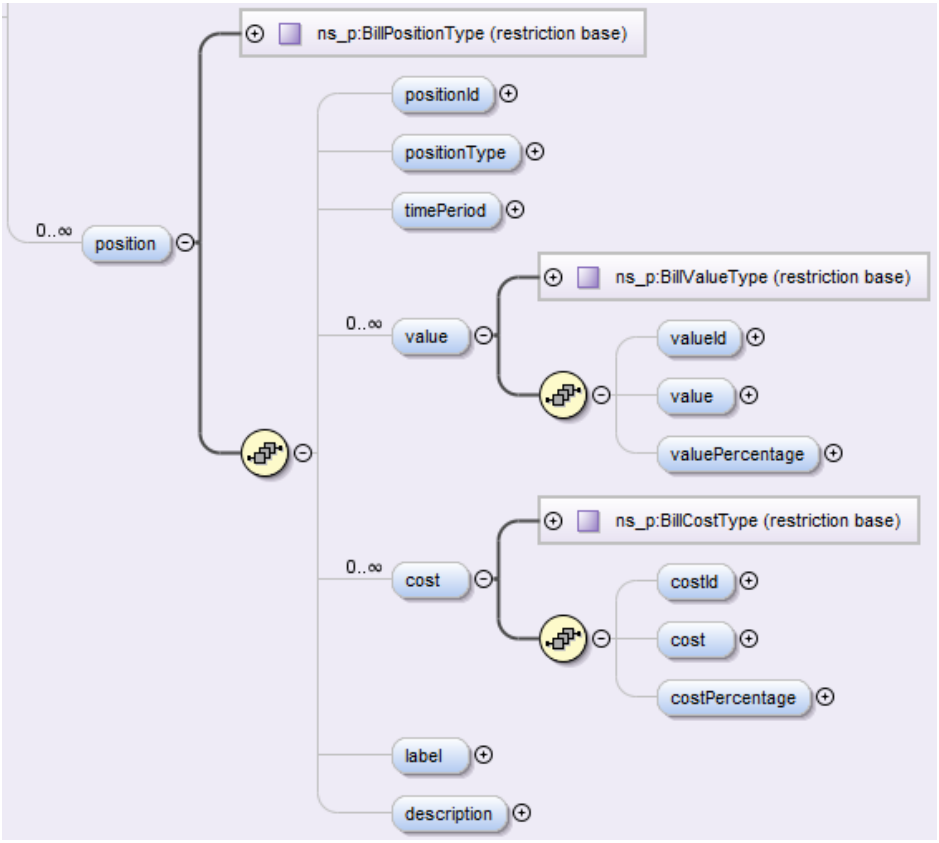


Figure 65: billListData function detail 1, "total"



5940

5941 *Figure 66: billListData function detail 2, "position"*

5942

5943 *5.3.4.2.2 Detailed description of elements*

Element	Type	Description
billId	*Identifier "BillIdType" (see* Table 339*).*	Allows the identification of a bill. Allows linking of the different functions to the same bill. Allows linking of the bill within other features that are placed in the same entity.
billType	*Union "*BillTypeType*":* *- Enum (see* Table 172*):* BillTypeEnumType - EnumExtendType (see section 3.10.1.5)	The type of the bill.
scopeType	*Common data type "ScopeTypeType". See section 3.10.1.21.*	Specifies a more detailed meaning of the bill.
total	*Restriction of "*BillPositionType*"*	Total information of the bill (in contrast to the positions, see below).
total. timePeriod	*Common data type "TimePeriodType". See section 3.10.2.1.*	Allows to define a time, or time period for a bill.
total. value (list)	*Restriction of "*BillValueType*"*	Different values can be listed for a bill. E.g. energy values or just a number of items.
total. value. valueId	*Identifier "BillValueIdType" (see* Table 339*).*	Allows the identification of a value within the bill.

total. value. unit	*Common data type "UnitOfMeasurementType". See section 3.10.1.17.*	Allows definition of a unit for the value.
total. value. value	*Common data type "ScaledNumberType". See section 3.10.1.8.*	Allows to express the total amount that was bought or sold of a certain commodity during the time or time period given in total.timePeriod.
total. cost (list)	*Restriction of "*BillCostType*"*	Different types of costs can be listed for a value. E.g. monetary costs like a price as well as environmental costs as CO2 emmission.
total. cost. costId	*Identifier "BillCostIdType" (see* Table 339*).*	Allows identification of costs within bill.
total. cost. costType	*Union "*BillCostTypeType*":* *- Enum (see* Table 173*):* BillCostTypeEnumType - EnumExtendType (see section 3.10.1.5)	Allows to describe the type of costs.
total. cost. valueId	*Identifier "BillValueIdType" (see* Table 339*).*	Allows to link cost to a value.
total. cost. unit	*Common data type "UnitOfMeasurementType". See section 3.10.1.17.*	Costs that are not monetary may need to describe a unit (.g. m^3).
total. cost. currency	*Common data type "CurrencyType". See section 3.10.1.19.*	Monetary costs may need to describe a currency.
total. cost. cost	*Common data type "ScaledNumberType". See section 3.10.1.8.*	Total costs generated during the time or time period given in total.timePeriod.
total. label	*Common data type "LabelType". See section 3.10.1.2.*	Allows to express a user friendly label for the bill.
total. description	*Common data type "DescriptionType". See section 3.10.1.3.*	Allows to express a user friendly description for the bill.
position (list)	*Restriction of "*BillPositionType*"*	For each total data different positions can be assigned that describe the composition of the total data.
position. positionId	*Identifier "BillPositionIdType" (see* Table 339*).*	Identifier of a position within a specific bill.
position. positionType	*Union "*BillPositionTypeType*":* *- Enum (see* Table 174*):* BillPositionTypeEnumType - EnumExtendType (see section 3.10.1.5)	Allows to describe the type of the a position E.g. different types of energy consumption like self produced or electricity grid energy.
position. timePeriod	*Common data type "TimePeriodType". See section 3.10.2.1.*	Allow to define a time, or time period for a position if the time or time period is different from the overall time period described in total.
position. value (list)	*Restriction of "*BillValueType*"*	Different values can be listed for each position.
position. value. valueId	*Identifier "BillValueIdType" (see* Table 339*).*	Identifier of a value within a specific bill (see also "total. value. valueId").

position. value. value	*Common data type "ScaledNumberType". See section 3.10.1.8.*	Allows to describe a value for each position. This can be helpful if some positions have negative while other positions have positive values, as this would can not be express with valuePercentage.
position. value. valuePercentage	*Common data type "ScaledNumberType". See section 3.10.1.8.*	Percentage of the total value (see "total").
position. cost (list)	*Restriction of "*BillCostType*"*	Different costs can be listed for each position
position. cost. costId	*Identifier "BillCostIdType" (see* Table 339*).*	Identifier of a cost within a specific bill (see also "total. cost. costId").
position. cost. cost	*Common data type "ScaledNumberType". See section 3.10.1.8.*	The actual costs for this position. This can be helpful if some positions have negative while other positions have positive costs, as this can not be expressed with costPercentage.
position. cost. costPercentage	*Common data type "ScaledNumberType". See section 3.10.1.8.*	Percentage of the total costs (see "total").
position. label	*Common data type "LabelType". See section 3.10.1.2.*	Allows to provide a user friendly label for each bill position.
position. description	*Common data type "DescriptionType". See section 3.10.1.3.*	Allows to provide user friendly information for each bill position.

5944 *Table 171: billListData function detailed description of elements*

5945 Enumeration **BillTypeEnumType**:

Value	Description
chargingSummary	This bill type allows to summarize a charging cycle. In most cases it is used to communicate the total energy that was charged and the total costs, as well as different positions for different types of energy. The summary may also be sent during charging, also multiple times. This allows to keep the customer updated during the charging process.

5946 *Table 172: Enumeration BillTypeEnumType*

5947 Enumeration **BillCostTypeEnumType**:

Value	Description
absolutePrice	Used for absolute prices.
relativePrice	Used for relative prices, e.g. a price that shows how much cheaper the costs are related to the average tariff in place.
co2Emission	In this case the cost value is used to express how much CO2 emission was emitted.
renewableEnergy	In this case the cost value is used to express how much renewable energy was used.
radioactiveWaste	In this case the cost value is used to express how much radioactive waste was generated.

5948 *Table 173: Enumeration BillCostTypeEnumType*

5949 Enumeration **BillPositionTypeEnumType**:

Value	Description

gridElectricEnergy	This position type specifically describes the electricity grid energy portion of the total energy.
selfProducedElectricEnergy	This position type specifically describes the self produced energy portion of the total energy.

Table 174: Enumeration BillPositionTypeEnumType

5.3.4.2.3 Available selectors

- billId
- scopeType

5.3.4.2.4 Examples

5.3.4.2.4.1 Notify

When there is an update for a specific bill, a server would send a notification to all subscribed clients that could look like this:

```
...
<billListData>
    <billData>
        <billId>1</billId>
        <billType>chargingSummary</billType>
        <total>
            <timePeriod>
                <startTime>2017-10-24T13:14:31.0Z</startTime>
                <endTime>2017-10-24T14:11:43.0Z</endTime>
            </timePeriod>
            <value>
                <valueId>1</valueId>
                <unit>Wh</unit>
                <value>
                    <number>35</number>
                    <scale>3</scale>
                </value>
            </value>
            <cost>
                <costId>1</costId>
                <costType>absolutePrice</costType>
                <valueId>1</valueId>
                <currency>EUR</currency>
                <cost>
                    <number>96</number>
                    <scale>-1</scale>
                </cost>
            </cost>
        </total>
        <position>
            <positionId>1</positionId>
            <positionType>gridElectricEnergy</positionType>
            <timePeriod>
                <startTime>2017-10-24T13:14:31.0Z</startTime>
                <endTime>2017-10-24T14:11:43.0Z</endTime>
            </timePeriod>
            <value>
                <valueId>1</valueId>
                <valuePercentage>
```



```
                    <number>73</number>
                    <scale>0</scale>
                </valuePercentage>
            </value>
            <cost>
                <costId>1</costId>
                <costPercentage>
                    <number>95</number>
                    <scale>0</scale>
                </costPercentage>
            </cost>
        </position>
        <position>
            <positionId>2</positionId>
            <positionType>selfProducedElectricEnergy</positionType>
            <timePeriod>
                <startTime>2017-10-24T13:14:31.0Z</startTime>
                <endTime>2017-10-24T14:11:43.0Z</endTime>
            </timePeriod>
            <value>
                <valueId>1</valueId>
                <valuePercentage>
                    <number>27</number>
                    <scale>0</scale>
                </valuePercentage>
            </value>
            <cost>
                <costId>1</costId>
                <costPercentage>
                    <number>5</number>
                    <scale>0</scale>
                </costPercentage>
            </cost>
        </position>
    </billData>
</billListData>
```

5.3.4.3 billConstraintsListData

5.3.4.3.1 General

The billConstraintsListData function allows a server to define certain constraints for each linked billListData list entry. This is especially important if a particular billListData list entry is writeable and allows the server to communicate the corresponding constraints before a client performs a write. The client can match the server constraints with own constraints and perform a corresponding write command.

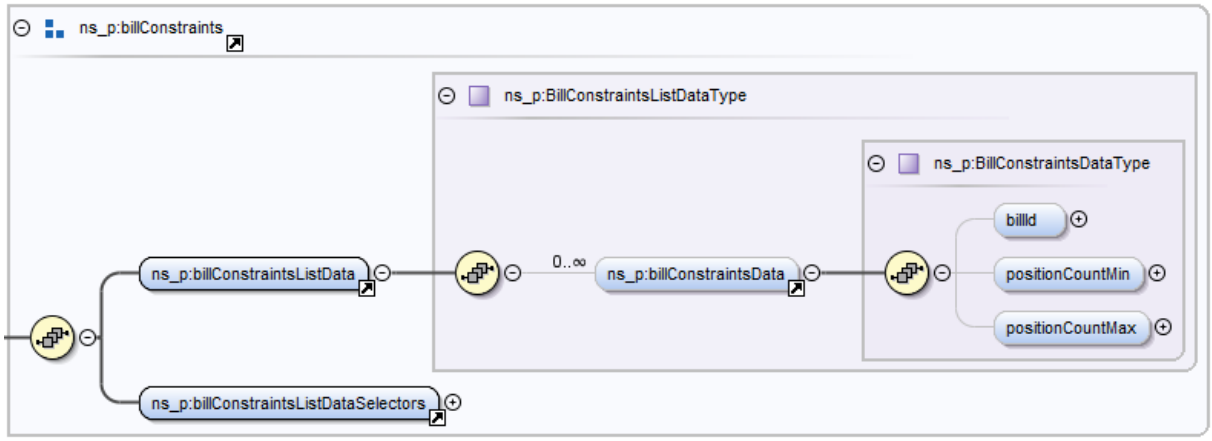


Figure 67: billConstraintsListData function overview

5.3.4.3.2 Detailed description of elements

Element	Type	Description
billId	*Identifier "BillIdType" (see* Table 339*).*	Allows the identification of a bill. Allows linking of the different functions to the same bill. Allows linking of the bill within other features that are placed in the same entity.
positionCountMin	*Simple type "BillPositionCountType" (restriction of "BillPositionIdType" (see* Table 339*)).*	Minimum amount of possible positions within the specified bill.
positionCountMax	*Simple type "BillPositionCountType" (restriction of "BillPositionIdType" (see* Table 339*)).*	Maximum amount of possible positions within the specified bill.

Table 175: billConstraintsListData function detailed description of elements

5.3.4.3.3 Available selectors

- billId

5.3.4.3.4 Examples

5.3.4.3.4.1 Read-reply

If one wants to know the constraints of a specific bill, he could send the following read command:

```
<function>billConstraintsListData</function>
<filter>
    <cmdControl>
        <partial/>
    </cmdControl>
    <billConstraintsListDataSelectors>
        <billId>1</billId>
    </billConstraintsListDataSelectors>
</filter>
<billConstraintsListData/>
```


The reply could look like this:

```
<function>billConstraintsListData</function>
<filter>
    <cmdControl>
        <partial/>
    </cmdControl>
</filter>
<billConstraintsListData>
    <billConstraintsData>
        <billId>1</billId>
        <positionCountMin>2</positionCountMin>
        <positionCountMax>4</positionCountMax>
    </billConstraintsData>
</billConstraintsListData>
```

5.3.4.4 billDescriptionListData

5.3.4.4.1 General

The billDescriptionListData function is especially important if a linked billListData list entry is writeable and allows a server to define if a bill is writeable and which billTypes are supported for the corresponding billId, so that a client can perform corresponding writes. Additionally, the server can request updates from a client with the updateRequired flag.

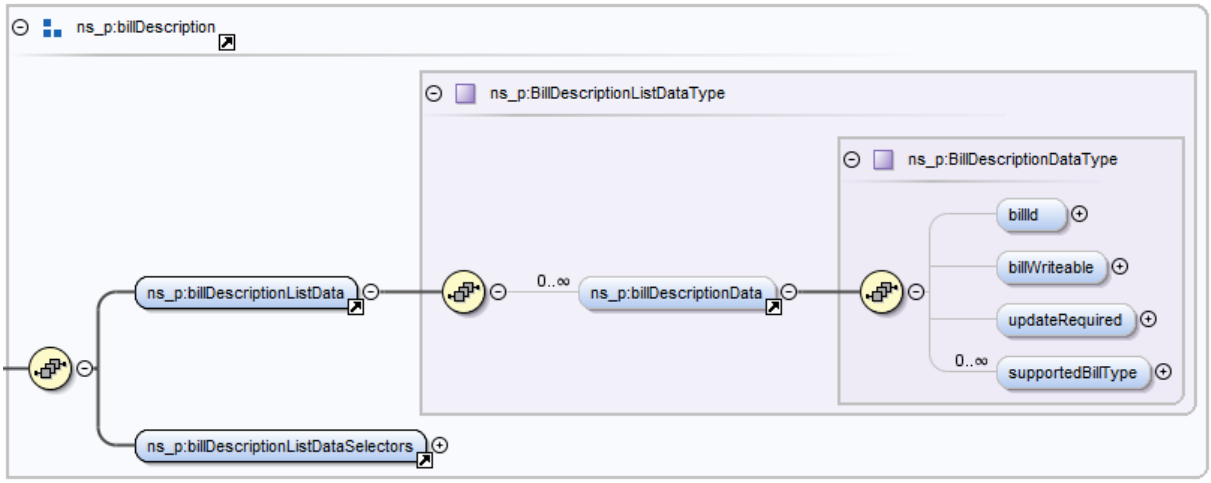


Figure 68: billDescriptionListData function overview

5.3.4.4.2 Detailed description of elements

Element	Type	Description
billId	*Identifier "BillIdType" (see* Table 339*).*	Allows the identification of a bill. Allows linking of the different functions to the same bill. Allows linking of the bill within other features that are placed in the same entity.
billWriteable	xs:boolean (W3C standard type)	Indicates whether the linked billData entry is writeable.
updateRequired	xs:boolean (W3C standard type)	Indicates whether the server requests an update for the bill.

supportedBillType (list)	*Union* "BillTypeType"*:* - *Enum (see* Table 172*):* BillTypeEnumType - EnumExtendType (see section 3.10.1.5)	List of all supported types for this billId.

Table 176: billDescriptionListData function detailed description of elements

5.3.4.4.3 Available selectors

- billId

5.3.4.4.4 Examples

5.3.4.4.4.1 Read-reply

If one is interested in the descriptions of all bills, he could send a standard read command to the server. The reply could look like this:

```
<billDescriptionListData>
    <billDescriptionData>
        <billId>1</billId>
        <billWriteable>true</billWriteable>
        <supportedBillType>chargingSummary</supportedBillType>
    </billDescriptionData>
</billDescriptionListData>
```

5.3.5 BindingManagement

5.3.5.1 Introduction

Bindings are used to simplify communication between devices. Based on matching features, finding suitable communication partners is a lot easier than via manual configuration only.

Each binding exists between a binding client and a binding server. Only clients may initiate new bindings. After a successful establishment of a binding, the binding client may do actions on the binding server, depending on the bound functionality. E.g. a binding server accepts load management configurations by the client and updates the client when changes in its parameters occur. The exact definition of rules and proceedings is out of scope of this documentation.

The *BindingManagement* class provides functions for establishing, reading and deleting bindings.

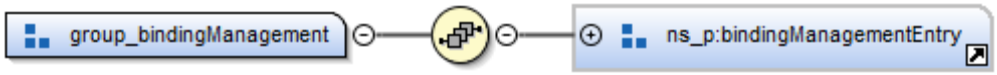


Figure 69: BindingManagement function-group overview (data)

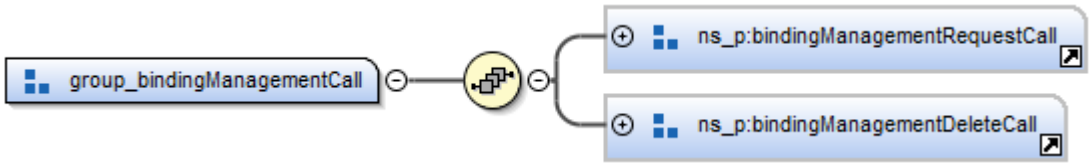


Figure 70: BindingManagement function-group overview (call)

5.3.5.2 *bindingManagementEntryListData*

5.3.5.2.1 General

All existing bindings can be listed in the *bindingManagementEntryListData* function. It may be used to store the entries in the device itself. If another device asks for the list of bindings, only the bindings with the requesting device should be replied.

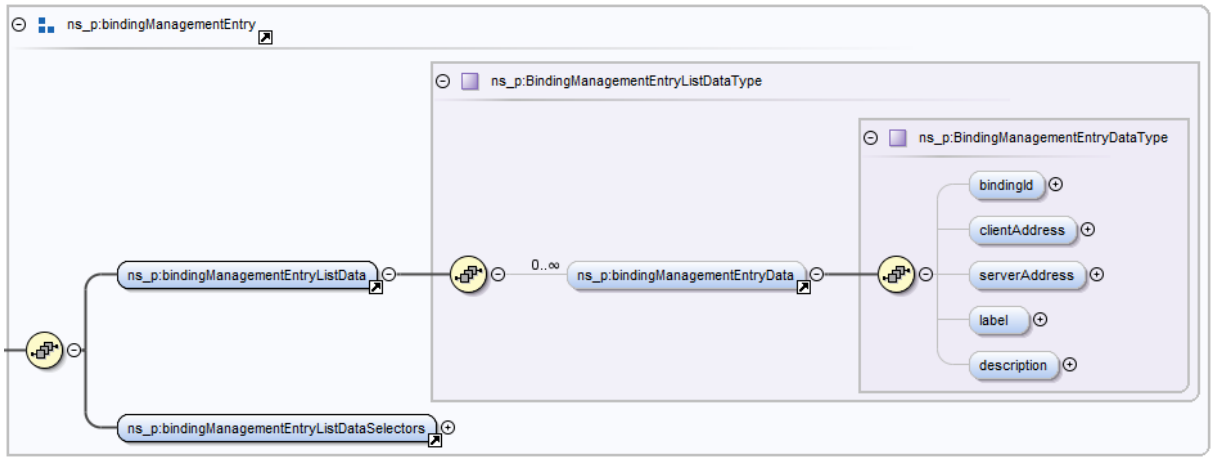


Figure 71: bindingManagementEntryListData function overview

5.3.5.2.2 Detailed description of elements

Element	Type	Description
bindingId	*Identifier "BindingIdType" (see* Table 339*).*	Identifier of the binding.
clientAddress	*Common data type "FeatureAddressType". See section 3.10.3.6.*	Address of the feature with the role *client* in this binding.
serverAddress	*Common data type "FeatureAddressType". See section 3.10.3.6.*	Address of the feature with the role *server* in this binding.
label	*Common data type "LabelType". See section 3.10.1.2.*	A short label of the binding.
description	*Common data type "DescriptionType". See section 3.10.1.3.*	Descriptive information on the binding.

Table 177: bindingManagementEntryListData function detailed description of elements

5.3.5.2.3 Available selectors

- bindingId
- clientAddress
- serverAddress

5.3.5.2.4 Examples

5.3.5.2.4.1 Read-reply

If one is interested in all bindings that are active between the own device and another one, he could send a standard read function to the device, which could reply something like this:

```
<bindingManagementEntryListData>
    <bindingManagementEntryData>
        <bindingId>0</bindingId>
        <clientAddress>
            <device>d:_i:46925_CP\_pqtl-276381626831726378123568l3</device>
            <entity>2</entity>
            <feature>1</feature>
        </clientAddress>
        <serverAddress>
            <device>d:_i:46925_CP\_arqf-64986751356897215468975468</device>
            <entity>1</entity>
            <feature>1</feature>
        </serverAddress>
        <label>SwitchLight_01</label>
    </bindingManagementEntryData>
    <bindingManagementEntryData>
        <bindingId>4</bindingId>
        <clientAddress>
            <device>d:_i:46925_CP\_pqtl-276381626831726378123568l3</device>
            <entity>2</entity>
            <feature>4</feature>
        </clientAddress>
        <serverAddress>
            <device>d:_i:46925_CP\_arqf-64986751356897215468975468</device>
            <entity>1</entity>
            <feature>2</feature>
        </serverAddress>
        <label>SwitchLight_02</label>
    </bindingManagementEntryData>
</bindingManagementEntryListData>
```

5.3.5.3 bindingManagementRequestCall

5.3.5.3.1 General

To initiate a new binding, an entity capable of node management, sends a *bindingManagementRequestCall* to an entity (on another device that should become the binding partner) capable of node management, too, with the payload element *clientAddress* set to one of its features with the role *client* and the payload element *serverAddress* set to one of the other device's feature address with the role *server*. The receiving device checks whether it will accept or decline the request and responds with an acknowledgement message (see [ProtocolSpecification], section "Acknowledgement message") with the *resultData* function and an according *errorNumber* set to 0 (accept binding request) or some other value (definition of the error code is out of scope of this documentation; decline binding request).

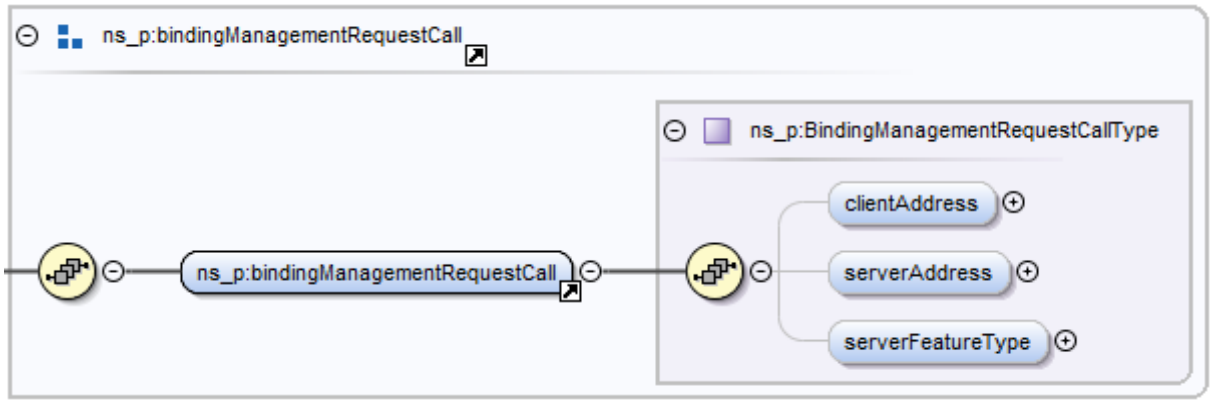


Figure 72: bindingManagementRequestCall function overview

5.3.5.3.2 Detailed description of elements

Element	Type	Description
clientAddress	*Common data type "FeatureAddressType". See section 3.10.3.6.*	Address of the feature with the role *client* in the binding request.
serverAddress	*Common data type "FeatureAddressType". See section 3.10.3.6.*	Address of the feature with the role *server* in the binding request.
serverFeatureType	*Common data type "FeatureTypeType". See section 3.10.1.29.*	The *Feature Type* of the server feature. Stated for verification.

Table 178: bindingManagementRequestCall function detailed description of elements

5.3.5.3.3 Available selectors

None.

5.3.5.3.4 Examples

5.3.5.3.4.1 Call

If a new binding shall be initiated, the client device could send a command like this to the server device:

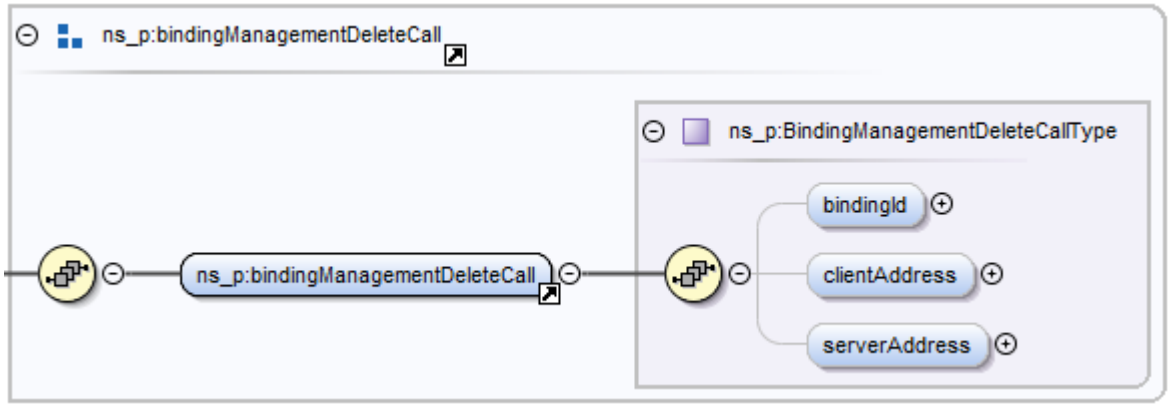
```
<bindingManagementRequestCall>
    <clientAddress>
        <device>d:_i:46925_CP\_pqtl-27638162683172637812356813</device>
        <entity>2</entity>
        <feature>1</feature>
    </clientAddress>
    <serverAddress>
        <device>d:_i:46925_CP\_arqf-64986751356897215468975468</device>
        <entity>1</entity>
        <feature>1</feature>
    </serverAddress>
    <serverFeatureType>ActuatorSwitch</serverFeatureType>
</bindingManagementRequestCall>
```


6210

6211 ### *5.3.5.4 bindingManagementDeleteCall*

6212 #### *5.3.5.4.1 General*

6213 A device may delete an existing binding with itself and another device. Therefore, this call is used. It
6214 may be used by clients and servers.

6215 

6216 *Figure 73: bindingManagementDeleteCall function overview*

6217

6218 #### *5.3.5.4.2 Detailed description of elements*

Element	Type	Description
bindingId	*Identifier "BindingIdType" (see* Table 339*).*	Identifier of the binding.
clientAddress	*Common data type "FeatureAddressType". See section 3.10.3.6.*	Address of the feature with the role *client* in this binding.
serverAddress	*Common data type "FeatureAddressType". See section 3.10.3.6.*	Address of the feature with the role *server* in this binding.

6219 *Table 179: bindingManagementDeleteCall function detailed description of elements*

6220

6221 #### *5.3.5.4.3 Available selectors*

6222 None.

6223

6224 #### *5.3.5.4.4 Examples*

6225 ##### *5.3.5.4.4.1 Call*

6226 If an existing binding (with the bindingId "2") shall be deleted, one device could send a command like
6227 this to the other device:

```
6228 <bindingManagementDeleteCall>
6229     <bindingId>2<bindingId>
6230     <clientAddress>
6231         <device>d:_i:46925_CP\_pqtl-27638162683172637812356813</device>
6232         <entity>2</entity>
```



```
            <feature>1</feature>
        </clientAddress>
        <serverAddress>
            <device>d:_i:46925_CP\_arqf-64986751356897215468975468</device>
            <entity>1</entity>
            <feature>1</feature>
        </serverAddress>
</bindingManagementDeleteCall>
```

5.3.6 DataTunneling

5.3.6.1 Introduction

DataTunneling provides a mechanism to submit data of proprietary protocols using the SPINE data models (esp. addressing other nodes). However, the nature of proprietary definitions imposes restrictions:

In the example below an XML makes use of SPINE dataTunneling sub-elements "purposeId", "channelId", "sequenceId", and "payload". Unless stated otherwise, the term “payload” refers to the sub-element of "dataTunnelingCall" within this section (in contrast to the definition of “payload” as sub-element of “datagram” as defined in document [ProtocolSpecification]).

Please note that the use of the above mentioned elements will always be proprietary. The SPINE specification will never define any specific protocol for dataTunneling.

To put it in other words, the SPINE data tunnelling model is not intended for any kind of flexible extension of the SPINE classes. This is mainly because a so-called technology specific mapping could hardly be defined in a standardized way. Another reason is that it shall always be transparent that data tunnelling contains proprietary data. This should not be considered a limitation of SPINE as the EEBus Initiative e.V. committees may define specific classes with functions (like messagingListData, timeInformationData, e.g.) including mappings for specific protocols, if need be.

Figure 74: DataTunneling function-group overview (call)

5.3.6.2 dataTunnelingCall

5.3.6.2.1 General

In general, several conditions have to be fulfilled in order to make use of SPINE data tunnelling. This shall be explained with an example:

A manufacturer "X" of a device "A" created a proprietary extension of a communications protocol (e.g. KNX, ZigBee, LON, ... – we call it "CP" here) in order to configure device "A" with special parameters. We assume there is no specific SPINE function defined so far for these parameters. There is also no mapping defined for it.

Another manufacturer "Y" creates a device "B" that runs "SPINE applications" (i.e. applications communicating with an internal interface using XMLs of the SPINE data models). The device "B" can as well communicate via "CP".

Now, manufacturer "Y" wants to make use of the proprietary configuration of device "A". Therefore, manufacturer "Y" needs to implement the proprietary extension of manufacturer "X"'s communications protocol as well. I.e. the "CP" implementation of device "B" has to be extended properly. Furthermore, manufacturer "Y" needs to define how the proprietary protocol shall be represented as SPINE data tunnelling towards the "SPINE applications" running on device "B" (i.e. define the proprietary mapping) and of course needs to implement this mapping. Finally, the applications need to implement the recognition and use of the corresponding SPINE data tunnelling XMLs.

The scenario above may look "unattractive" or even raise concerns. However, it shall be mentioned that proprietary extensions of standardized protocols are not unusual. This practice shall not be judged here.

In the aforementioned scenario it is manufacturer "B" who implements a number of extensions in order to comply with the extensions of device "A". But it could also be possible that manufacturer "A" already provided some definitions and implementation for other manufacturers. Then, manufacturer "B" could integrate this solution and potentially implements less by its own.

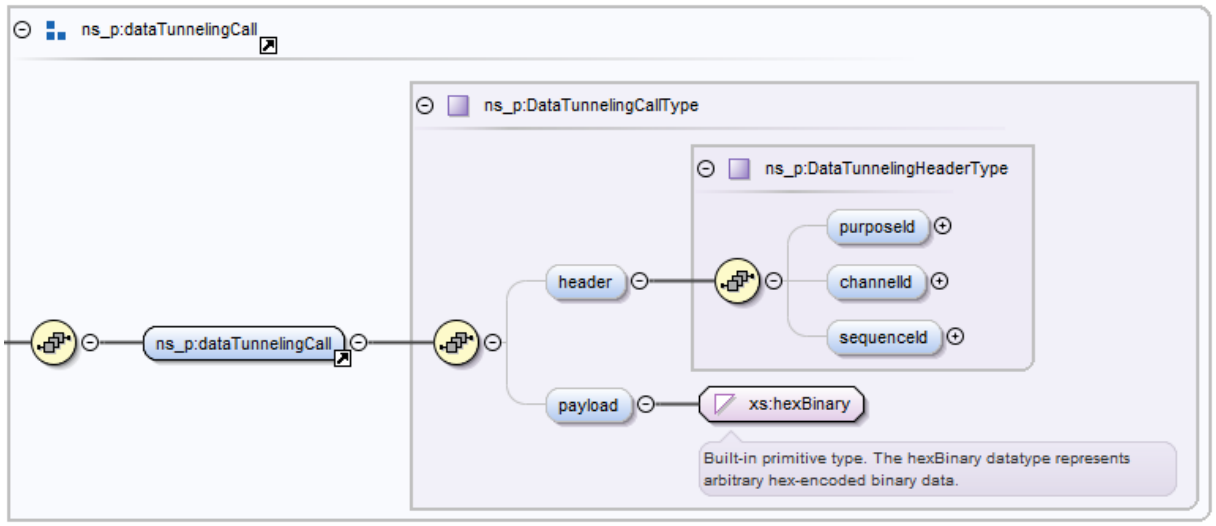
Some may find a comparison with Virtual Private Networks helpful (though VPNs have a different scope): Two computers "A", "B" each have to run a compatible (proprietary) program (i.e. mapping/implementation) that know how to communicate with each other (i.e. set up the communication and provide a specific authentication and encryption). There are manufacturers of VPN gateways that provide client software for different operating systems. This way a proprietary implementation is made available for others.

The combination { purposeId, channelId, sequenceId } should be considered unique for a communication from "A" to "B". Within a system (e.g. house) several communications may be used in parallel. The combinations used in these communications are in general independent of each other.

Please note that an instance of a dataTunnelingCall implementation permits bidirectional communication in general. I.e. if a dataTunnelingCall functionality of “B” submits a message to a dataTunnelingCall functionality of “A”, it is also permissive that the dataTunnelingCall functionality of “A” submits dataTunnelingCall messages to the dataTunnelingCall functionality of “B”. However, similar to a VPN, it may well be that the establishment for such message exchanges is “unidirectional” to some extent. I.e. it may be that the dataTunnelingCall functionality of “A” determines which kind of messages (or permissive values for “purposeId”) are accepted on this functionality and that the functionality first of all acts as some kind of “listening port”. This means the functionality of “A” may wait for an initial incoming message before it also sends dataTunnelingCall messages back.

Please note that the example above does not prevent device “B” from implementing another dataTunnelingCall functionality as “listening port” with own permitted values for “purposeId” as well. In that case device “A” could send a proper initial message to this “listening functionality” of “B”.

6311 SPINE data tunnelling is intended to be used for call-operations only. No read-reply or write
6312 procedures are supported.



6313

6314 *Figure 75: dataTunnelingCall function overview*

6315

6316 ### *5.3.6.2.2 Detailed description of elements*

Element	Type	Description
header	*Complex type "DataTunnelingHeaderType"*	A data tunneling specific header.
header. purposeId	*Simple type "PurposeIdType" (restriction of xs:string)*	A string that identifies the purpose / kind of tunnelled data / protocol.
header. channelId	*Simple type "ChannelIdType" (restriction of xs:unsignedInt)*	Between two nodes there might be several channels used in parallel using the same purposeId. channelId helps to separate these channels. It can also be used to differentiate subsequent (rather than parallel) sessions.
header. sequenceId	xs:unsignedInt (W3C standard type)	This helps to give the payload for each combination of { purposeId, channelId } a chronological order. Note: This “sequenceId” is data tunneling specific and should not be confused with “sequenceId” elements of other classes!
payload	xs:hexBinary (W3C standard type)	The actual proprietary data.

6317 *Table 180: dataTunnelingData function detailed description of elements*

6318

6319 ### *5.3.6.2.3 Available selectors*

6320 None.

6321

5.3.6.2.4 Examples

5.3.6.2.4.1 Call

Device "B" is configured to submit certain proprietary data to device "A". Due to a specific occasion it sends a call classifier together with the following XML:

```
<dataTunnelingCall>
    <header>
        <purposeId>FooManufacturer:BarProtocolWithVersion</purposeId>
        <channelId>7</channelId>
        <sequenceId>6425</sequenceId>
    </header>
<payload>3c3f786d6c207665727369 6f6e3d22312e302220656e636f64696e673d22555446
2d38223f3e0d0a3c6d6573736167654461746120786d6c6e733d22687474703a2f2f646f637
32e65656275732e6f72672f62657461312f312e302f787364223e0d0a202020203c74696d65
7374616d703e323030362d30352d30345431383a31333a35312e305a3c2f74696d657374616
d703e0d0a202020203c6d6573736167654e756d6265723e37323c2f6d6573736167654e756d
6265723e0d0a202020203c747970653e696e666f726d6174696f6e3c2f747970653e0d0a202
020203c746578743e55706461746520636f6d706c657465642e3c2f746578743e0d0a3c2f6d
6573736167654461746174613e0d0a</payload>
</dataTunnelingCall>
```

5.3.7 DeviceClassification

5.3.7.1 Introduction

Each device can have associated information useful for its identification and which kind of power it requires. This information (mostly plaintext) is bundled in the *DeviceClassification* class. It is intended to describe the physical device, but may be used for logical devices (= *entities*), too.

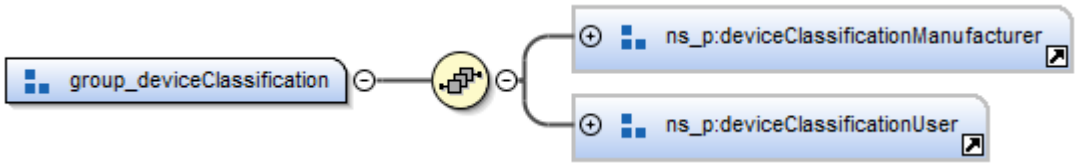


Figure 76: DeviceClassification function-group overview

5.3.7.2 deviceClassificationManufacturerData

5.3.7.2.1 General

Information which is usually NOT writable (hard coded in the device or preconfigured information, e.g.), is modelled with *deviceClassificationManufacturerData*.

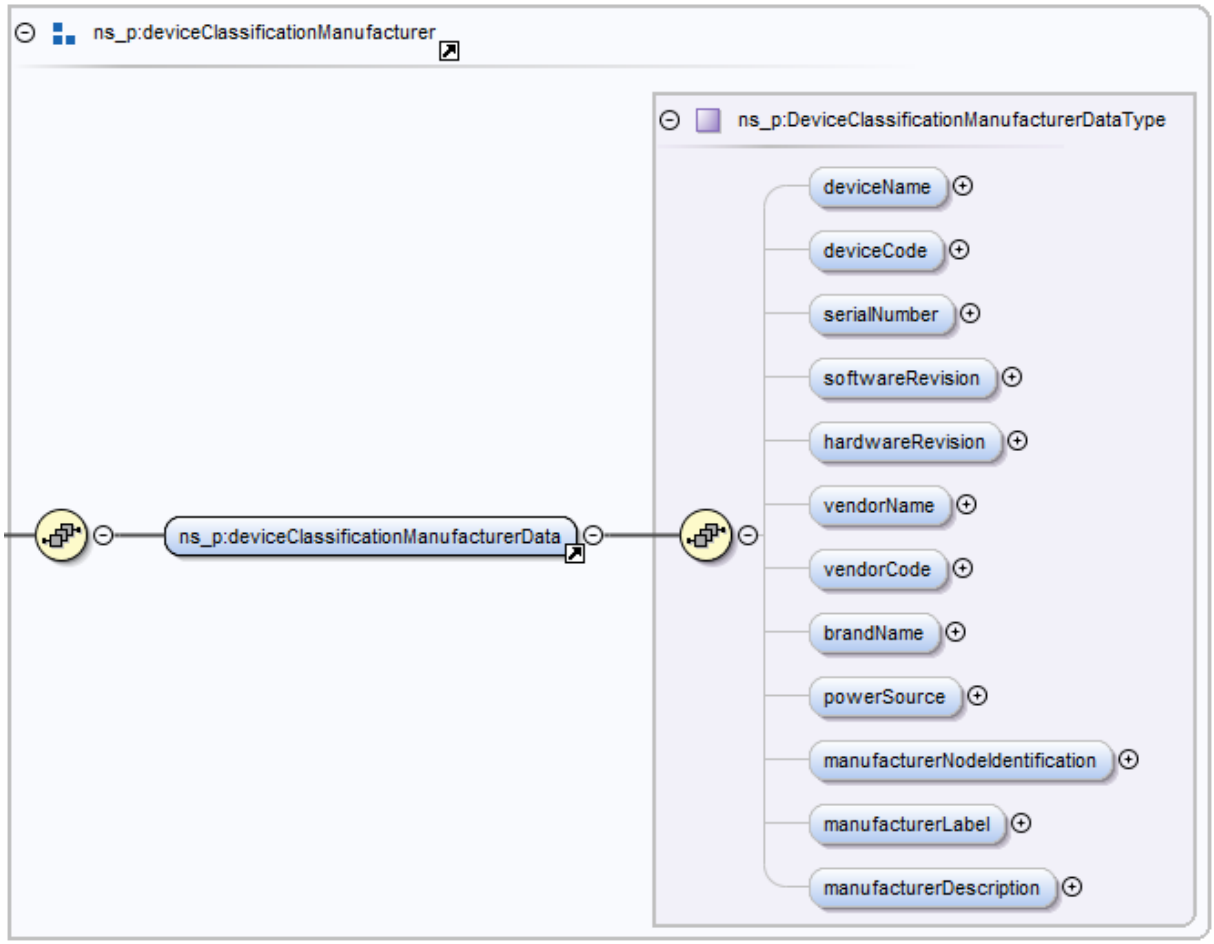


Figure 77: deviceClassificationManufacturerData function overview

5.3.7.2.2 Detailed description of elements

Please note that most of the subsequent elements permit any content. I.e. the underlying SPINE data model usually does not impose any format for the content of these elements. An exception applies if an enumeration simple-type is used (see *powerSource* element).

Element	Type	Description
deviceName	*Simple type "DeviceClassificationStringType" (restriction of xs:string)*	The name of the (physical or logical) device as defined by the manufacturer.
deviceCode	*Simple type "DeviceClassificationStringType" (restriction of xs:string)*	A device code for the (physical or logical) device as defined by the manufacturer.
serialNumber	*Simple type "DeviceClassificationStringType" (restriction of xs:string)*	The serial number of the (physical or logical) device as defined by the manufacturer. Usually the same as printed on the case.
softwareRevision	*Simple type "DeviceClassificationStringType" (restriction of xs:string)*	The software revision of the (physical or logical) device as

		defined by the manufacturer.
hardwareRevision	*Simple type "DeviceClassificationStringType" (restriction of xs:string)*	The hardware revision of the (physical or logical) device as defined by the manufacturer.
vendorName	*Simple type "DeviceClassificationStringType" (restriction of xs:string)*	The name of the vendor of the (physical or logical) device as defined by the manufacturer.
vendorCode	*Simple type "DeviceClassificationStringType" (restriction of xs:string)*	A code for the vendor of the (physical or logical) device as defined by the manufacturer.
brandName	*Simple type "DeviceClassificationStringType" (restriction of xs:string)*	The name of the brand. Useful where the name of the brand and the vendor differs.
powerSource	*Union "*PowerSourceType*":* *- Enum (see* Table 182*):* PowerSourceEnumType - EnumExtendType (see section 3.10.1.5)	This element describes for which kind of primary power source the device is designed for. It is only used for informative purposes. It is NOT used to describe how or to which kind of power source the device is finally connected (this means it is not considered here whether the device's single mains is fed by a house's mains supply or a backup sytem, e.g.).
manufacturerNodeIdentification	*Simple type "DeviceClassificationStringType" (restriction of xs:string)*	A node identification for the (physical or logical) device as defined by the manufacturer. This could be used for the identification of a (physical or logical) device, even if it was removed from the network and rejoined later with changed node address.
manufacturerLabel	*Common data type "LabelType". See section 3.10.1.2.*	A short label of the (physical or logical) device as defined by the manufacturer.
manufacturerDescription	*Common data type "DescriptionType". See section 3.10.1.3.*	A description for the (physical or logical) device as defined by the manufacturer.

6361 *Table 181: deviceClassificationManufacturerData function detailed description of elements*

6362 Enumeration **PowerSourceEnumType**:

Value	Description
unknown	Power source not known.
mainsSinglePhase	AC power connection, 1 phase.
mains3Phase	AC power connection, 3 phases.
battery	DC powered with a battery.
dc	DC powered with a constant energy source.

Table 182: Enumeration PowerSourceEnumType

5.3.7.2.3 Available selectors

None.

5.3.7.2.4 Examples

5.3.7.2.4.1 Read-reply

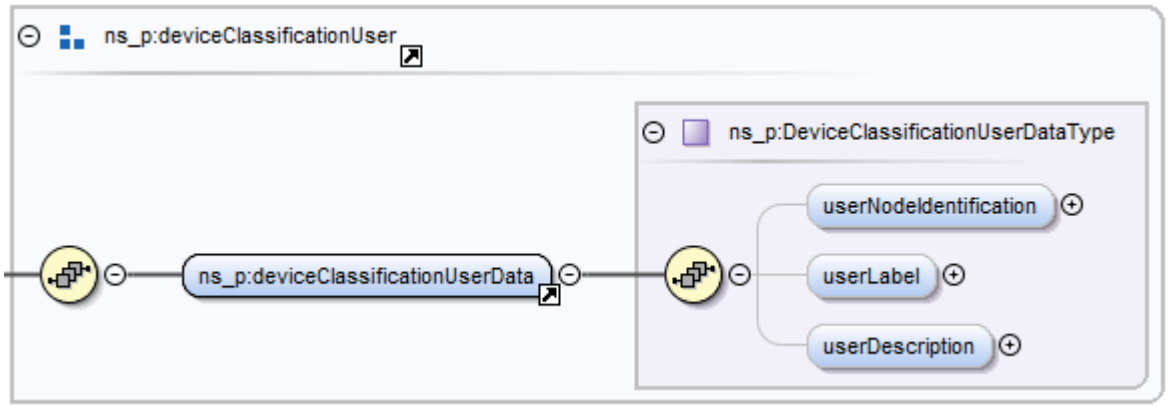
An example of a *deviceClassificationManufacturerData* instance, send as reply:

```
<deviceClassificationManufacturerData>
    <deviceName>SPINETestDevice_A</deviceName>
    <deviceCode>SPINE_TD_A</deviceCode>
    <serialNumber>159753852456</serialNumber>
    <softwareRevision>1.2.0</softwareRevision>
    <hardwareRevision>1.0.1</hardwareRevision>
    <vendorName>EEBus Initiative e.V.</vendorName>
    <vendorCode>4545427573</vendorCode>
    <brandName>TestBrand</brandName>
    <powerSource>mainsSinglePhase</powerSource>
<manufacturerNodeIdentification>SPINE_TD_A_1.2.0_159753852456_A</manufactur
erNodeIdentification>
    <manufacturerLabel>SPINE_TD_A@EEBus_e.V.</manufacturerLabel>
    <manufacturerDescription>SPINETestDevice_A Installation @ EEBus
Initiative e.V.</manufacturerDescription>
</deviceClassificationManufacturerData>
```

5.3.7.3 deviceClassificationUserData

5.3.7.3.1 General

Information which is usually writable (changeable by a user / application) is modelled with *deviceClassificationUserData*.



6392

6393 *Figure 78: deviceClassificationUserData function overview*

6394

6395 *5.3.7.3.2 Detailed description of elements*

Element	Type	Description
userNodeIdentification	*Simple type "DeviceClassificationStringType" (restriction of xs:string)*	Similar to *manufacturerNodeIdentification* but defined by a user or application.
userLabel	*Common data type "LabelType". See section 3.10.1.2.*	Similar to *manufacturerLabel* but defined by a user or application.
userDescription	*Common data type "DescriptionType". See section 3.10.1.3.*	Similar to *manufacturerDescription* but defined by a user or application.

6396 *Table 183: deviceClassificationUserData function detailed description of elements*

6397

6398 *5.3.7.3.3 Available selectors*

6399 None.

6400

6401 *5.3.7.3.4 Examples*

6402 *5.3.7.3.4.1 Read-reply*

6403 An example of a *deviceClassificationUserData* instance, send as reply:

```
6404 <deviceClassificationUserData>
6405     <userNodeIdentification>TD_A_159753852456</userNodeIdentification>
6406     <userLabel>TD_A of EEBus e.V.</userLabel>
6407     <userDescription>Test Device A; Install Date:
6408 15.07.13</userDescription>
6409 </deviceClassificationUserData>
```

6410

6411 **5.3.8 DeviceConfiguration**

6412 ***5.3.8.1 Introduction***

6413 Some information cannot be represented by the "normal" SPINE classes, but are important for the
6414 interoperable functionality of a device, hence should not be modelled with the DataTunneling class

(see section 5.3.6) or other proprietary ways. For this purpose, the DeviceConfiguration class should be used. It provides a key-value based modelling of information. Some data, important for interoperability, is already defined by the EEBus Initiative e.V. and listed in Table 187.

The information is split into a rather static part (see section 5.3.8.3) and the actual value (see section 5.3.8.2).

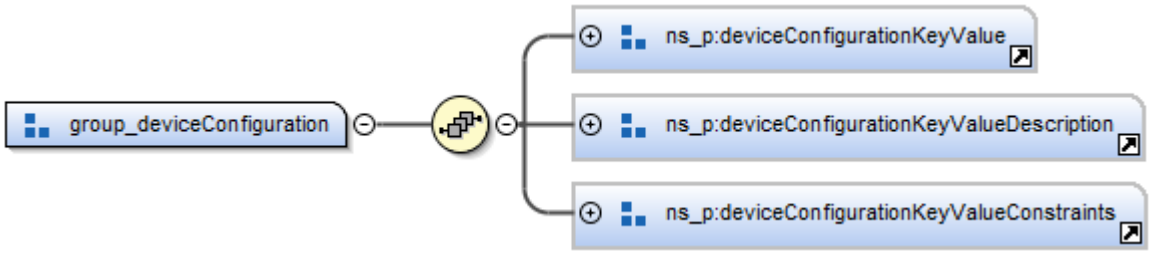


Figure 79: DeviceConfiguration function-group overview

5.3.8.2 deviceConfigurationKeyValueListData

5.3.8.2.1 General

The actual values belonging to a specific key are modelled within this function.

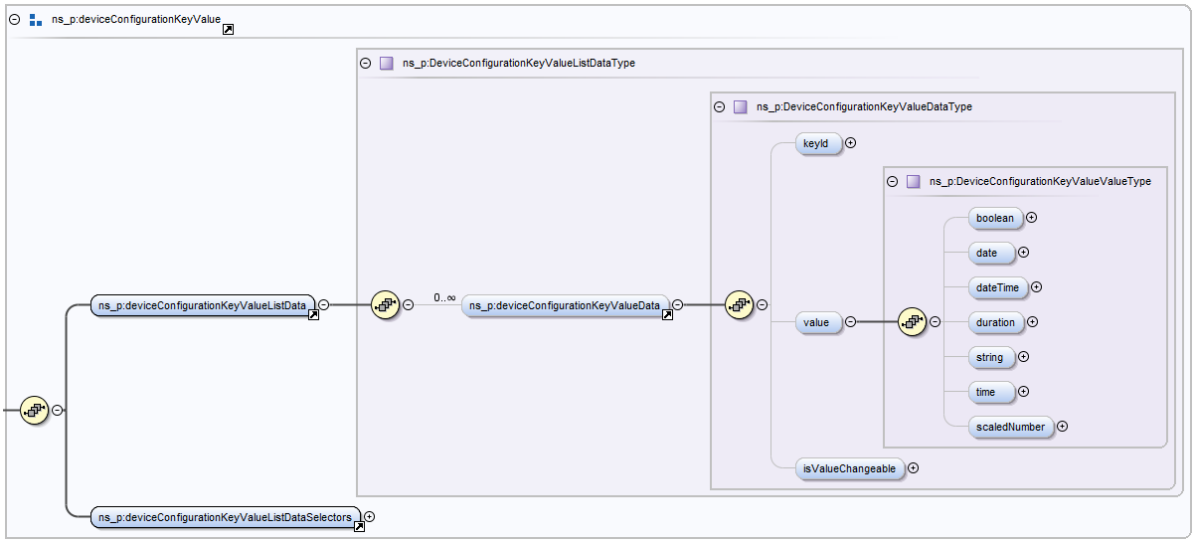


Figure 80: deviceConfigurationKeyValueListData function overview

5.3.8.2.2 Detailed description of elements

Element	Type	Description
keyId	*Identifier "DeviceConfigurationKeyIdType" (see* Table 339*).*	Enables the identification of different keys on one SPINE feature.
value	*Complex type "DeviceConfigurationKeyValueValueType". See* Table 185.	The actual value belonging to the *keyId*.

isValueChangeable	xs:boolean (W3C standard type)	States whether the value belonging to the *keyId* is changeable by a client or not.

Table 184: deviceConfigurationKeyValueListData function detailed description of elements

Complex type **DeviceConfigurationKeyValueValueType**:

Element	Type	Description
boolean	xs:boolean (W3C standard type)	Used, if the *value* is of type *boolean*.
date	xs:date (W3C standard type)	Used, if the *value* is of type *date*.
dateTime	xs:dateTime (W3C standard type)	Used, if the *value* is of type *dateTime*.
duration	xs:duration (W3C standard type)	Used, if the *value* is of type *duration*.
string	*Simple type "DeviceConfigurationKeyValueStringType" (restriction of xs:string)*	Used, if the *value* is of type *string*.
time	xs:time (W3C standard type)	Used, if the *value* is of type *time*.
scaledNumber	*Common data type "ScaledNumberType". See section 3.10.1.8.*	Used, if the *value* is of type *scaledNumber*.

Table 185: Complex type DeviceConfigurationKeyValueValueType

5.3.8.2.3 Available selectors

- keyId

5.3.8.2.4 Examples

5.3.8.2.4.1 Read-reply

If one is interested in the value of a specific key, he could send the following read command:

```
<function>deviceConfigurationKeyValueListData</function>
<filter>
    <cmdControl>
        <partial/>
    </cmdControl>
    <deviceConfigurationKeyValueListDataSelectors>
        <keyId>2</keyId>
    </deviceConfigurationKeyValueListDataSelectors>
</filter>
<deviceConfigurationKeyValueListData/>
```

If the receiving device has a proper value for this *keyId*, it would respond with the following function:

```
<deviceConfigurationKeyValueListData>
    <deviceConfigurationKeyValueData>
        <keyId>2</keyId>
        <value>
            <scaledNumber>
                <number>18</number>
                <scale>1</scale>
            </scaledNumber>
        </value>
```



```
            <isValueChangeable>false</isValueChangeable>
        </deviceConfigurationKeyValueData>
</deviceConfigurationKeyValueListData>
```

5.3.8.2.4.2 Notify

If a value changes, it will be notified to all subscribed clients:

```
<deviceConfigurationKeyValueListData>
    <deviceConfigurationKeyValueData>
        <keyId>2</keyId>
        <value>
            <scaledNumber>
                <number>32</number>
                <scale>2</scale>
            </scaledNumber>
        </value>
        <isValueChangeable>false</isValueChangeable>
    </deviceConfigurationKeyValueData>
</deviceConfigurationKeyValueListData>
```

5.3.8.3 deviceConfigurationKeyValueDescriptionListData

5.3.8.3.1 General

The rather static part of the device configuration keys is modelled with this function, including an identifier for the key, a name that defines the meaning of the key and further additional elements.

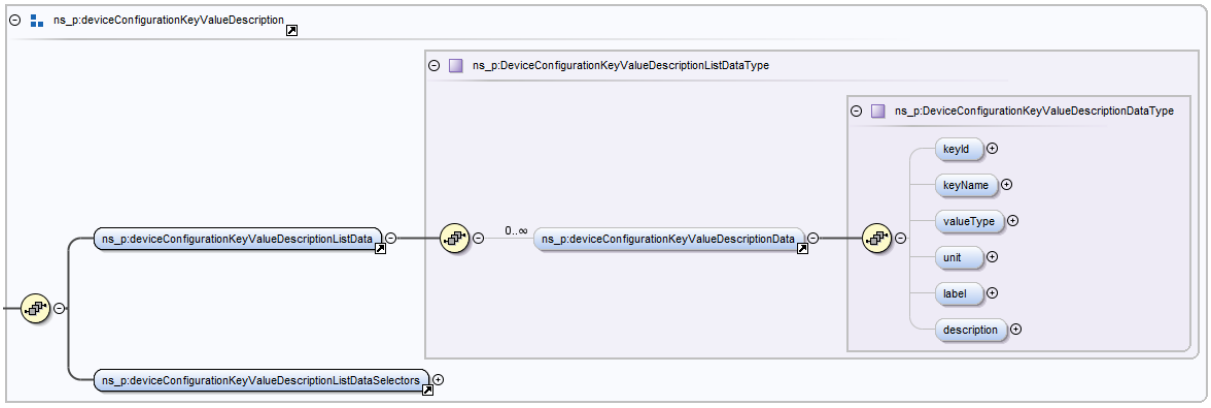


Figure 81: deviceConfigurationKeyValueDescriptionListData function overview

5.3.8.3.2 Detailed description of elements

Element	Type	Description
keyId	*Identifier "DeviceConfigurationKeyIdType" (see* Table 339*).*	The key identifier.
keyName	*Union* "DeviceConfigurationKeyNameType*":* *- Enum (see* Table 187*):* DeviceConfigurationKeyNameEnumType	A certain key name. If one of the keys defined by the EEBus Initiative e.V. is used, the meaning is defined in Table 187. Vendors may define own key names (but then have to mark them

	- EnumExtendType (see section 3.10.1.5)	as vendor specific, see [ProtocolSpecification], section "Rules for vendor specific extensions").
valueType	*Enumeration* DeviceConfigurationKeyValueTypeType *(see* Table 188*).*	Different value types are possible, but only one may be chosen. It is denoted in this element.
unit	*Common data type "UnitOfMeasurementType". See section 3.10.1.17.*	The unit in which the value of the key is given. Not all keys need the unit element.
label	*Common data type "LabelType". See section 3.10.1.2.*	A label for this key.
description	*Common data type "DescriptionType". See section 3.10.1.3.*	A description for this key.

Table 186: deviceConfigurationKeyValueDescriptionListData function detailed description of elements

Enumeration **DeviceConfigurationKeyNameEnumType**:

Value	**Description**
peakPowerOfPvSystem	The percentage of the peak power of the pv system that is permitted to be fed into the electricity grid through the electricity grid connection point.
pvCurtailmentLimitFactor	The nominal peak power of the production of the installed PV System.
asymmetricChargingSupported	This configuration parameter is used by the Entity EV to indicate the Entity SECC if it is possible to charge with different current per phase.
communicationsStandard	Defines the used communications standard for the entity where the DeviceConfiguration feature is located.

Table 187: Enumeration DeviceConfigurationKeyNameEnumType

Enumeration **DeviceConfigurationKeyValueTypeType**:

Value	**Description**
boolean	The corresponding key value is of type xs:boolean.
date	The corresponding key value is of type xs:date.
dateTime	The corresponding key value is of type xs:dateTime.
duration	The corresponding key value is of type xs:duration.
string	The corresponding key value is of type xs:string.
time	The corresponding key value is of type xs:time.
scaledNumber	The corresponding key value is of type *"ScaledNumberType" (see section 3.10.1.8*).

Table 188: Enumeration DeviceConfigurationKeyValueTypeType

5.3.8.3.3 Available selectors

- keyId
- keyName

5.3.8.3.4 Examples

5.3.8.3.4.1 Read-reply

If one is interested in the rather static information of the device configuration keys, he could request the information with a read command. The device replies with information like this:

```
<deviceConfigurationKeyValueDescriptionListData>
    <deviceConfigurationKeyValueDescriptionData>
        <keyId>1</keyId>
        <keyName>pvCurtailmentLimitFactor</keyName>
        <valueType>scaledNumber</valueType>
        <unit>pct</unit>
        <label>PV Curtailment Limit Factor</label>
        <description>The nominal peak power of the production of
the  installed pv system</description>
    </deviceConfigurationKeyValueDescriptionData>
    <deviceConfigurationKeyValueDescriptionData>
        <keyId>2</keyId>
        <keyName>peakPowerOfPvSystem</keyName>
        <valueType>scaledNumber</valueType>
        <unit>W</unit>
        <label>Peak Power of PV System</label>
        <description>The percentage of the peak power of the pv system that
is permitted to be fed into the electricity grid through the electricity
grid connection point.</description>
    </deviceConfigurationKeyValueDescriptionData>
</deviceConfigurationKeyValueDescriptionListData>
```

5.3.8.4 deviceConfigurationKeyValueConstraintsListData

5.3.8.4.1 General

In cases where a configuration value has some range, it could make sense to specify the upper and lower bounds that are possible for this value as well as a step size. Therefore, the deviceConfigurationKeyValueConstraintsListData function can be used.

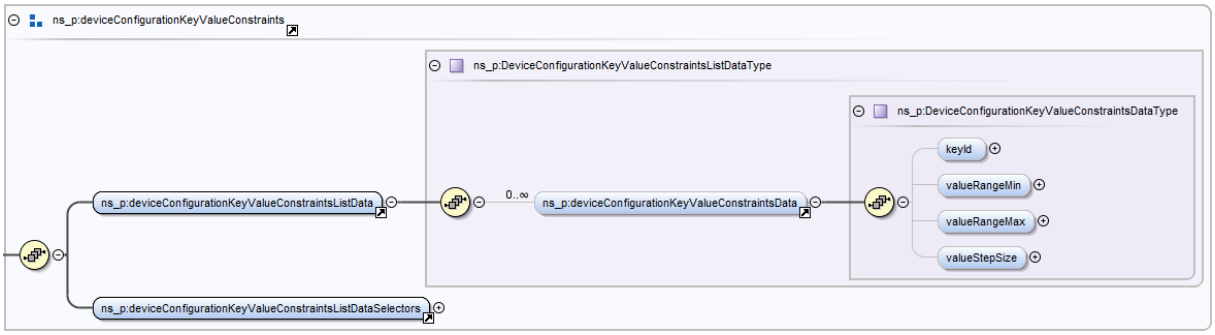


Figure 82: deviceConfigurationKeyValueConstraintsListData function overview

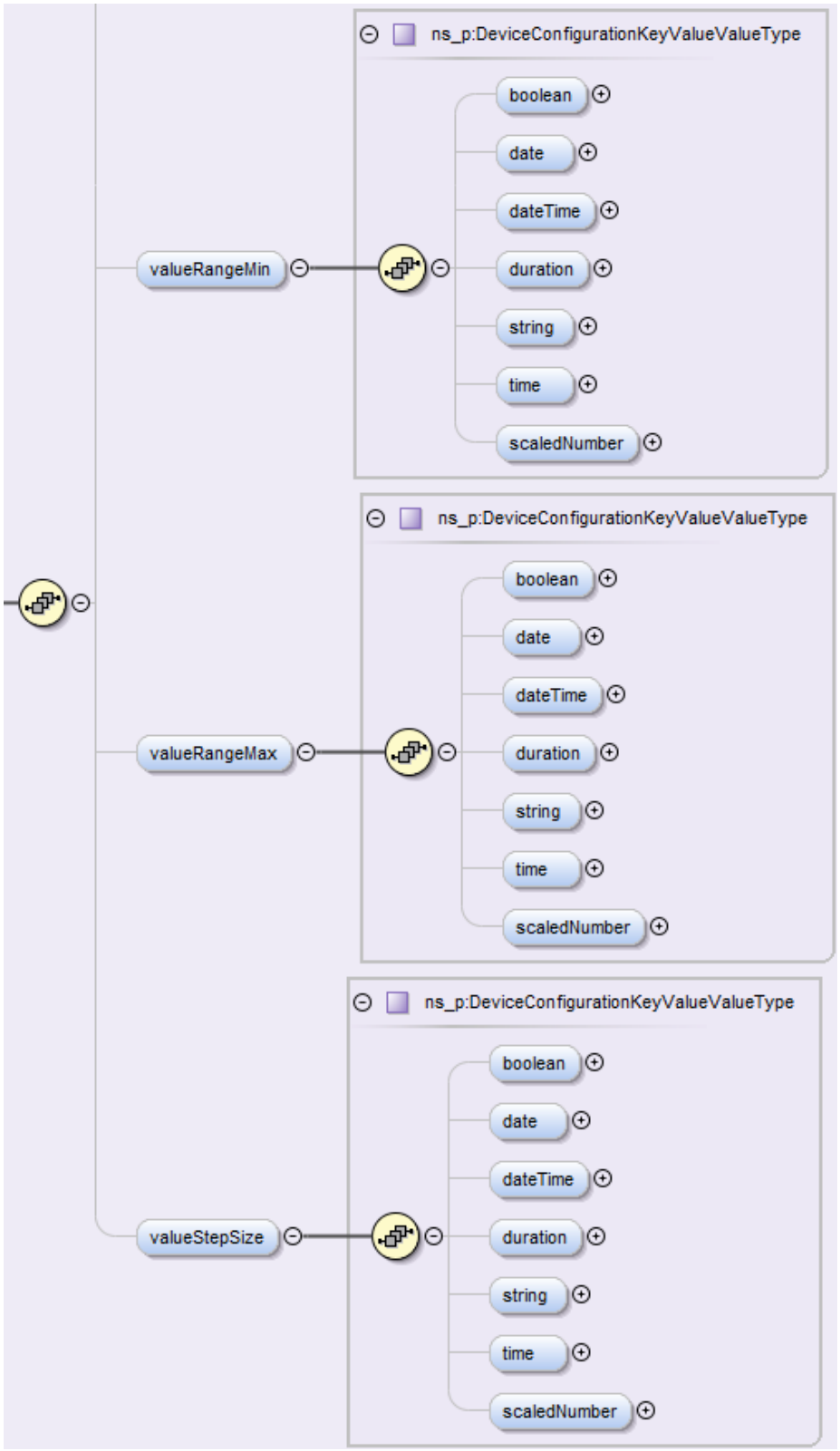


Figure 83: deviceConfigurationKeyValueConstraintsListData function detail

5.3.8.4.2 Detailed description of elements

Element	Type	Description

keyId	*Identifier "DeviceConfigurationKeyIdType" (see* Table 339*).*	Enables the identification of different keys on one SPINE feature.
valueRangeMin	*Complex type "DeviceConfigurationKeyValueValueType". See* Table 185*.*	The lower bound that is possible for the related value.
valueRangeMax	*Complex type "DeviceConfigurationKeyValueValueType". See* Table 185*.*	The upper bound that is possible for the related value.
valueStepSize	*Complex type "DeviceConfigurationKeyValueValueType". See* Table 185*.*	The step size that is possible for the related value.

Table 189: deviceConfigurationKeyValueListData function detailed description of elements

5.3.8.4.3 Available selectors

- keyId

5.3.8.4.4 Examples

5.3.8.4.4.1 Read-reply

If one is interested in the constraints of all device configuration keys, he could request the information with a read command. The device replies with information like this:

```
<deviceConfigurationKeyValueConstraintsListData>
    <deviceConfigurationKeyValueConstraintsData>
        <keyId>1</keyId>
        <valueRangeMin>
            <scaledNumber>
                <number>0</number>
            </scaledNumber>
        </valueRangeMin>
        <valueRangeMax>
            <scaledNumber>
                <number>1</number>
                <scale>2</scale>
            </scaledNumber>
        </valueRangeMax>
        <valueStepSize>
            <scaledNumber>
                <number>5</number>
                <scale>-1</scale>
            </scaledNumber>
        </valueStepSize>
    </deviceConfigurationKeyValueConstraintsData>
    <deviceConfigurationKeyValueConstraintsData>
        <keyId>2</keyId>
        <valueRangeMin>
            <scaledNumber>
                <number>5</number>
                <scale>3</scale>
            </scaledNumber>
        </valueRangeMin>
        <valueRangeMax>
            <scaledNumber>
```



```
                    <number>7</number>
                    <scale>3</scale>
                </scaledNumber>
            </valueRangeMax>
            <valueStepSize>
                <scaledNumber>
                    <number>1</number>
                </scaledNumber>
            </valueStepSize>
        </deviceConfigurationKeyValueConstraintsData>
</deviceConfigurationKeyValueConstraintsListData>
```

5.3.9 DeviceDiagnosis

5.3.9.1 Introduction

This class contains information about the state of a device, heartbeat counting and service data.

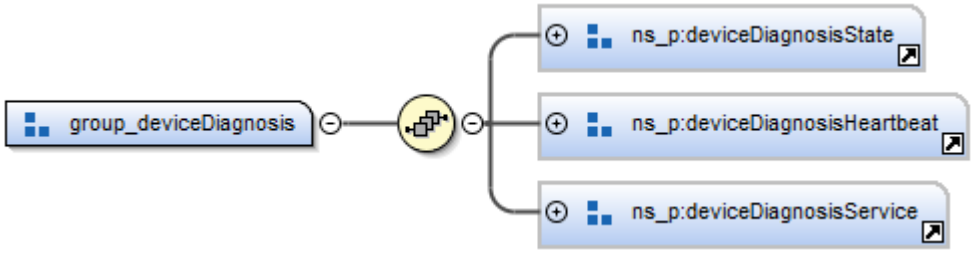
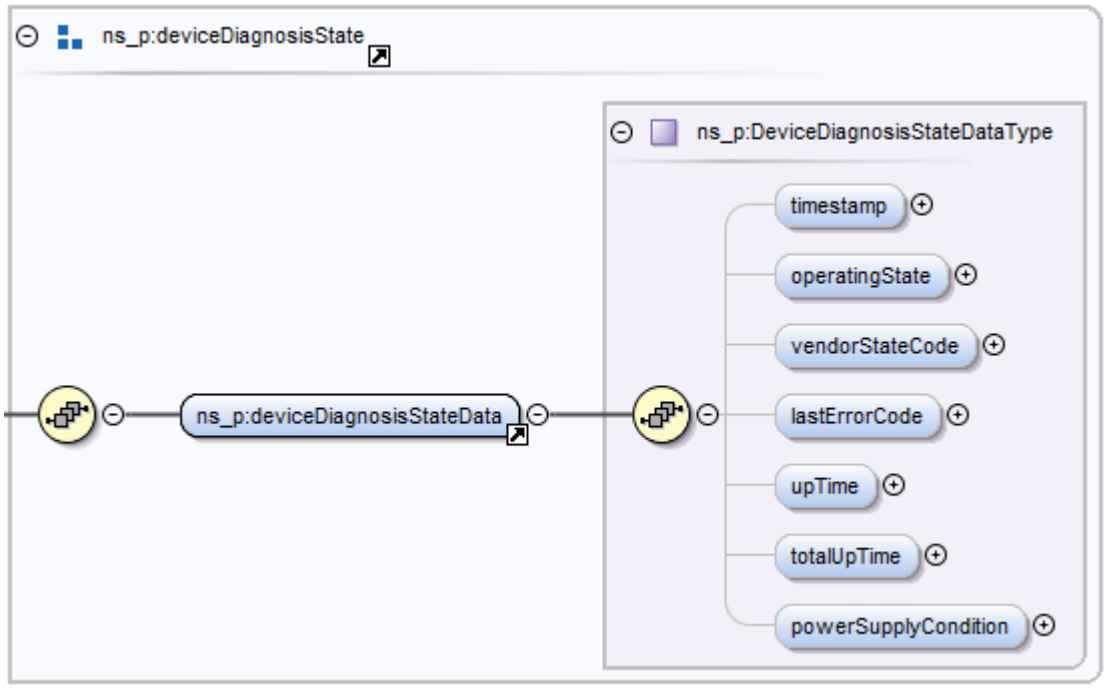


Figure 84: DeviceDiagnosis function-group overview

5.3.9.2 deviceDiagnosisStateData

5.3.9.2.1 General

The *deviceDiagnosisStateData* function provides information about dynamically changing state information.



6596

6597 *Figure 85: deviceDiagnosisStateData function overview*

6598

6599 *5.3.9.2.2 Detailed description of elements*

Element	Type	Description
timestamp	*Common data type "AbsoluteOrRelativeTimeType". See section 3.10.2.3.*	The time of creation of the state.
operatingState	*Union* "DeviceDiagnosisOperatingStateType*":* *- Enum (see* Table 191*):* DeviceDiagnosisOperatingStateEnumType - EnumExtendType (see section 3.10.1.5)	The current operating state of the device.
vendorStateCode	*Simple type "VendorStateCodeType" (restriction of xs:string)*	A vendor specific state. May contain any state, the vendor uses for its devices.
lastErrorCode	*Simple type "LastErrorCodeType" (restriction of xs:string)*	The last error code that occurred is stored in this element. Even if the device is in *normalOperation* again, the error code remains here.
upTime	xs:duration (W3C standard type)	The duration the functionality is running since the last start.
totalUpTime	xs:duration (W3C standard type)	The total duration the functionality is running since installation (see *installationTime* element in *deviceDiagnosisServiceData* function).

powerSupplyCondition	*Union* "PowerSupplyConditionType"*:* - *Enum (see* Table 192*):* PowerSupplyConditionEnumType - EnumExtendType (see section 3.10.1.5)	The condition of the power supply can be stated here.

6600 *Table 190: deviceDiagnosisStateData function detailed description of elements*

6601 Enumeration **DeviceDiagnosisOperatingStateEnumType**:

Value	Description
normalOperation	The device does its normal operation without any errors or limitations.
standby	The device is in standby mode and does nothing but waits for user interaction and listens on its communications protocol for external commands.
failure	A failure occurred and the device cannot run as desired. Its normal functionality is likely not given.
serviceNeeded	The device needs some kind of service. Possibly the normal functionality is not given.
overrideDetected	The device detected an override. Normal operation may be given or the device is in some safe mode.
inAlarm	An alarm (or emergency) occurred and user interaction is needed. Normal operation may be given or the device is in some safe mode.
notReachable	The device is not reachable via its non-SPINE communications protocol. This operating state can only be set by some technology gateway device that handles the normal communication to the device.
finished	The device has finished its temporary operation. As long as it remains in state *finished*, the normal operation is not given.

6602 *Table 191: Enumeration DeviceDiagnosisOperatingStateEnumType*

6603 Enumeration **PowerSupplyConditionEnumType**:

Value	Description
good	The power supply is in a good condition.
low	The power supply has low capacity. It would make sense to check it (e.g. change battery).
critical	The power supply is in critical state. If it is supplied by a battery, a change is advised.
unknown	The power supply condition is unknown. A notification to a user could make sense.
error	The power supply is in error condition.

6604 *Table 192: Enumeration PowerSupplyConditionEnumType*

6605

6606 *5.3.9.2.3 Available selectors*

6607 None.

6608

6609 *5.3.9.2.4 Examples*

6610 *5.3.9.2.4.1 Notify*

6611 The current state of a device is notified to all subscribed clients:

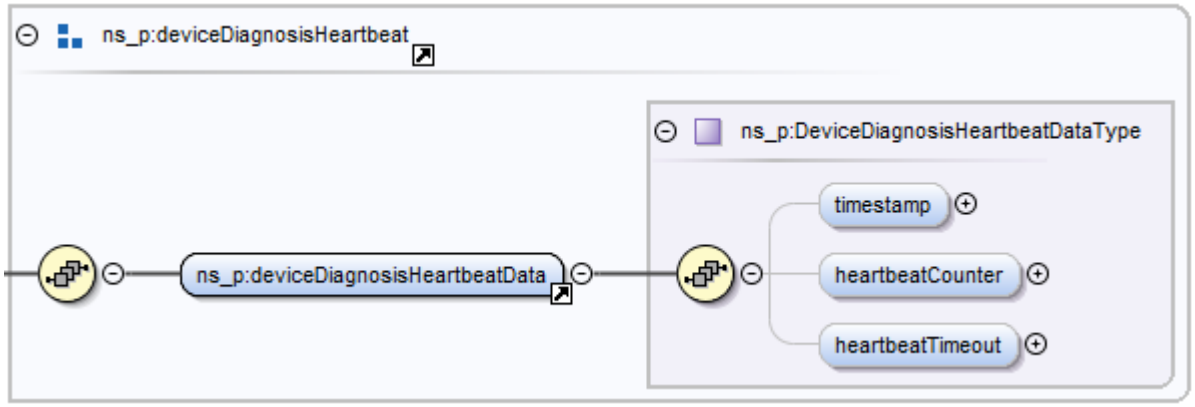
```
6612 <deviceDiagnosisStateData>
6613     <timestamp>2014-04-28T17:43:14.0Z</timestamp>
6614     <operatingState>normalOperation</operatingState>
6615     <vendorStateCode>heating up</vendorStateCode>
6616     <lastErrorCode>13</lastErrorCode>
6617     <upTime>P3DT4H7M14S</upTime>
6618     <totalUpTime>P2Y8M15DT7H32M17S</totalUpTime>
6619     <powerSupplyCondition>good</powerSupplyCondition>
6620 </deviceDiagnosisStateData>
```

6621

6622 *5.3.9.3 deviceDiagnosisHeartbeatData*

6623 *5.3.9.3.1 General*

6624 The *deviceDiagnosisHeartbeatData* function enables sending a specific signal, which indicates a
6625 device is still running.



6626

6627 *Figure 86: deviceDiagnosisHeartbeatData function overview*

6628

6629 *5.3.9.3.2 Detailed description of elements*

Element	Type	Description
timestamp	*Common data type "AbsoluteOrRelativeTimeType". See section 3.10.2.3.*	The time of creation of the data.
heartbeatCounter	xs:unsignedLong (W3C standard type)	An incrementing counter of the heartbeat.
heartbeatTimeout	xs:duration (W3C standard type)	The heartbeatTimeout element contains the period, in which the deviceDiagnosisHeartbeatData function is sent by the device (NOT the remaining time till the next sending).

6630 *Table 193: deviceDiagnosisHeartbeatData function detailed description of elements*

6631

5.3.9.3.3 Available selectors

None.

5.3.9.3.4 Examples

5.3.9.3.4.1 Read-reply

One is interested in the heartbeat information of a device and sends a standard read command to the device which responds with a *deviceDiagnosisHeartbeatData* function:

```
<deviceDiagnosisHeartbeatData>
    <timestamp>2014-04-28T17:54:38.0Z</timestamp>
    <heartbeatCounter>238</heartbeatCounter>
    <heartbeatTimeout>PT15M</heartbeatTimeout>
</deviceDiagnosisHeartbeatData>
```

5.3.9.4 deviceDiagnosisServiceData

5.3.9.4.1 General

The *deviceDiagnosisServiceData* function informs about service information related to a device's installation or maintenance.

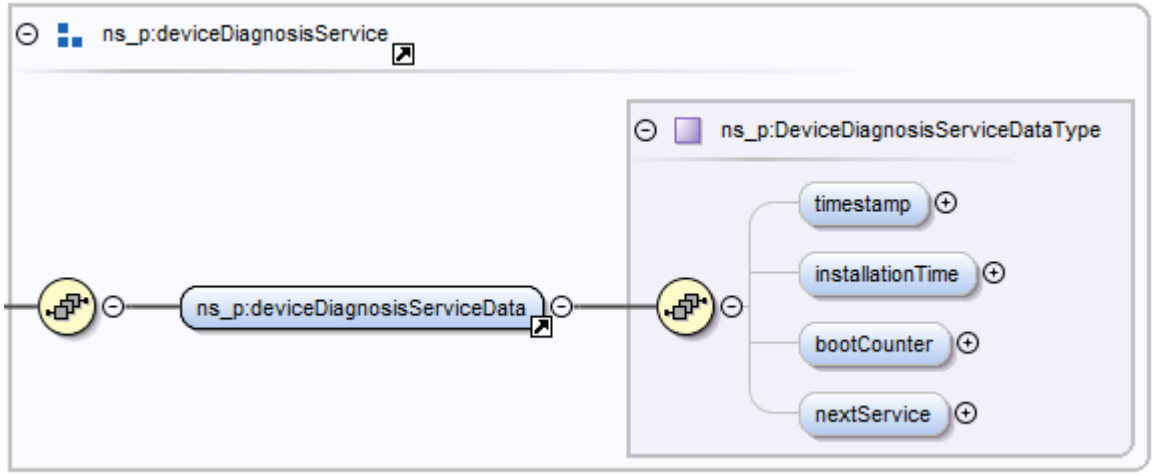


Figure 87: deviceDiagnosisServiceData function overview

5.3.9.4.2 Detailed description of elements

Element	Type	Description
timestamp	*Common data type "AbsoluteOrRelativeTimeType". See section 3.10.2.3.*	The time of creation of the data.
installationTime	*Common data type "AbsoluteOrRelativeTimeType". See section 3.10.2.3.*	The point in time the device was installed (commissioned).
bootCounter	xs:unsignedLong (W3C standard type)	The count of boot processes the device completed since *installationTime*.

nextService	*Common data type "AbsoluteOrRelativeTimeType". See section 3.10.2.3.*	The point in time, the next service should be done, is stated by this element. Alternatively, the decrementing duration till the next service can be indicated.

Table 194: deviceDiagnosisServiceData function detailed description of elements

5.3.9.4.3 Available selectors

None.

5.3.9.4.4 Examples

5.3.9.4.4.1 Read-reply

One is interested in the service information of a device and sends a standard read command to the device which responds with a *deviceDiagnosisServiceData* function:

```
<deviceDiagnosisServiceData>
    <timestamp>2014-04-28T17:59:05.0Z</timestamp>
    <installationTime>2011-08-15T11:05:08.0Z</installationTime>
    <bootCounter>83</bootCounter>
    <nextService>2015-08-15T11:05:08.0Z</nextService>
</deviceDiagnosisServiceData>
```

5.3.10 DirectControl

5.3.10.1 Introduction

Some kind of electricity consuming or producing devices require (in contrast to the PowerSequences Class, see section 5.3.19) a rather instantaneous energy consumption or generation control. This can be modelled with two primary information sets:

3. Notification of instantaneous consumption/generation as well as instantaneous control. This is covered by DirectControl and discussed in this section.
4. Information on the power/energy constraints and implications. This is covered by OperatingConstraints and discussed in section 5.3.18.

Information on and control of energy/power can be modelled using directControlActivity. directControlDescription is used to model some general constant data.

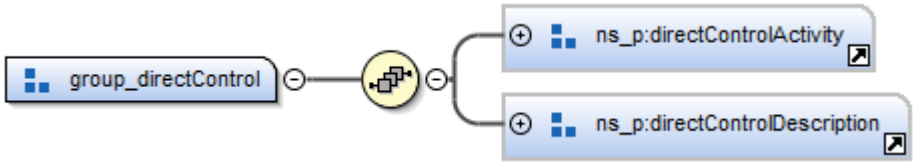


Figure 88: DirectControl function-group overview

5.3.10.2 directControlActivityListData

5.3.10.2.1 General

The function *directControlActivityListData* combines some information on the activity of a given time. Apart from information on state, power and energy it can contain information whether power, energy, etc. can be modified with a proper (partial) write command.

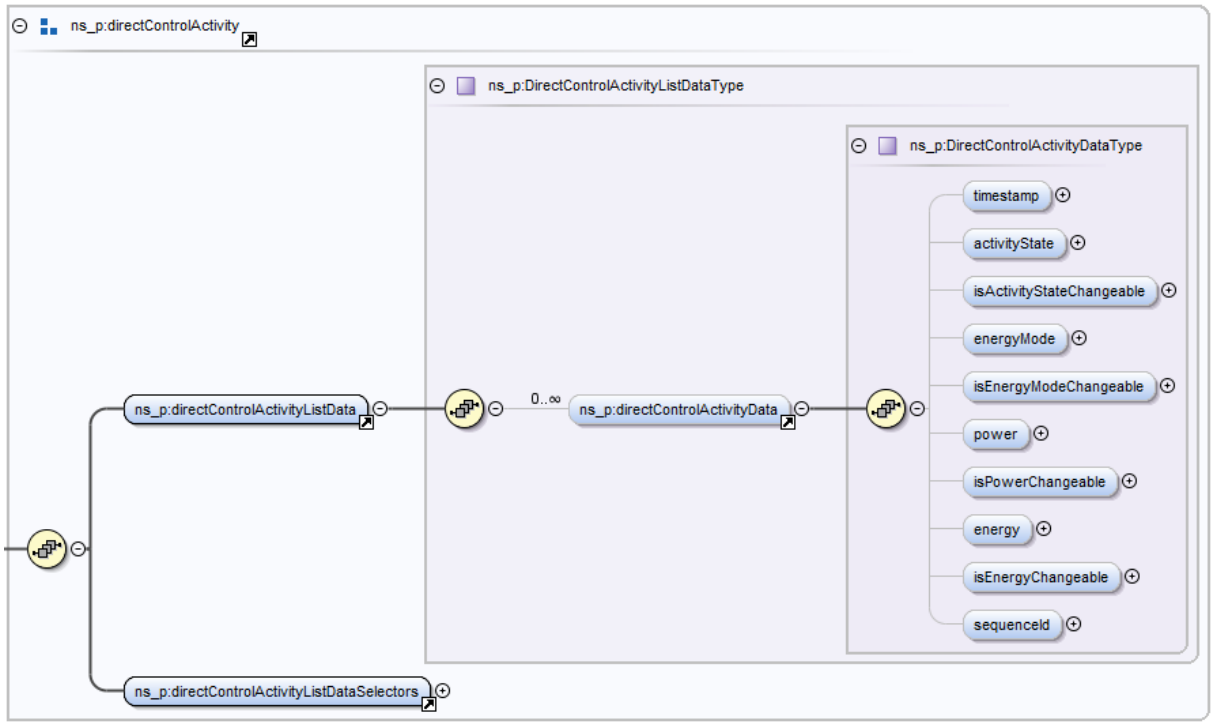


Figure 89: directControlActivityListData function overview

5.3.10.2.2 Detailed description of elements

Element	Type	Description
timestamp	*Common data type "AbsoluteOrRelativeTimeType". See section 3.10.2.3.*	The timestamp of this function instance.
activityState	*Union* "DirectControlActivityStateType*":* *- Enum (see* Table 196*):* DirectControlActivityStateEnumType - EnumExtendType (see section 3.10.1.5)	The state of the activity.
isActivityStateChangeable	xs:boolean (W3C standard type)	Indicates whether the "activityState" can be modified by a client or not.
energyMode	*Common data type "EnergyModeType". See section 3.10.1.15.*	Energy mode of the device.
isEnergyModeChangeable	xs:boolean (W3C standard type)	Indicates whether the "energyMode" can be modified by a client or not.

power	*Common data type "ScaledNumberType". See section 3.10.1.8.*	The overall electrical power value belonging to this activity information. Please note: Phase-specific information is not given here.
isPowerChangeable	xs:boolean (W3C standard type)	Indicates whether the "power" can be modified by a client or not.
energy	*Common data type "ScaledNumberType". See section 3.10.1.8.*	The electrical energy value belonging to this activity information.
isEnergyChangeable	xs:boolean (W3C standard type)	Indicates whether the "energy" can be modified by a client or not.
sequenceId	xs:unsignedInt (W3C standard type)	The related power sequence identifier (if available).

Table 195: directControlActivityListData function detailed description of elements

Enumeration **DirectControlActivityStateEnumType**:

Value	Description
running	The activity is running.
paused	The activity is paused.
inactive	The activity is inactive.

Table 196: Enumeration DirectControlActivityStateEnumType

5.3.10.2.3 Available selectors

- timestampInterval

5.3.10.2.4 Examples

5.3.10.2.4.1 Notify

When the activity state changes, a server will notify the new information to all subcribed devices with a message like this (the “filter/partial” part to announce the “restricted function exchange is simplified by “...” just to improve the readability”):

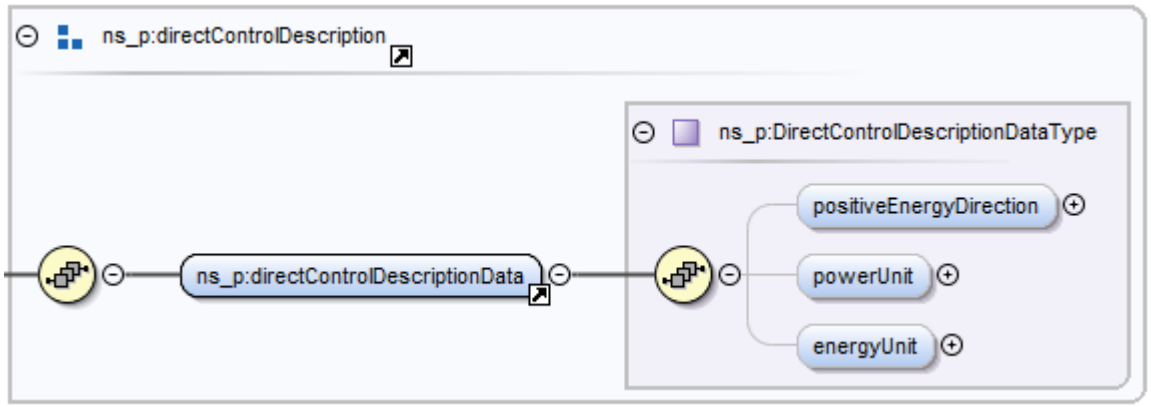
```
...
<directControlActivityListData>
    <directControlActivityData>
        <timestamp>2015-11-23T12:05:04.3Z</timestamp>
        <activityState>running</activityState>
        <isActivityStateChangeable>true</isActivityStateChangeable>
        <energyMode>consume</energyMode>
        <isEnergyModeChangeable>false</isEnergyModeChangeable>
        <power>
            <number>2</number>
            <scale>3</scale>
        </power>
        <isPowerChangeable>false</isPowerChangeable>
        <sequenceId>1</sequenceId>
    </directControlActivityData>
</directControlActivityListData>
```


6720

6721 ***5.3.10.3 directControlDescriptionData***

6722 *5.3.10.3.1 General*

6723 This function contains the rather static information available in this class.



6724

6725 *Figure 90: directControlDescriptionData function overview*

6726

6727 *5.3.10.3.2 Detailed description of elements*

Element	Type	Description
positiveEnergyDirection	*Common data type "EnergyDirectionType". See section 3.10.1.13.*	The *positiveEnergyDirection* states whether energy consumption or production will be counted as positive value.
powerUnit	*Common data type "UnitOfMeasurementType". See section 3.10.1.17.*	This is the unit for all power values related to consumption or production.
energyUnit	*Common data type "UnitOfMeasurementType". See section 3.10.1.17.*	This is the unit for all energy values related to consumption or production.

6728 *Table 197: directControlDescriptionData function detailed description of elements*

6729

6730 *5.3.10.3.3 Available selectors*

6731 None.

6732

6733 *5.3.10.3.4 Examples*

6734 *5.3.10.3.4.1 Read-reply*

6735 If one is interested in the rather static information of the direct control, he could request the
6736 information with a read command. The device replies with information like this:

```
6737 <directControlDescriptionData>
6738     <positiveEnergyDirection>consume</positiveEnergyDirection>
6739     <powerUnit>W</powerUnit>
6740     <energyUnit>Wh</energyUnit>
```



```
</directControlDescriptionData>
```

5.3.11 ElectricalConnection

5.3.11.1 Introduction

Electrical connections have some more complex parameters (e.g. which AC phase is measured, real/reactive/apparent measurement, amplitude/rms, number of measured harmonic, etc.), which are modelled with this class. Some electrical connection specific state information (e.g. AC or DC, number of connected phases (only AC), positive energy direction, label, etc.) is modelled, too.

Measuring electrical values is still done with the Measurement class (see section 5.3.15). But the additional parameters are only described within the ElectricalConnection class. The *measurementId* from the Measurement class is stated in the *electricalConnectionParameterDescriptionListData* function, to have a proper reference.

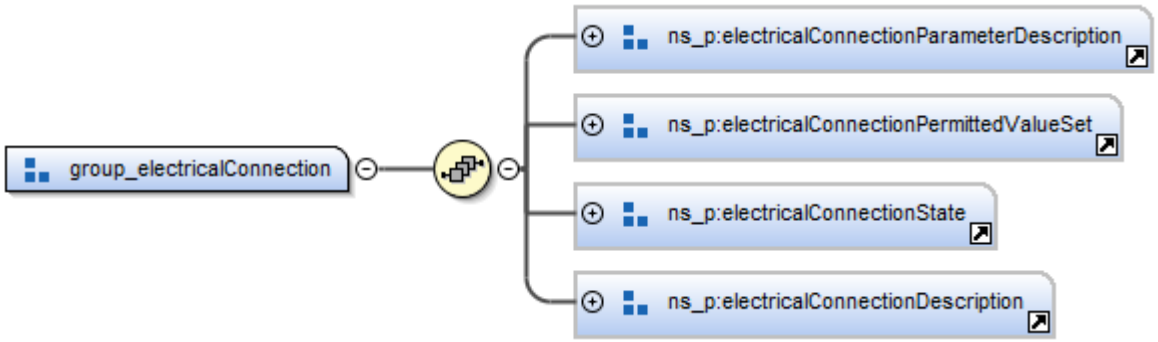


Figure 91: ElectricalConnection function-group overview

5.3.11.2 electricalConnectionParameterDescriptionListData

5.3.11.2.1 General

The aforementioned parameters for electrical connection measurement are described in this function. The value itself is modelled in the Measurement class (section 5.3.15). Only a reference is given in this function.

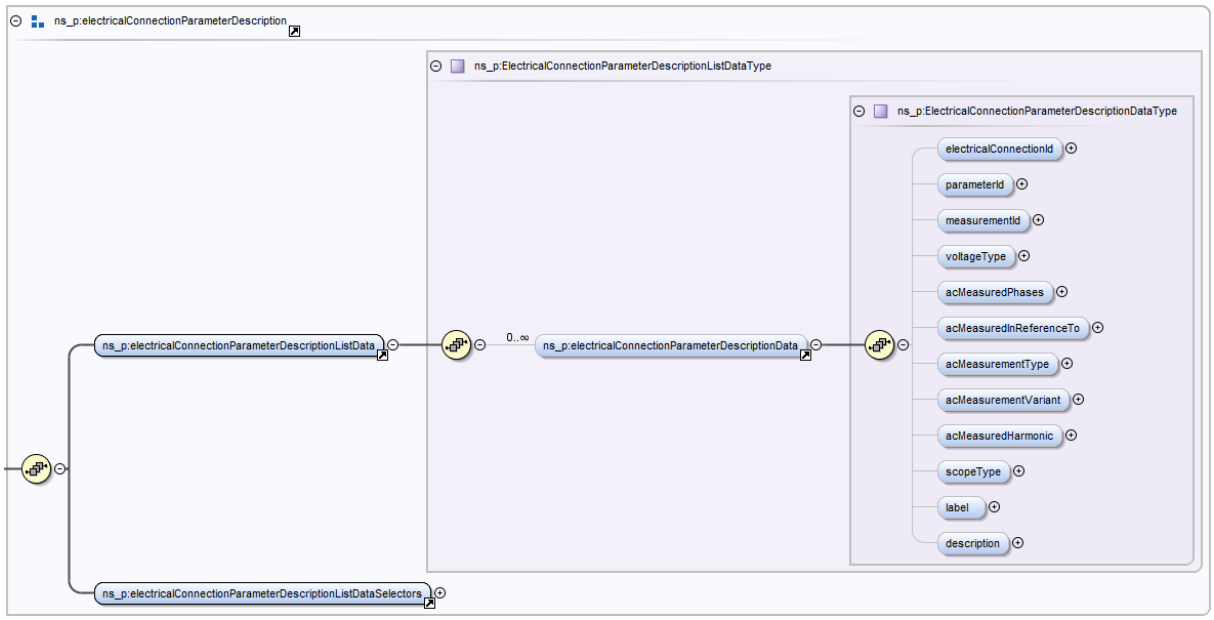


Figure 92: electricalConnectionParameterDescriptionListData function overview

5.3.11.2.2 Detailed description of elements

Element	Type	Description
electricalConnectionId	*Identifier "ElectricalConnectionIdType" (see* Table 339*).*	Reference to the electrical connection, described in the *electricalConnectionDescriptionListData* function (see section 5.3.11.5).
parameterId	*Identifier "ElectricalConnectionParameterIdType" (see* Table 339*).*	One electrical connection can have multiple parameter combinations, which can be measured. The *parameterId* helps to distinguish them.
measurementId	*Identifier "MeasurementIdType" (see* Table 339*).*	Reference to the Measurement class (see section 5.3.15), which contains the actual measured values for this parameter combination.
voltageType	*Union "*ElectricalConnectionVoltageTypeType*":* *- Enum (see* Table 199*):* ElectricalConnectionVoltageTypeEnumType - EnumExtendType (see section 3.10.1.5)	Specifies which kind of electricity is measured ("ac" or "dc").
acMeasuredPhases	*Union "*ElectricalConnectionPhaseNameType*":* *- Enum (see* Table 200*):* ElectricalConnectionPhaseNameEnumType - EnumExtendType (see section 3.10.1.5)	In case of *powerSupplyType* = "ac", this element states the phase, which is measured. Combinations of phases (e.g. "ab") are possible.
acMeasuredInReferenceTo	*Union "*ElectricalConnectionPhaseNameType*":*	In case of *powerSupplyType* = "ac", this element states the

	- *Enum (see* Table 200*):* ElectricalConnectionPhaseNameEnumType - EnumExtendType (see section 3.10.1.5)	phase, which *acMeasuredPhases* is measured against (e.g. "neutral").
acMeasurementType	*Union "*ElectricalConnectionAcMeasurementTypeType*":* - *Enum (see* Table 201*):* ElectricalConnectionAcMeasurementTypeEnumType - EnumExtendType (see section 3.10.1.5)	In case of *powerSupplyType* = "ac", this element states the kind of ac measurement is done (e.g. "real").
acMeasurementVariant	*Union "*ElectricalConnectionMeasurandVariantType*":* - *Enum (see* Table 202*):* ElectricalConnectionMeasurandVariantEnumType - EnumExtendType (see section 3.10.1.5)	In case of *powerSupplyType* = "ac", this element states the variation of the ac measurement (e.g. "amplitude").
acMeasuredHarmonic	xs:unsignedByte (W3C standard type)	In case of *powerSupplyType* = "ac", this element states the harmonic, which is measured. A value of "1" indicates that the normal value is measured.
scopeType	*Common data type "ScopeTypeType". See section 3.10.1.21.*	A certain meaning of the parameter combination.
label	*Common data type "LabelType". See section 3.10.1.2.*	A label for this electrical connection parameter.
description	*Common data type "DescriptionType". See section 3.10.1.3.*	A description for this electrical connection parameter.

Table 198: electricalConnectionParameterDescriptionListData function detailed description of elements

Enumeration **ElectricalConnectionVoltageTypeEnumType**:

Value	Description
ac	Alternating current.
dc	Direct current.

Table 199: Enumeration ElectricalConnectionVoltageTypeEnumType

Enumeration **ElectricalConnectionPhaseNameEnumType**:

Value	Description
a	Phase a (or L1).
b	Phase b (or L2).
c	Phase c (or L3).
ab	Phase a and b (or L1 and L2).
bc	Phase b and c (or L2 and L3).
ac	Phase a and c (or L1 and L3).
abc	Phase a, b and c (or L1, L2 and L3).
neutral	The neutral conductor of a connection.
ground	The ground conductor of a connection.
none	No phase specified or available.

Table 200: Enumeration ElectricalConnectionPhaseNameEnumType

Enumeration **ElectricalConnectionAcMeasurementTypeEnumType**:

Value	Description
real	AC measurement of the real part.
reactive	AC measurement of the reactive part.
apparent	AC measurement of the apparent part.
phase	AC measurement of the phase.

Table 201: Enumeration ElectricalConnectionAcMeasurementTypeEnumType

Enumeration **ElectricalConnectionMeasurandVariantEnumType**:

Value	Description
amplitude	Measurement of the amplitude.
rms	Measurement of the RMS (Root Mean Square).
instantaneous	Measurement of the instantaneous value.
angle	Measurement of the angle.
cosPhi	Measurement of the cosine phi.

Table 202: Enumeration ElectricalConnectionMeasurandVariantEnumType

5.3.11.2.3 Available selectors

- electricalConnectionId
- parameterId
- measurementId
- scopeType

5.3.11.2.4 Examples

5.3.11.2.4.1 Read-reply

If one only knows the *measurementId* of a specific measurand and is interested in the electrical connection parameters, he could send the following read command:

```
<function>electricalConnectionParameterDescriptionListData</function>
<filter>
    <cmdControl>
        <partial/>
    </cmdControl>
    <electricalConnectionParameterDescriptionListDataSelectors>
        <measurementId>6</measurementId>
    </electricalConnectionParameterDescriptionListDataSelectors>
</filter>
<electricalConnectionParameterDescriptionListData/>
```

If the receiving device has a proper parameter description for this *measurementId*, it would respond with the following function:

```
<function>electricalConnectionParameterDescriptionListData</function>
<filter>
    <cmdControl>
        <partial/>
    </cmdControl>
</filter>
<electricalConnectionParameterDescriptionListData>
```



```
    <electricalConnectionParameterDescriptionData>
        <electricalConnectionId>1</electricalConnectionId>
        <parameterId>3</parameterId>
        <measurementId>6</measurementId>
        <voltageType>ac</voltageType>
        <acMeasuredPhases>a</acMeasuredPhases>
        <acMeasuredInReferenceTo>neutral</acMeasuredInReferenceTo>
        <acMeasurementType>real</acMeasurementType>
        <acMeasurementVariant>rms</acMeasurementVariant>
        <acMeasuredHarmonic>1</acMeasuredHarmonic>
        <scopeType>acYieldTotal</scopeType>
    </electricalConnectionParameterDescriptionData>
</electricalConnectionParameterDescriptionListData>
```

5.3.11.3 electricalConnectionPermittedValueSetListData

5.3.11.3.1 General

If an electrical connection parameter has some defined value(s) or value range(s) that are allowed or possible, they can be modelled with this function. It enables the server to state single values, value ranges or combinations of both.

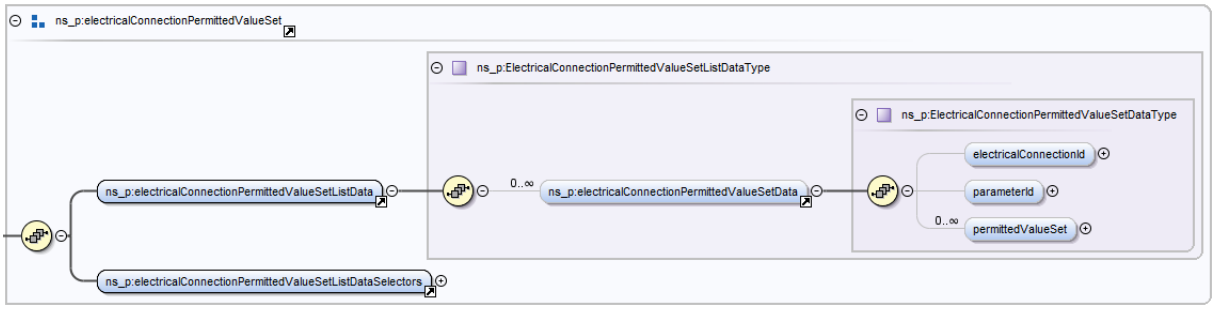


Figure 93: electricalConnectionPermittedValueSetListData function overview

5.3.11.3.2 Detailed description of elements

Element	Type	Description
electricalConnectionId	*Identifier "ElectricalConnectionIdType" (see* Table 339*).*	Electrical connection, this static information belongs to.
parameterId	*Identifier "ElectricalConnectionParameterIdType" (see* Table 339*).*	Relation to the according parameter.
permittedValuesSet (list)	*Common data type "ScaledNumberSetType". See section 3.10.1.9.*	The permittedValueSet allows to define an arbitrary set of permitted data. Compared to a simple value range the permittedValueSet also may have gaps. Please refer to the ScaledNumberSetType for more details.

Table 203: electricalConnectionPermittedValueSetListData function detailed description of elements

5.3.11.3.3 Available selectors

- electricalConnectionId
- parameterId

5.3.11.3.4 Examples

5.3.11.3.4.1.1 Read-reply

If one is interested in all electrical connection permitted value sets of a device, he could send the read command to the device, which will respond with something like the following command:

```
<electricalConnectionPermittedValueSetListData>
    <electricalConnectionPermittedValueSetData>
        <electricalConnectionId>1</electricalConnectionId>
        <parameterId>2</parameterId>
        <permittedValueSet>
            <range>
                <min>
                    <number>0</number>
                </min>
                <max>
                    <number>35</number>
                    <scale>2</scale>
                </max>
            </range>
        </permittedValueSet>
    </electricalConnectionPermittedValueSetData>
    <electricalConnectionPermittedValueSetData>
        <electricalConnectionId>1</electricalConnectionId>
        <parameterId>3</parameterId>
        <permittedValueSet>
            <value>
                <number>0</number>
            </value>
            <range>
                <min>
                    <number>5</number>
                    <scale>1</scale>
                </min>
                <max>
                    <number>2</number>
                    <scale>3</scale>
                </max>
            </range>
        </permittedValueSet>
    </electricalConnectionPermittedValueSetData>
</electricalConnectionPermittedValueSetListData>
```

5.3.11.4 electricalConnectionStateListData

5.3.11.4.1 General

Some electrical connection specific state information is modelled in this function.

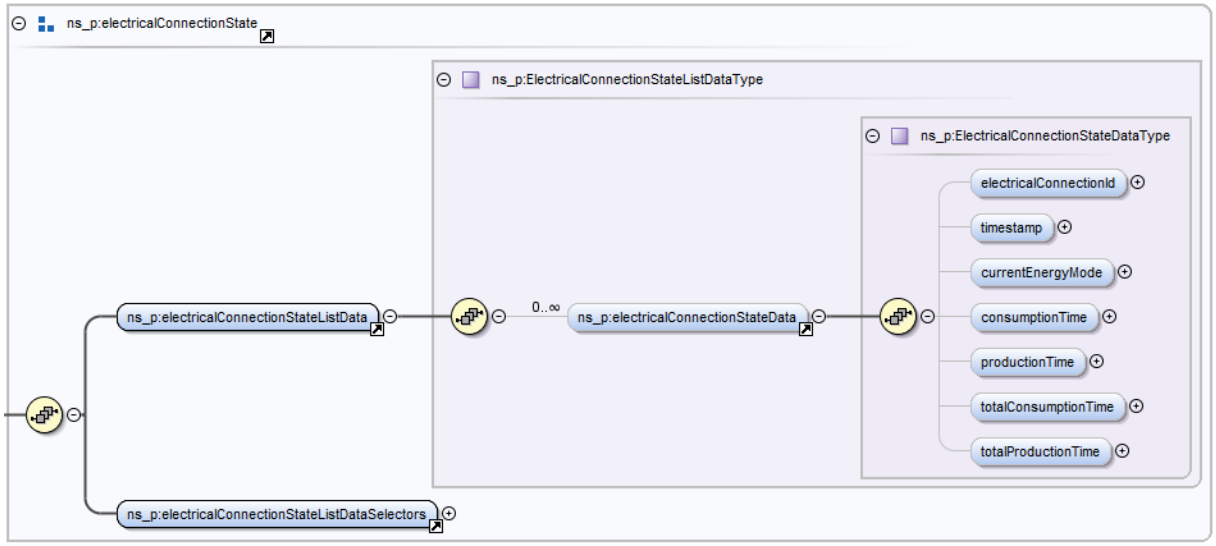


Figure 94: electricalConnectionStateListData function overview

5.3.11.4.2 Detailed description of elements

Element	Type	Description
electricalConnectionId	*Identifier "ElectricalConnectionIdType" (see* Table 339*).*	Electrical connection, this state information belongs to.
timestamp	*Common data type "AbsoluteOrRelativeTimeType". See section 3.10.2.3.*	Point in time, the state information was generated.
currentEnergyMode	*Common data type "EnergyModeType". See section 3.10.1.15.*	Whether a device is consuming, producing or idle is stated by this element.
consumptionTime	xs:duration (W3C standard type)	The duration, the device is in energy mode "consume" since the last reboot (see section 5.3.9.2).
productionTime	xs:duration (W3C standard type)	The duration, the device is in energy mode "produce" since the last reboot (see section 5.3.9.2).
totalConsumptionTime	xs:duration (W3C standard type)	The duration, the device is in energy mode "consume" since its *installationTime* (see section 5.3.9.4).
totalProductionTime	xs:duration (W3C standard type)	The duration, the device is in energy mode "producing" since its *installationTime* (see section 5.3.9.4).

Table 204: electricalConnectionStateListData function detailed description of elements

5.3.11.4.3 Available selectors

- electricalConnectionId

5.3.11.4.4 Examples

5.3.11.4.4.1 Read-reply

The state of a specific electrical connection can be requested with this read command:

```
<function>electricalConnectionStateListData</function>
<filter>
    <cmdControl>
        <partial/>
    </cmdControl>
    <electricalConnectionStateListDataSelectors>
        <electricalConnectionId>1</electricalConnectionId>
    </electricalConnectionStateListDataSelectors>
</filter>
<electricalConnectionStateListData/>
```

If the requested device has proper information, it will respond them with the following command:

```
<function>electricalConnectionStateListData</function>
<filter>
    <cmdControl>
        <partial/>
    </cmdControl>
</filter>
<electricalConnectionStateListData>
    <electricalConnectionStateData>
        <electricalConnectionId>1</electricalConnectionId>
        <timestamp>2014-05-12T17:55:30.0Z</timestamp>
        <currentEnergyMode>consume</currentEnergyMode>
        <consumptionTime>PT2H10M15S</consumptionTime>
        <productionTime>PT0S</productionTime>
        <totalConsumptionTime>P1Y3M2DT21H14M57S</totalConsumptionTime>
        <totalProductionTime>PT0S</totalProductionTime>
    </electricalConnectionStateData>
</electricalConnectionStateListData>
```

5.3.11.5 electricalConnectionDescriptionListData

5.3.11.5.1 General

Some electrical connection specific information, that are rather static, are modelled within this function.

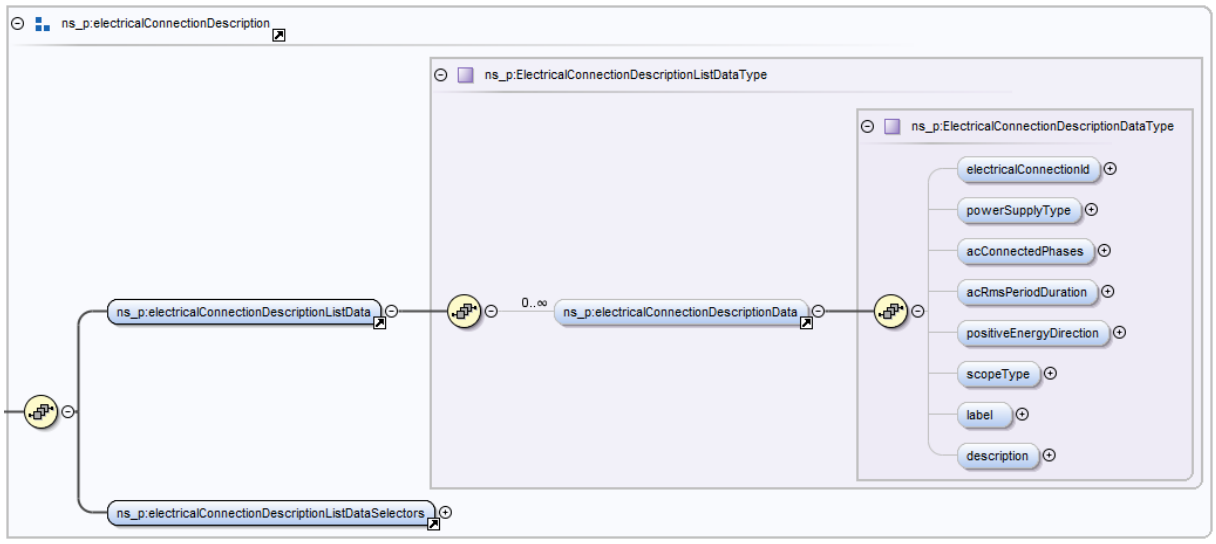


Figure 95: electricalConnectionStateDescriptionData function overview

5.3.11.5.2 Detailed description of elements

Element	Type	Description
electricalConnectionId	*Identifier "ElectricalConnectionIdType" (see* Table 339*).*	Electrical connection, this static information belongs to.
powerSupplyType	*Union* "ElectricalConnectionVoltageTypeType*":* - *Enum (see* Table 199*):* ElectricalConnectionVoltageTypeEnumType - EnumExtendType (see section 3.10.1.5)	States whether the electrical connection is of type "ac" or "dc".
acConnectedPhases	xs:unsignedInt (W3C standard type)	If powerSupplyType="ac", this element contains the number of phases, this electrical connection has (typically 1, 2 or 3).
acRmsPeriodDuration	xs:duration (W3C standard type)	If the "rms" value of a parameter combination (see element *acMeasurementVariant* in section 5.3.11.2.2) is measured, this element states the duration, the rms value is determined in.
positiveEnergyDirection	*Common data type "EnergyDirectionType". See section 3.10.1.13.*	This element states whether energy consumption or production will be counted as positive value.
scopeType	*Common data type "ScopeTypeType". See section 3.10.1.21.*	A certain meaning of the electrical connection.
label	*Common data type "LabelType". See section 3.10.1.2.*	A label for this electrical connection.

description	*Common data type "DescriptionType". See section 3.10.1.3.*	A description for this electrical connection.

Table 205: electricalConnectionDescriptionListData function detailed description of elements

5.3.11.5.3 Available selectors

- electricalConnectionId
- scopeType

5.3.11.5.4 Examples

5.3.11.5.4.1 Read-reply

If one is interested in all electrical connection descriptions of a device, he could send the read command to the device, which will respond with something like the following command:

```
<electricalConnectionDescriptionListData>
    <electricalConnectionDescriptionData>
        <electricalConnectionId>1</electricalConnectionId>
        <powerSupplyType>ac</powerSupplyType>
        <acConnectedPhases>1</acConnectedPhases>
        <acRmsPeriodDuration>PT0.1S</acRmsPeriodDuration>
        <positiveEnergyDirection>consume</positiveEnergyDirection>
        <label>mains connection</label>
        <description>1-phase mains connection of the device</description>
    </electricalConnectionDescriptionData>
    <electricalConnectionDescriptionData>
        <electricalConnectionId>2</electricalConnectionId>
        <powerSupplyType>dc</powerSupplyType>
        <positiveEnergyDirection>consume</positiveEnergyDirection>
        <label>battery connection</label>
        <description>Backup battery connection of the device</description>
    </electricalConnectionDescriptionData>
</electricalConnectionDescriptionListData>
```

5.3.12 HVAC

5.3.12.1 Introduction

Heating, Ventilation and Air Conditioning (HVAC) devices have some special functionality which is covered by this class.

All system functions (like heating, cooling, domestic hot water preparation (dhw), ventilation, etc.) have their specific HVAC operation modes (like on, off, auto, etc.). Relations between these are modelled as well as relations to (optional) power sequences, which represent a concrete run of a device's program.

HVAC overruns (like an one time domestic-hot-water (DHW) preparation), which do have an impact on the normal operation mode of a HVAC system function, are modelled in further SPINE functions of the HVAC Class.

Figure 96: HVAC function-group overview

5.3.12.2 *hvacSystemFunctionListData*

5.3.12.2.1 General

The hvacSystemFunctionListData function announces the current status of the HVAC system functions, including the current HVAC operation mode, a flag that indicates, if the HVAC operation mode can be changed by some external communications partner (client) and a flag that indicates whether an HVAC overrun is currently active or not.

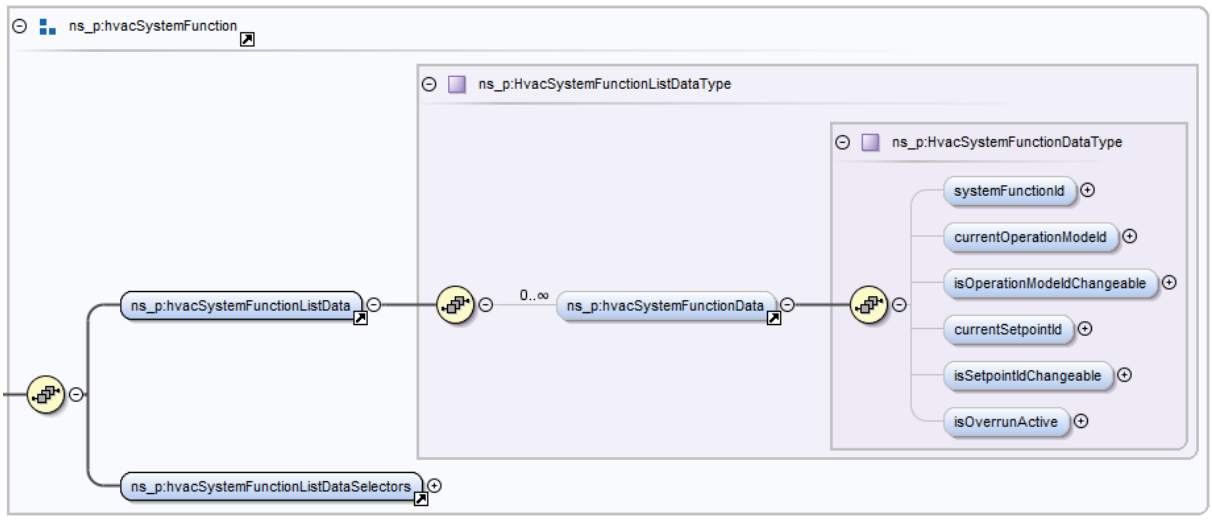


Figure 97: hvacSystemFunctionListData function overview

5.3.12.2.2 Detailed description of elements

Element	Type	Description

systemFunctionId	*Identifier "HvacSystemFunctionIdType" (see* Table 339*).*	Identifier for a specific HVAC system function.
currentOperationModeId	*Identifier "HvacOperationModeIdType" (see* Table 339*).*	States which HVAC operation mode is currently active for this specific HVAC system function
isOperationModeIdChangeable	xs:boolean (W3C standard type)	Denotes whether the current HVAC operation mode identifier is changeable (by some external communications partner) or not.
currentSetpointId	*Identifier "SetpointIdType" (see* Table 339*).*	States which setpoint is active for this specific HVAC system function
isSetpointIdChangeable	xs:boolean (W3C standard type)	Denotes whether the current setpoint identifier is changeable (by some external communications partner) or not.
isOverrunActive	xs:boolean (W3C standard type)	If an HVAC overrun is currently active, this element is set to *true*. The HVAC overrun itself is modelled with other functions (see section 5.3.12.8 and 5.3.12.9) and may be located on a different feature or even a different entity or device.

Table 206: hvacSystemFunctionListData function detailed description of elements

5.3.12.2.3 Available selectors

- systemFunctionId

5.3.12.2.4 Examples

5.3.12.2.4.1 Notify

When the *currentOperationMode* of a HVAC system function changes, the server sends a notify to all its subscribed clients (the “filter/partial” part to announce the “restricted function exchange is simplified by “...” just to improve the readability”):

```
...
<hvacSystemFunctionListData>
    <hvacSystemFunctionData>
        <systemFunctionId>2</systemFunctionId>
        <currentOperationModeId>3</currentOperationModeId>
    </hvacSystemFunctionData>
</hvacSystemFunctionListData>
```


5.3.12.3 hvacSystemFunctionOperationModeRelationListData

5.3.12.3.1 General

Each HVAC system function has its own relation to all for this HVAC system function available HVAC operation modes. E.g. there may be three HVAC operation modes available on the feature: *on*, *off*, *auto* (see section 5.3.12.7). HVAC System function 1 only supports HVAC operation mode 1 (*on*) and 2 (*off*). HVAC System function 2 supports all three HVAC operation modes. The enumeration of the available HVAC system functions is described in section 5.3.12.6.2.

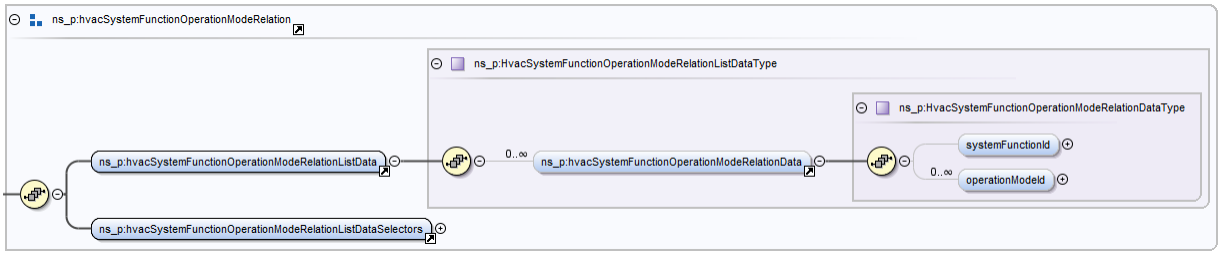


Figure 98: hvacSystemFunctionOperationModeRelationListData function overview

5.3.12.3.2 Detailed description of elements

Element	Type	Description
systemFunctionId	*Identifier "HvacSystemFunctionIdType" (see* Table 339*).*	Identifier for a specific HVAC system function.
operationModeId (list)	*Identifier "HvacOperationModeIdType" (see* Table 339*).*	Lists the HVAC operation modes that are available (selectable) for a specific HVAC system function.

Table 207: hvacSystemFunctionOperationModeRelationListData function detailed description of elements

5.3.12.3.3 Available selectors

- systemFunctionId

5.3.12.3.4 Examples

5.3.12.3.4.1 Read-reply

If one is interested in the supported HVAC operation modes of all HVAC system functions, he could send a standard read command to the server. The reply could look like this:

```
<hvacSystemFunctionOperationModeRelationListData>
    <hvacSystemFunctionOperationModeRelationData>
        <systemFunctionId>1</systemFunctionId>
        <operationModeId>1</operationModeId>
        <operationModeId>2</operationModeId>
        <operationModeId>3</operationModeId>
    </hvacSystemFunctionOperationModeRelationData>
    <hvacSystemFunctionOperationModeRelationData>
        <systemFunctionId>2</systemFunctionId>
        <operationModeId>1</operationModeId>
        <operationModeId>2</operationModeId>
```



```
    </hvacSystemFunctionOperationModeRelationData>
</hvacSystemFunctionOperationModeRelationListData>
```

5.3.12.4 hvacSystemFunctionSetpointRelationListData

5.3.12.4.1 General

The setpoint class (see section 5.3.21) may be used to model some HVAC relevant setpoints like desired room temperature. An HVAC system function may relate to some of these setpoints. The relation is modelled with this SPINE function.

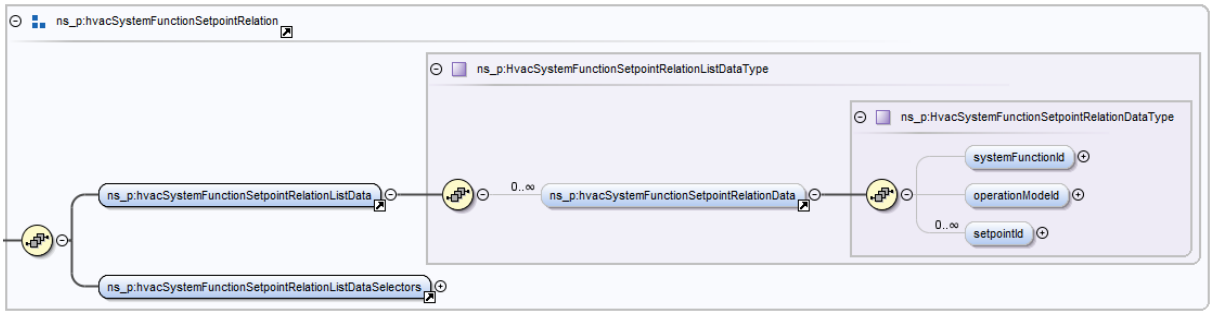


Figure 99: hvacSystemFunctionSetpointRelationListData function overview

5.3.12.4.2 Detailed description of elements

Element	Type	Description
systemFunctionId	*Identifier "HvacSystemFunctionIdType" (see* Table 339*).*	Identifier for a specific HVAC system function.
operationModeId	*Identifier "HvacOperationModeIdType" (see* Table 339*).*	Identifier for a specific HVAC operation mode.
setpointId (list)	*Identifier "SetpointIdType" (see* Table 339*).*	List of setpoints that are related with a specific HVAC system function.

Table 208: hvacSystemFunctionSetpointRelationListData function detailed description of elements

5.3.12.4.3 Available selectors

- systemFunctionId
- operationModeId

5.3.12.4.4 Examples

5.3.12.4.4.1 Read-reply

If one is interested in the related setpoints of all HVAC system functions, he could send a standard read command to the server. The reply could look like this:

```
<hvacSystemFunctionSetpointRelationListData>
    <hvacSystemFunctionSetpointRelationData>
```



```
            <systemFunctionId>1</systemFunctionId>
            <operationModeId>1</operationModeId>
            <setpointId>1</setpointId>
        </hvacSystemFunctionSetpointRelationData>
        <hvacSystemFunctionSetpointRelationData>
            <systemFunctionId>1</systemFunctionId>
            <operationModeId>2</operationModeId>
            <setpointId>2</setpointId>
            <setpointId>3</setpointId>
        </hvacSystemFunctionSetpointRelationData>
        <hvacSystemFunctionSetpointRelationData>
            <systemFunctionId>1</systemFunctionId>
            <operationModeId>3</operationModeId>
            <setpointId>4</setpointId>
        </hvacSystemFunctionSetpointRelationData>
</hvacSystemFunctionSetpointRelationListData>
```

5.3.12.5 hvacSystemFunctionPowerSequenceRelationListData

5.3.12.5.1 General

Power sequences (see section 5.3.19) describe tasks of a functionality in detail. An HVAC system function may relate to some of these power sequences. The relation is modelled with this function.

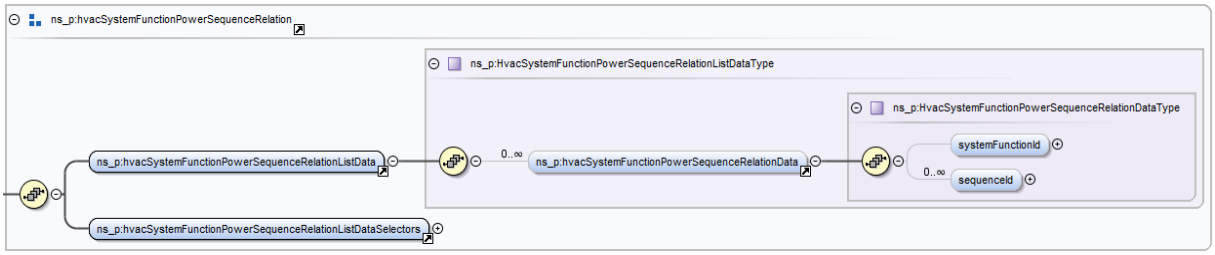


Figure 100: hvacSystemFunctionPowerSequenceRelationListData function overview

5.3.12.5.2 Detailed description of elements

Element	Type	Description
systemFunctionId	*Identifier "HvacSystemFunctionIdType" (see* Table 339*).*	Identifier for a specific HVAC system function.
sequenceId (list)	*Identifier "PowerSequenceIdType" (see* Table 339*).*	List of power sequences that are related with a specific HVAC system function.

Table 209: hvacSystemFunctionPowerSequenceRelationListData function detailed description of elements

5.3.12.5.3 Available selectors

- systemFunctionId

5.3.12.5.4 Examples

5.3.12.5.4.1 Read-reply

If one is interested in the related power sequences of all HVAC system functions, he could send a standard read command to the server. The reply could look like this:

```
<hvacSystemFunctionPowerSequenceRelationListData>
    <hvacSystemFunctionPowerSequenceRelationData>
        <systemFunctionId>1</systemFunctionId>
        <sequenceId>1</sequenceId>
        <sequenceId>2</sequenceId>
        <sequenceId>11</sequenceId>
    </hvacSystemFunctionPowerSequenceRelationData>
    <hvacSystemFunctionPowerSequenceRelationData>
        <systemFunctionId>2</systemFunctionId>
        <sequenceId>5</sequenceId>
    </hvacSystemFunctionPowerSequenceRelationData>
</hvacSystemFunctionPowerSequenceRelationListData>
```

5.3.12.6 hvacSystemFunctionDescriptionListData

5.3.12.6.1 General

The rather static information about the HVAC system functions are modelled within this SPINE function.

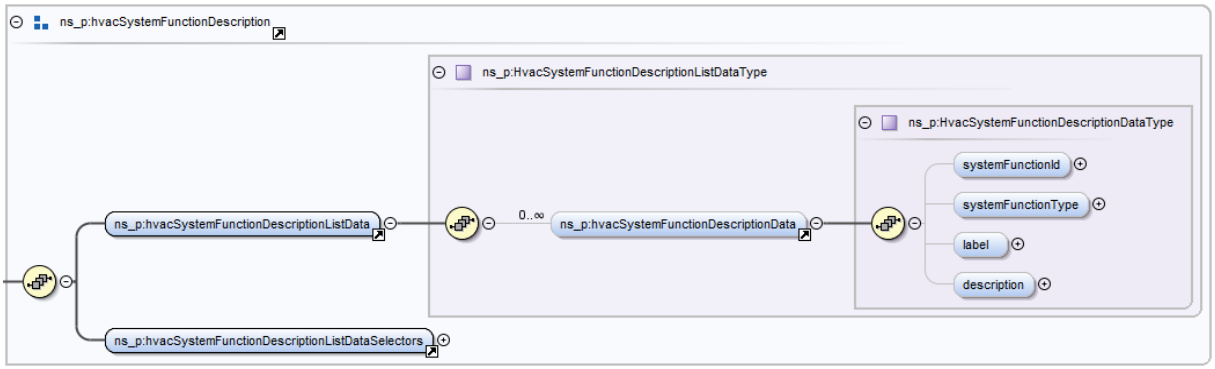


Figure 101: hvacSystemFunctionDescriptionListData function overview

5.3.12.6.2 Detailed description of elements

Element	Type	Description
systemFunctionId	*Identifier "HvacSystemFunctionIdType" (see* Table 339*).*	Identifier for a specific HVAC system function.
systemFunctionType	*Union* "HvacSystemFunctionTypeType*":* *- Enum (see* Table 211*):* HvacSystemFunctionTypeEnumType - EnumExtendType (see section 3.10.1.5)	Denotes the type of the HVAC system function.
label	*Common data type "LabelType". See section 3.10.1.2.*	A short label of the HVAC system function.

description	*Common data type "DescriptionType". See section 3.10.1.3.*	Descriptive information on the HVAC system function.

Table 210: hvacSystemFunctionDescriptionListData function detailed description of elements

Enumeration **HvacSystemFunctionTypeEnumType**:

Value	Description
heating	HVAC system function for space heating.
cooling	HVAC system function for space cooling.
ventilation	HVAC system function for ventilation.
dhw	HVAC system function for domestic hot water preparation.

Table 211: Enumeration HvacSystemFunctionTypeEnumType

5.3.12.6.3 Available selectors

- systemFunctionId

5.3.12.6.4 Examples

5.3.12.6.4.1 Read-reply

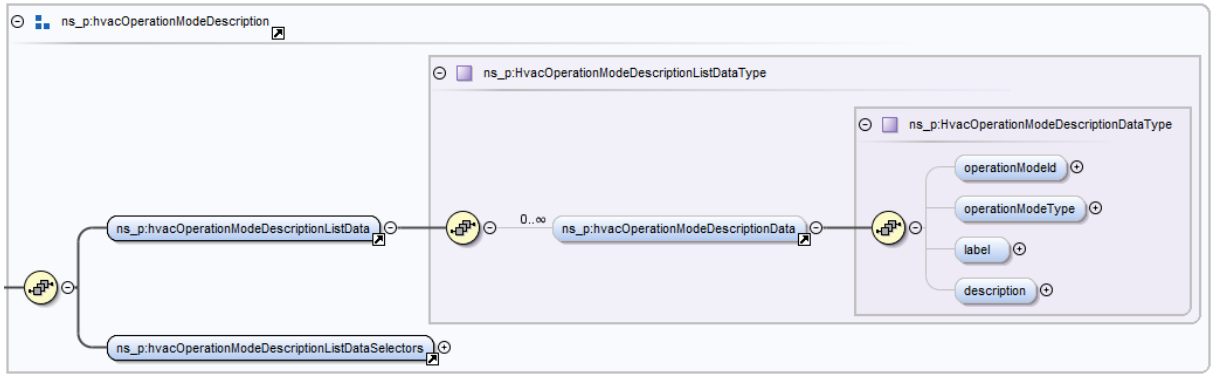
If one is interested in the description of all HVAC system functions, he could send a standard read command to the server. The reply could look like this:

```
<hvacSystemFunctionDescriptionListData>
    <hvacSystemFunctionDescriptionData>
        <systemFunctionId>1</systemFunctionId>
        <systemFunctionType>heating</systemFunctionType>
        <label>Heating function</label>
    </hvacSystemFunctionDescriptionData>
    <hvacSystemFunctionDescriptionData>
        <systemFunctionId>2</systemFunctionId>
        <systemFunctionType>dhw</systemFunctionType>
        <label>Domestic hot water</label>
    </hvacSystemFunctionDescriptionData>
</hvacSystemFunctionDescriptionListData>
```

5.3.12.7 hvacOperationModeDescriptionListData

5.3.12.7.1 General

HVAC operation modes are used to describe which behaviour the HVAC system functions have. Each HVAC operation mode may be used by several HVAC system functions.



7135

7136 *Figure 102: hvacOperationModeDescriptionListData function overview*

7137

7138 *5.3.12.7.2 Detailed description of elements*

Element	Type	Description
operationModeId	*Identifier "HvacOperationModeIdType" (see* Table 339*).*	Identifier for the HVAC operation mode.
operationModeType	*Union "*HvacOperationModeTypeType*":* *- Enum (see* Table 213*):* HvacOperationModeTypeEnumType - EnumExtendType (see section 3.10.1.5)	Denotes the type of the HVAC operation mode.
label	*Common data type "LabelType". See section 3.10.1.2.*	A short label of the HVAC operation mode.
description	*Common data type "DescriptionType". See section 3.10.1.3.*	Descriptive information on the HVAC operation mode.

7139 *Table 212: hvacOperationModeDescriptionListData function detailed description of elements*

7140 Enumeration **HvacOperationModeTypeEnumType**:

Value	Description	
auto	**HVAC System Function**	**Meaning**
	heating / cooling / dhw / ventilation	HVAC System function is controlled depending on a time table (i.e. the current setpoint for the HVAC system function is changed depending on a time table)
on	**HVAC System Function**	**Meaning**
	heating / cooling / dhw / ventilation	HVAC System function is controlled (e.g. Setpoint) independant from time tables (i.e. the current setpoint for the HVAC system function is not depending on any time table).
off	**HVAC System Function**	**Meaning**
	heating / dhw	Protection mode against freezing (no physically turning off). The (temperature) setpoint for freezing protection may be stated with a setpointId.
	cooling	Space cooling function not active.
	ventilation	Ventilation function not active (protection functions like humidity protection may be active).
eco	**HVAC System Function**	**Meaning**

	Heating / cooling / dhw / ventilation	HVAC System function is controlled (e.g. current Setpoint) independant from time tables (i.e. the current setpoint for the HVAC system function is not depending on any time table). HVAC System function is controlled by a different setpoint as for “on”

Table 213: Enumeration HvacOperationModeTypeEnumType

5.3.12.7.3 Available selectors

- operationModeId

5.3.12.7.4 Examples

5.3.12.7.4.1 Read-reply

If one is interested in the description of all HVAC operation modes, he could send a standard read command to the server. The reply could look like this:

```
<hvacOperationModeDescriptionListData>
    <hvacOperationModeDescriptionData>
        <operationModeId>1</operationModeId>
        <operationModeType>on</operationModeType>
    </hvacOperationModeDescriptionData>
    <hvacOperationModeDescriptionData>
        <operationModeId>2</operationModeId>
        <operationModeType>off</operationModeType>
    </hvacOperationModeDescriptionData>
    <hvacOperationModeDescriptionData>
        <operationModeId>3</operationModeId>
        <operationModeType>auto</operationModeType>
    </hvacOperationModeDescriptionData>
</hvacOperationModeDescriptionListData>
```

5.3.12.8 hvacOverrunListData

5.3.12.8.1 General

The normal HVAC operation may be overrided by some HVAC specific functionality. There are some HVAC overruns defined by the EEBus Initiative e.V., but vendors may add further vendor-specific HVAC overruns, too (which are then out of interoperability scope).

When the HVAC overrun is active, some (not all! See section 5.3.12.9) HVAC system functions have a specific behaviour, other than the normal operation would be. After the HVAC overrun is inactive again, the HVAC system returns to its normal operation.

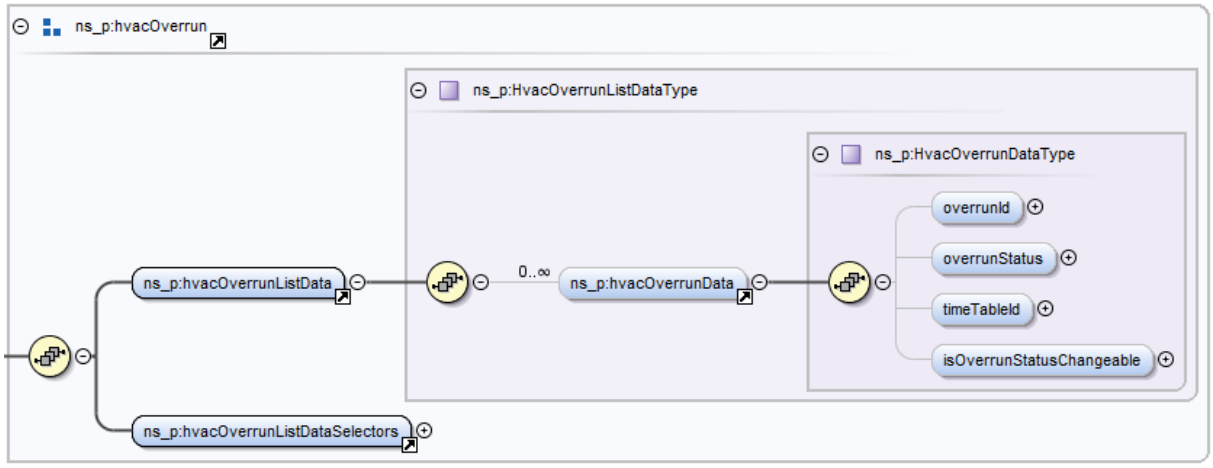


Figure 103: hvacOverrunListData function overview

5.3.12.8.2 Detailed description of elements

Element	Type	Description
overrunId	*Identifier "HvacOverrunIdType" (see* Table 339*).*	Identifier for the HVAC overrun.
overrunStatus	*Union* "HvacOverrunStatusType"*:* *- Enum (see* Table 215*):* HvacOverrunStatusEnumType - EnumExtendType (see section 3.10.1.5)	The current status of the HVAC overrun.
timeTableId	*Identifier "TimeTableIdType" (see* Table 339*).*	Reference to a time table. Needed, if an overrun has some start and end point(s).
isOverrunStatusChangeable	xs:boolean (W3C standard type)	Whether the overrun status is changeable by a client or not is denoted in this element.

Table 214: hvacOverrunListData function detailed description of elements

Enumeration **HvacOverrunStatusEnumType**:

Value	Description
active	HVAC overrun triggered, but not started yet.
running	HVAC overrun running.
finished	HVAC overrun finished.
inactive	HVAC overrun inactive.

Table 215: Enumeration HvacOverrunStatusEnumType

5.3.12.8.3 Available selectors

- overrunId

5.3.12.8.4 Examples

5.3.12.8.4.1 Read-reply

If one is interested in the status of all HVAC overruns, he could send a standard read command to the server. The reply could look like this:

```
<hvacOverrunListData>
    <hvacOverrunData>
        <overrunId>1</overrunId>
        <overrunStatus>inactive</overrunStatus>
        <timeTableId>5</timeTableId>
    </hvacOverrunData>
    <hvacOverrunData>
        <overrunId>2</overrunId>
        <overrunStatus>active</overrunStatus>
    </hvacOverrunData>
</hvacOverrunListData>
```

5.3.12.9 hvacOverrunDescriptionListData

5.3.12.9.1 General

Each HVAC overrun is defined by its description part. Here, the HVAC overrun type is specified as well as the list of HVAC system functions that are affected by the HVAC overrun (if this information is available).

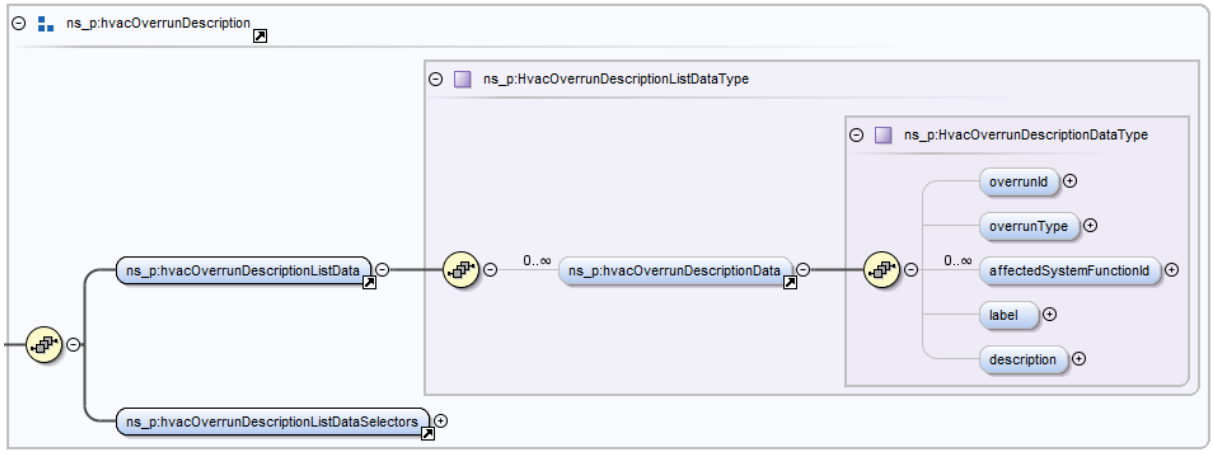


Figure 104: hvacOverrunDescriptionListData function overview

5.3.12.9.2 Detailed description of elements

Element	Type	Description
overrunId	*Identifier "HvacOverrunIdType" (see* Table 339*).*	Identifier for the HVAC overrun.
overrunType	*Union "*HvacOverrunTypeType*":* *- Enum (see* Table 217*):* HvacOverrunTypeEnumType - EnumExtendType (see section 3.10.1.5)	The type of the overrun.

affectedSystemFunctionId (list)	*Identifier "HvacSystemFunctionIdType" (see* Table 339*).*	Identifiers of all HVAC system functions that are affected by this HVAC overrun.
label	*Common data type "LabelType". See section 3.10.1.2.*	A short label of the HVAC overrun.
description	*Common data type "DescriptionType". See section 3.10.1.3.*	Descriptive information on the HVAC overrun.

7209 *Table 216: hvacOverrunDescriptionListData function detailed description of elements*

7210 Enumeration **HvacOverrunTypeEnumType**:

Value	Description
oneTimeDhw	A one-time domestic hot water loading (independend from current HVAC operation mode or time table).
party	During a party, the HVAC system will stay in the normal day-operation mode and not turn off at the usual time. Example: HVAC system function “heating” is controlled by the day setpoint, independend from operation mode and time table, until HVAC overrun “party” is finished.
sgReadyCondition1	The SGReady condition 1 will be applied. Please consider [SGReady].
sgReadyCondition3	The SGReady condition 3 will be applied. Please consider [SGReady].
sgReadyCondition4	The SGReady condition 4 will be applied. Please consider [SGReady].
oneDayAway	Overrides the normal daily routine to a out-of-home configuration (e.g. eco-mode the whole day).
oneDayAtHome	Overrides the normal daily routine to an at-home configuration (e.g. on-mode the whole day).
oneTimeVentilation	Activates the ventilation once, independent from the configured routine.
hvacSystemOff	Deactivates the HVAC system (frost protection is still active and the system is in a network-standby mode, in which it is reachable for re-configuration).

7211 *Table 217: Enumeration HvacOverrunTypeEnumType*

7212

7213 *5.3.12.9.3 Available selectors*

7214 - overrunId

7215

7216 *5.3.12.9.4 Examples*

7217 *5.3.12.9.4.1 Read-reply*

7218 If one is interested in the description of all HVAC overruns, he could send a standard read command
7219 to the server. The reply could look like this:

```
7220 <hvacOverrunDescriptionListData>
7221     <hvacOverrunDescriptionData>
7222         <overrunId>1</overrunId>
7223         <overrunType>party</overrunType>
7224         <affectedSystemFunctionId>1</affectedSystemFunctionId>
7225         <affectedSystemFunctionId>2</affectedSystemFunctionId>
7226         <label>label0</label>
7227         <description>description0</description>
7228     </hvacOverrunDescriptionData>
7229     <hvacOverrunDescriptionData>
```



```
        <overrunId>2</overrunId>
        <overrunType>oneTimeDhw</overrunType>
        <affectedSystemFunctionId>2</affectedSystemFunctionId>
        <label>label1</label>
        <description>description1</description>
    </hvacOverrunDescriptionData>
</hvacOverrunDescriptionListData>
```

5.3.13 Identification

5.3.13.1 Introduction

The Identification Class can be used to express certain identification values of the parent Entity.

Figure 105: Identification function-group overview

5.3.13.2 identificationListData

5.3.13.2.1 General

The identificationListData function allows to express certain identification values of the corresponding Entity (the parent Entity of the Identification Feature).

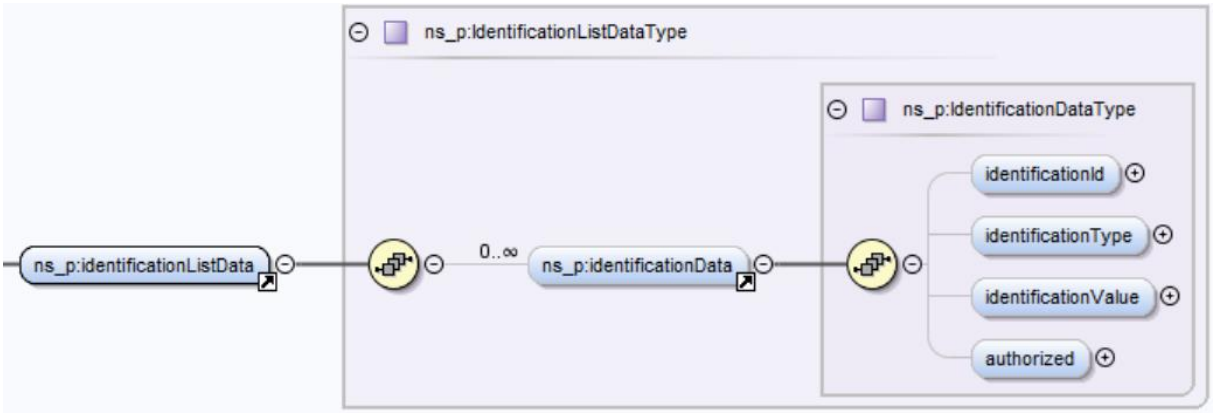


Figure 106: identificationListData function overview

5.3.13.2.2 Detailed description of elements

Element	Type	Description
identificationId	*Identifier "IdentificationIdType" (see* Table 339*).*	The identificationId is used to identify the different list entries within IdentificationListData.
identificationType	*Union* "IdentificationTypeType"*:* *- Enum (see* Table 219*):* IdentificationTypeEnumType - EnumExtendType (see section 3.10.1.5)	The identificationType allows to describe a certain type of identification, e.g. a MAC address or an RFID tag.

identificationValue	*Simple type "IdentificationValueType" (restriction of "xs:string")*	This holds the value of the identification and therefore is the essential element of IdentificationData.
authorized	xs:boolean	The "authorized" element can be used to directly authorize an identification value with a write command.

Table 218: identificationListData Element rules

Enumeration **IdentificationTypeEnumType**:

Value	Description
eui48	eui48 MAC address
eui64	eui64 MAC address
userRfidTag	RFID Tag

Table 219: Enumeration IdentificationTypeEnumType

5.3.13.2.3 Available selectors

- identificationId

5.3.13.2.4 Examples

5.3.13.2.4.1 Notify

If the content of a list entries elements changes, the information is sent to all subscribed resources:

```
<identificationListData>
    <identificationData>
        <identificationId>1</identificationId>
        <identificationType>eui64</identificationType>
        <identificationValue>12-AB-34-CD-56-EF-78-90</identificationValue>
        <authorized>true</authorized>
    </identificationData>
</identificationListData>
```

5.3.14 LoadControl

5.3.14.1 Introduction

The LoadControl class has two different approaches to control the load of an appliance:

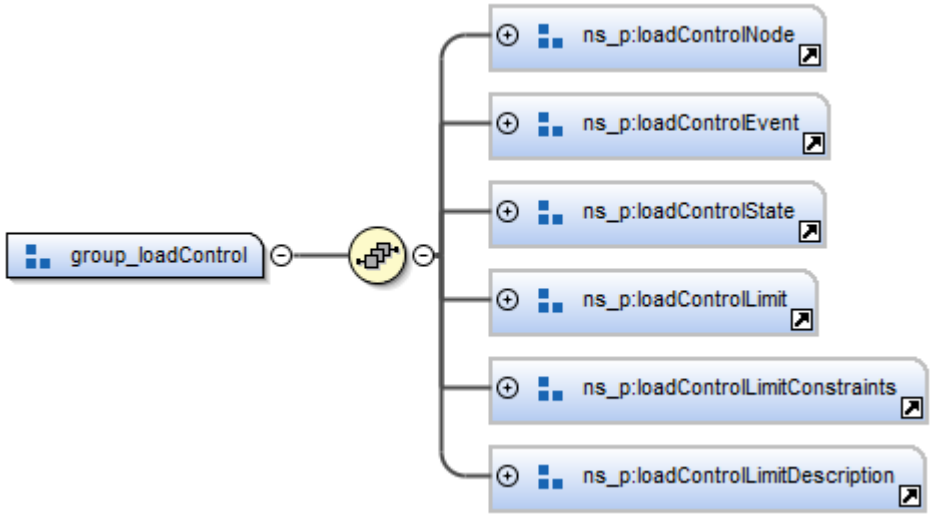
- Event-based load control with "loadControlEvent" (see section 5.3.14.3) and "loadControlState" (see section 5.3.14.4)
- Limit-based load control with "loadControlLimit" (see section 5.3.14.5) and "loadControlLimitDescription" (see section 5.3.14.7)

The event-based load control is used to change the energy demand (load) or production of a household by some external contract partner or energy manager. The household or device (owner) may reject the change commands, but then may have to pay more to the external partner or receive less reward. The concept defines two function-groups. One (*loadControlEvent*) that keeps the

7282 received change commands (a household or device may store more than one command, which may
7283 be active in different time periods; these commands are requestable by some other client side) and
7284 another one (*loadControlState*) for the status of these commands (where more than one could be
7285 requestable, too). Possible commands are: pausing / resuming the demand or production, decreasing
7286 / increasing it, an emergency mode (where the electricity grid or local house power supply is in a
7287 critical condition and only an immediate substantial reduction of the energy consumption or feeding
7288 of energy into the electricity grid can prevent a blackout situation) and the "back-to-normal"
7289 command.

7290 The limit-based load control is used where limits (changeable by a client) shall affect the behaviour of
7291 a node automatically. E.g. the power demand of an appliance could be limited to a certain value to
7292 protect the household of an overload.

7293 Additionally with the information given in loadControlNode (see section 5.3.14.2), a node can define
7294 whether it is (at the moment) remote controllable or not.



7295

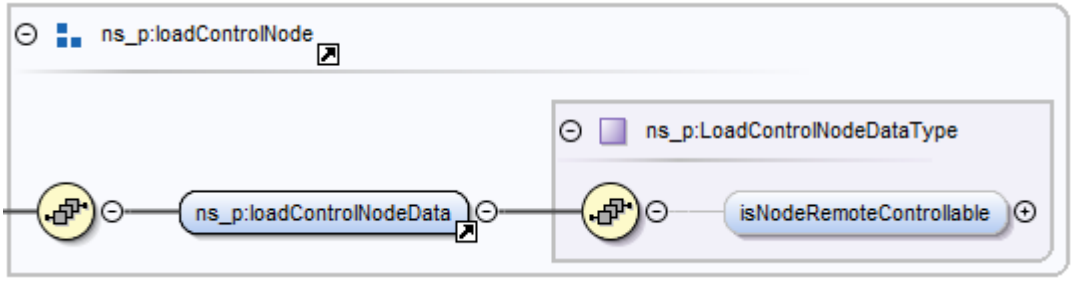
7296 *Figure 107: LoadControl function-group overview*

7297

7298 ### *5.3.14.2 loadControlNodeData*

7299 #### *5.3.14.2.1 General*

7300 The loadControlNodeData includes node-wide information related to load control (e.g. whether the
7301 node is remote controllable).



7302

7303 *Figure 108: loadControlNodeData function overview*

7304

7305 *5.3.14.2.2 Detailed description of elements*

Element	Type	Description
isNodeRemoteControllable	xs:boolean (W3C standard type)	Defines whether the node is (at the moment) remote controllable by a client, or not.

7306 *Table 220: loadControlNodeData function detailed description of elements*

7307

7308 *5.3.14.2.3 Available selectors*

7309 None.

7310

7311 *5.3.14.2.4 Examples*

7312 *5.3.14.2.4.1 Notify*

7313 If the content of the element isNodeRemoteControllable changes (e.g. due to a configuration change
7314 by the user), the information is sent to all subscribed resources:

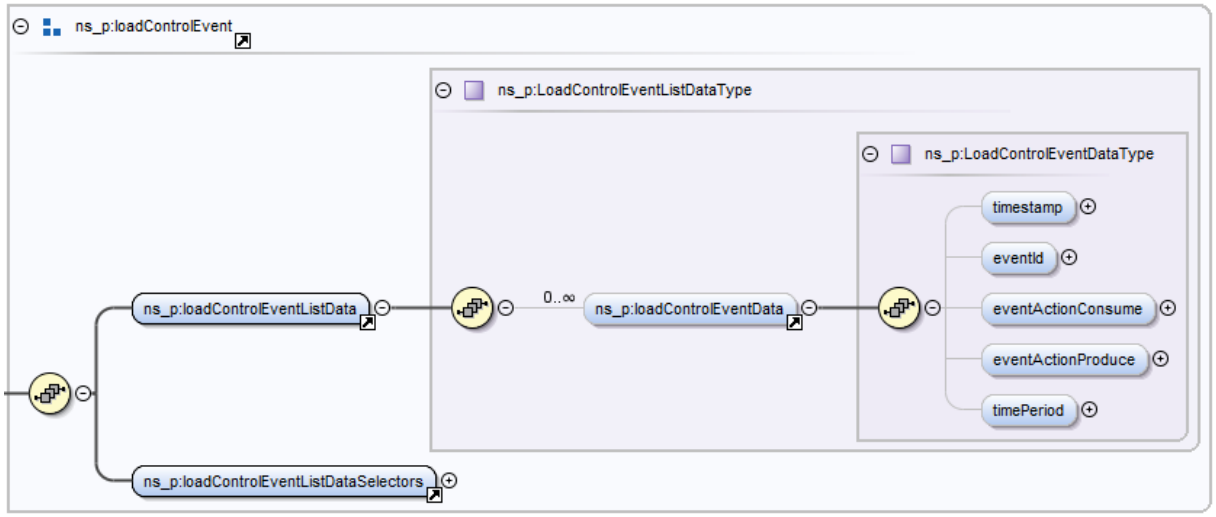
```
7315 <loadControlNodeData>
7316     <isNodeRemoteControllable>true</isNodeRemoteControllable>
7317 </loadControlNodeData>
```

7318

7319 ***5.3.14.3 loadControlEventListData***

7320 *5.3.14.3.1 General*

7321 With the loadControlEventListData function, a household or device can show the commands (events)
7322 it received to change its load demand or energy production (or both). These commands contain
7323 rather simple instructions like "reduce", "increase", "pause", "resume", etc. This function can also be
7324 used to model a single command from another point (external contract partner or energy manager,
7325 e.g.) to the houshold or device (this could be a "write" operation to add a new
7326 "loadControlEventData" to the loadControlEventListData instance, e.g.).



7327

7328 *Figure 109: loadControlEventListData function overview*

7329

7330 *5.3.14.3.2 Detailed description of elements*

Element	Type	Description
timestamp	*Common data type "AbsoluteOrRelativeTimeType". See section 3.10.2.3.*	The time when the message was generated.
eventId	*Identifier "LoadControlEventIdType" (see* Table 339*).*	An incrementing number, enabling the linking of a *loadControlState* command to a *loadControlEvent* command.
eventActionConsume	*Union* "LoadControlEventActionType*":* *- Enum (see* Table 222*):* LoadControlEventActionEnumType - EnumExtendType (see section 3.10.1.5)	The action for consuming that shall be applied by the receiving device (feature server).
eventActionProduce	*Union* "LoadControlEventActionType*":* *- Enum (see* Table 222*):* LoadControlEventActionEnumType - EnumExtendType (see section 3.10.1.5)	The action for producing that shall be applied by the receiving device (feature server).
timePeriod	*Common data type "TimePeriodType". See section 3.10.2.1.*	The period of time, the event shall be active (e.g. from 12:00 to 14:00 or with relative times, from 1 hour in the future to 3 hours in the future).

7331 *Table 221: loadControlEventListData function detailed description of elements*

7332 Enumeration **LoadControlEventActionEnumType**:

Value	Description
pause	Demanding the pausing of consumption / production.
resume	Demanding the resumption of consumption / production.
reduce	Demanding the reduction of consumption / production.
increase	Demanding the increasing of consumption / production.

emergency	Demanding the reduction of consumption to the absolutely minimum level that is possible without risking anything. Not to be used for production.
normal	Demanding the return to the normal operation.

Table 222: Enumeration LoadControlEventActionEnumType

5.3.14.3.3 Available selectors

- timestampInterval
- eventId

5.3.14.3.4 Examples

5.3.14.3.4.1 Write (partial)

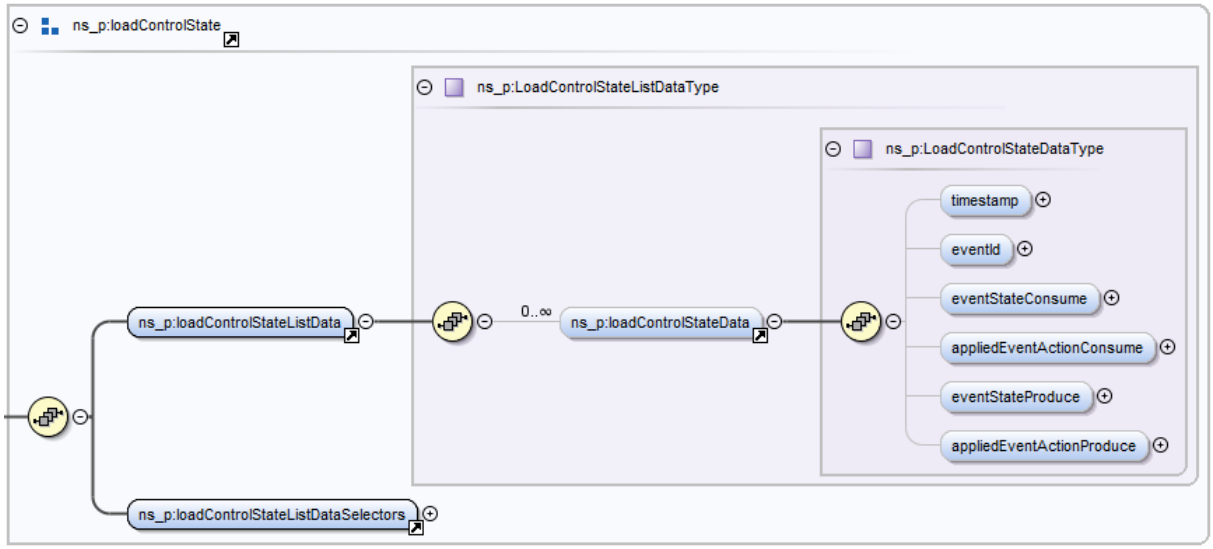
If an energy provider wants a household to reduce its consumption and increase its production for a certain period of time, it could send the following command to the CEM:

```
<function>loadControlEventListData</function>
<filter>
    <cmdControl>
        <partial/>
    </cmdControl>
</filter>
<loadControlEventListData>
    <loadControlEventData>
        <timestamp>2015-10-15T18:40:23.2Z</timestamp>
        <eventId>1</eventId>
        <eventActionConsume>reduce</eventActionConsume>
        <eventActionProduce>increase</eventActionProduce>
        <timePeriod>
            <startTime>PT1H10M</startTime>
            <endTime>PT3H15M</endTime>
        </timePeriod>
    </loadControlEventData>
</loadControlEventListData>
```

5.3.14.4 loadControlStateListData

5.3.14.4.1 General

A device that supports being controlled by LoadControl mechanisms (see function "loadControlEventListData") should express the states it currently assigns to each event.



7366

7367 *Figure 110: loadControlStateListData function overview*

7368

7369 *5.3.14.4.2 Detailed description of elements*

Element	Type	Description
timestamp	*Common data type "AbsoluteOrRelativeTimeType". See section 3.10.2.3.*	The time when the message was generated.
eventId	*Identifier "LoadControlEventIdType" (see* Table 339*).*	An incrementing number, enabling the linking of a *loadControlState* command to a *loadControlEvent* command.
eventStateConsume	*Union* "LoadControlEventStateType"*:* - *Enum (see* Table 224*):* LoadControlEventStateEnumType - EnumExtendType (see section 3.10.1.5)	Current state for consumption of the event.
appliedEventActionConsume	*Union* "LoadControlEventActionType"*:* - *Enum (see* Table 222*):* LoadControlEventActionEnumType - EnumExtendType (see section 3.10.1.5)	The event action for consumption, actually applied by the server.
eventStateProduce	*Union* "LoadControlEventStateType"*:* - *Enum (see* Table 224*):* LoadControlEventStateEnumType - EnumExtendType (see section 3.10.1.5)	Current state for production of the event.
appliedEventActionProduce	*Union* "LoadControlEventActionType"*:* - *Enum (see* Table 222*):* LoadControlEventActionEnumType - EnumExtendType (see section 3.10.1.5)	The event action for production, actually applied by the server.

Table 223: loadControlStateListData function detailed description of elements

Enumeration **LoadControlEventStateEnumType**:

Value	Description
eventAccepted	The server accepted the event command. It has not started yet!
eventStarted	The server started the event, meaning that, e.g., the server has started to reduce its consumption or is increasing its production.
eventStopped	The server stopped the event due to reaching the end time, of the event (stated in the *timePeriod* element of the *loadControlEvent* command).
eventRejected	The server rejected the command due to some reason, not stated here. It depends on the contract of the household owner how often he may reject a *loadControlEvent* command.
eventCancelled	The server cancelled the event due to some reason, not stated here.
eventError	An error occurred during executing the event.

Table 224: Enumeration LoadControlEventStateEnumType

5.3.14.4.3 Available selectors

- timestampInterval
- eventId

5.3.14.4.4 Examples

5.3.14.4.4.1 Notify

As a result of the load-reducing-command example from section 5.3.14.3.4.1, the CEM could inform the DSO with the following (complete) notification:

```
<loadControlStateListData>
    <loadControlStateData>
        <timestamp>2015-10-15T18:40:23.9Z</timestamp>
        <eventId>1</eventId>
        <eventStateConsume>eventAccepted</eventStateConsume>
        <appliedEventActionConsume>reduce</appliedEventActionConsume>
        <eventStateProduce>eventAccepted</eventStateProduce>
        <appliedEventActionProduce>increase</appliedEventActionProduce>
    </loadControlStateData>
</loadControlStateListData>
```

Following this example, exact one hour and ten minutes later, the CEM would send the following (complete) notification to the DSO:

```
<loadControlStateListData>
    <loadControlStateData>
        <timestamp>2015-10-15T19:50:23.9Z</timestamp>
        <eventId>1</eventId>
        <eventStateConsume>eventStarted</eventStateConsume>
        <appliedEventActionConsume>reduce</appliedEventActionConsume>
        <eventStateProduce>eventStarted</eventStateProduce>
        <appliedEventActionProduce>increase</appliedEventActionProduce>
    </loadControlStateData>
</loadControlStateListData>
```


5.3.14.5 loadControlLimitListData

5.3.14.5.1 General

The loadControlLimitListData function provides the value of the limit (and an optional time period for that limit) and further information whether the limit is active or not and if it is changeable by a client or not.

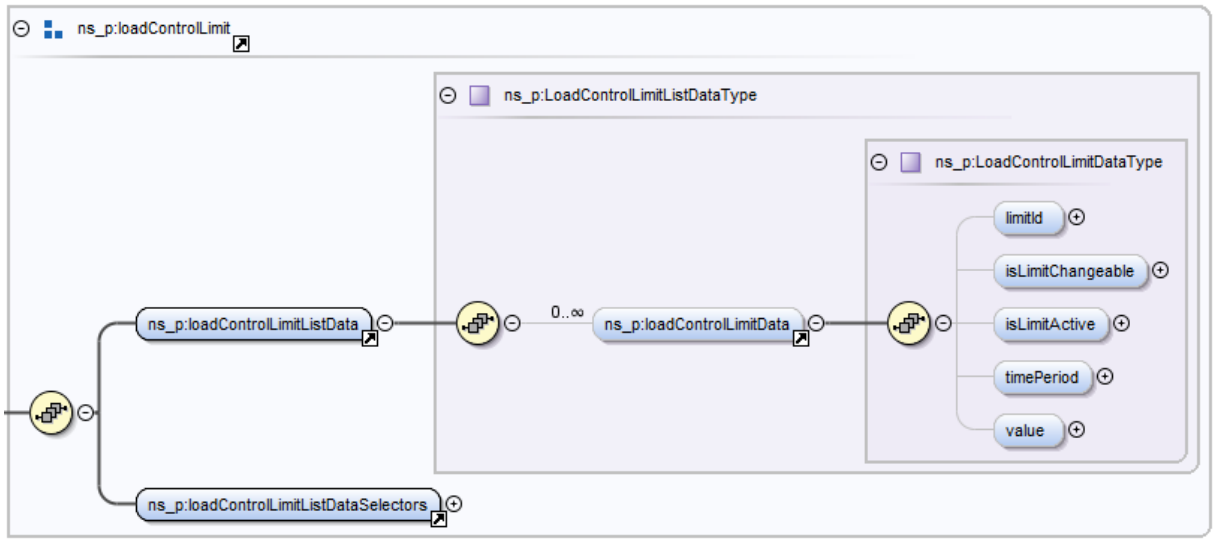


Figure 111: loadControlLimitListData function overview

5.3.14.5.2 Detailed description of elements

Element	Type	Description
limitId	*Identifier "LoadControlLimitIdType" (see* Table 339*).*	Identifier for the limit.
isLimitChangeable	xs:boolean (W3C standard type)	States whether the limit may be changed by a client or not.
isLimitActive	xs:boolean (W3C standard type)	Indicates whether the limit is currently active or not.
timePeriod	*Common data type "TimePeriodType". See section 3.10.2.1.*	The period where the limit shall be active.
value	*Common data type "ScaledNumberType". See section 3.10.1.8.*	The actual limit.

Table 225: loadControlLimitListData function detailed description of elements

5.3.14.5.3 Available selectors

- limitId

5.3.14.5.4 Examples

5.3.14.5.4.1 Read-reply

If one is interested in the current value of a specific limit, he could send the following read command:


```
<function>loadControlLimitListData</function>
<filter>
    <cmdControl>
        <partial/>
    </cmdControl>
    <loadControlLimitListDataSelectors>
        <limitId>2</limitId>
    </loadControlLimitListDataSelectors>
</filter>
<loadControlLimitListData/>
```

If the receiving device has a proper value for this *keyId*, it would respond with the following function:

```
<loadControlLimitListData>
    <loadControlLimitData>
        <limitId>2</limitId>
        <isLimitActive>true</isLimitActive>
        <isLimitChangeable>true</isLimitChangeable>
        <timePeriod>
            <startTime>-PT13M30S</startTime>
            <endTime>PT16M30S</endTime>
        </timePeriod>
        <value>
            <number>15</number>
            <scale>1</scale>
        </value>
    </loadControlLimitData>
</loadControlLimitListData>
```

5.3.14.5.4.2 Write (partial)

If an client wants to limit the consumption of an appliance to a certain value that shall begin in 15 minutes and end in 2 hours, it could send the following command to the load control resource:

```
<function>loadControlLimitListData</function>
<filter>
    <cmdControl>
        <partial/>
    </cmdControl>
</filter>
<loadControlLimitListData>
    <loadControlLimitData>
        <limitId>2</limitId>
        <timePeriod>
            <startTime>PT15M</startTime>
            <endTime>PT2H</endTime>
        </timePeriod>
        <value>
            <number>1</number>
            <scale>2</scale>
        </value>
    </loadControlLimitData>
</loadControlLimitListData>
```


5.3.14.6 loadControlLimitConstraintsListData

5.3.14.6.1 General

Constraints that shall be held when trying to change a limit, are modelled with the loadControlLimitConstraintsListData function, including a mininum value, a maximum value and the stepsize of the values between.

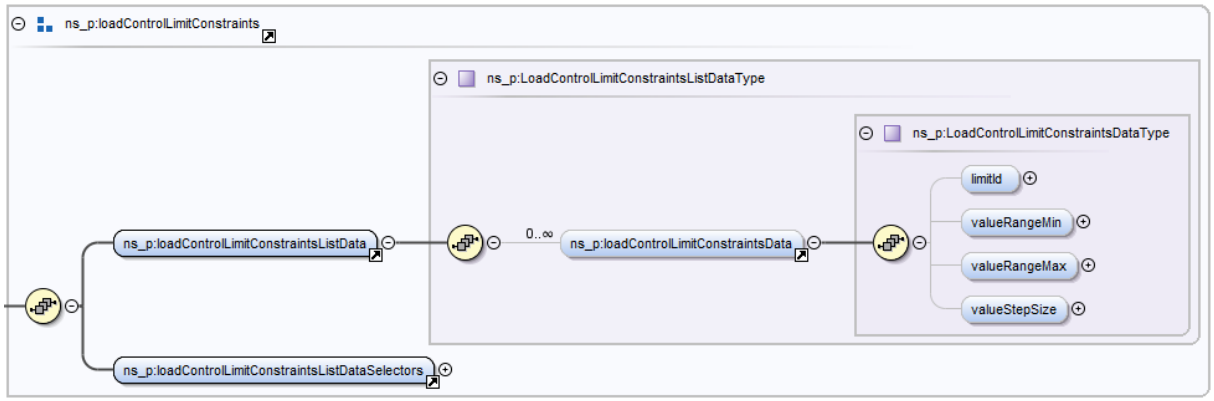


Figure 112: loadControlLimitConstraintsListData function overview

5.3.14.6.2 Detailed description of elements

Element	Type	Description
limitId	*Identifier "LoadControlLimitIdType" (see* Table 339*).*	Identifier for the limit.
valueRangeMin	*Common data type "ScaledNumberType". See section 3.10.1.8.*	This element allows to define a minimum limit for the element value within *loadControlLimitListData*.
valueRangeMax	*Common data type "ScaledNumberType". See section 3.10.1.8.*	This element allows to define a maximum limit for the element value within *loadControlLimitListData*.
valueStepSize	*Common data type "ScaledNumberType". See section 3.10.1.8.*	The minimum step size between two different limit values.

Table 226: loadControlLimitConstraintsListData function detailed description of elements

5.3.14.6.3 Available selectors

- limitId

5.3.14.6.4 Examples

5.3.14.6.4.1 Read-reply

If one is interested in the value constraints of a specific limit, he could send the following read command:

```
<function>loadControlLimitConstraintsListData</function>
<filter>
```



```
    <cmdControl>
        <partial/>
    </cmdControl>
    <loadControlLimitConstraintsListDataSelectors>
        <limitId>2</limitId>
    </loadControlLimitConstraintsListDataSelectors>
</filter>
<loadControlLimitConstraintsListData/>
```

If the receiving device has a proper value for this *limitId*, it would respond with the following function:

```
<loadControlLimitConstraintsListData>
    <loadControlLimitConstraintsData>
        <limitId>2</limitId>
        <valueRangeMin>
            <scaledNumber>
                <number>0</number>
            </scaledNumber>
        </valueRangeMin>
        <valueRangeMax>
            <scaledNumber>
                <number>36</number>
                <scale>2</scale>
            </scaledNumber>
        </valueRangeMax>
        <valueStepSize>
            <scaledNumber>
                <number>5</number>
                <scale>-1</scale>
            </scaledNumber>
        </valueStepSize>
    </loadControlLimitConstraintsData>
</loadControlLimitConstraintsListData>
```

5.3.14.7 loadControlLimitDescriptionListData

5.3.14.7.1 General

The rather static information about the available limit(s) are modelled within the loadControlLimitDescriptionListData function.

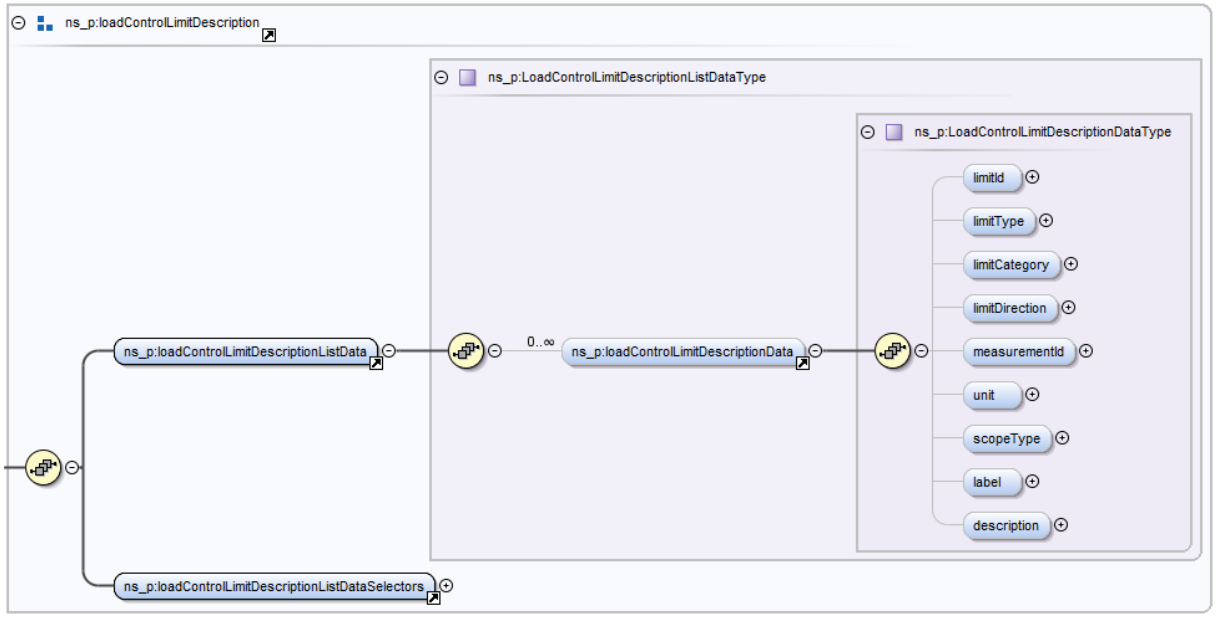


Figure 113: loadControlLimitDescriptionListData function overview

5.3.14.7.2 Detailed description of elements

Element	Type	Description
limitId	*Identifier "LoadControlLimitIdType" (see* Table 339*).*	Identifier for the limit.
limitType	*Union* "LoadControlLimitTypeType*":* *- Enum (see* Table 228*):* LoadControlLimitTypeEnumType - EnumExtendType (see section 3.10.1.5)	Whether the limit shall be used as a minumum or maximum is denoted within this element.
limitCategory	*Union* "LoadControlLimitCategoryType*":* *- Enum (see* Table 229*):* LoadControlLimitCategoryEnumType - EnumExtendType (see section 3.10.1.5)	There are different levels of how important the limit is (see enumeration description).
limitDirection	*Common data type "EnergyDirectionType". See section 3.10.1.13.*	If a device supports both, energy consumption and energy production, the direction which shall be limited is stated in this element (e.g. if only the consumption of a battery shall be limited, the enumeration "consume" will be set; if both, consumption and production shall be limited, two limits have to be set).
measurementId	*Identifier "MeasurementIdType" (see* Table 339*).*	If a measurand is linked to the limit, its identifier is denoted here.

unit	*Common data type "UnitOfMeasurementType". See section 3.10.1.17.*	The unit, which is used for the limit value, is denoted with this element.
scopeType	*Common data type "ScopeTypeType". See section 3.10.1.21.*	A certain meaning of the limit.
label	*Common data type "LabelType". See section 3.10.1.2.*	A label for the limit.
description	*Common data type "DescriptionType". See section 3.10.1.3.*	A description for the limit.

Table 227: loadControlLimitDescriptionListData function detailed description of elements

Enumeration **LoadControlLimitTypeEnumType**:

Value	Description
minValueLimit	The limit set in the "value" shouldn't be underrun (value > minValueLimit).
maxValueLimit	The limit set in the "value" shouldn't be overrun (value < maxValueLimit).

Table 228: Enumeration LoadControlLimitTypeEnumType

Enumeration **LoadControlLimitCategoryEnumType**:

Value	Description
obligation	Used for limits that are absolutely important to be hold. A loss of comfort could happen and will be accepted. The normal operation will be interrupted for keeping the limit (if needed).
recommendation	Used for limits that are quite important for the energy management. A loss of comfort could happen and will be accepted for a short period of time. The normal operation should not be interrupted.
optimization	Used for limits that just optimize the energy management. No loss of comfort will be accepted. The normal operation shall not be interrupted.

Table 229: Enumeration LoadControlLimitCategoryEnumType

5.3.14.7.3 Available selectors

- limitId
- limitType
- limitDirection
- measurementId
- scopeType

5.3.14.7.4 Examples

5.3.14.7.4.1 Read-reply

A *loadControlLimitDescriptionListData* instance sent as reply to a standard read command:

```
<loadControlLimitDescriptionListData>
    <loadControlLimitDescriptionData>
        <limitId>1</limitId>
        <limitType>maxValueLimit</limitType>
```



```
            <limitCategory>obligation</limitCategory>
            <limitDirection>consume</limitDirection>
            <measurementId>3</measurementId>
            <unit>W</unit>
        </loadControlLimitDescriptionData>
        <loadControlLimitDescriptionData>
            <limitId>2</limitId>
            <limitType>maxValueLimit</limitType>
            <limitCategory>obligation</limitCategory>
            <limitDirection>produce</limitDirection>
            <measurementId>5</measurementId>
            <unit>W</unit>
        </loadControlLimitDescriptionData>
</loadControlLimitDescriptionListData>
```

5.3.15 Measurement

5.3.15.1 Introduction

The SPINE Measurement class provides functionality for transmitting measured, calculated or empirically obtained values from sensors. Mainly these are sensors that provide some kind of information about the environment or physical parameters. Clients can use the information for further actions or just visualize them for the information of a user.

Due to the nature of measurement, the measured values are only readable. Some control systems permit a user to select a preferred value (e.g. a desired room temperature). Such “desired” values obviously require writable fields. This can be achieved with the "Setpoint" class (see section 5.3.21).

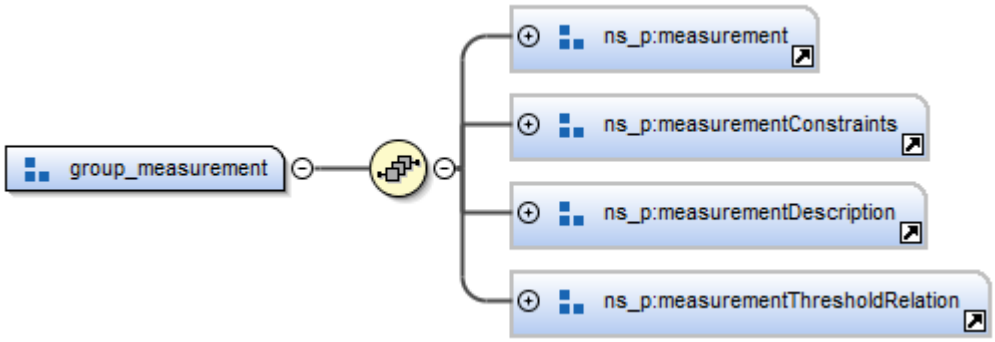
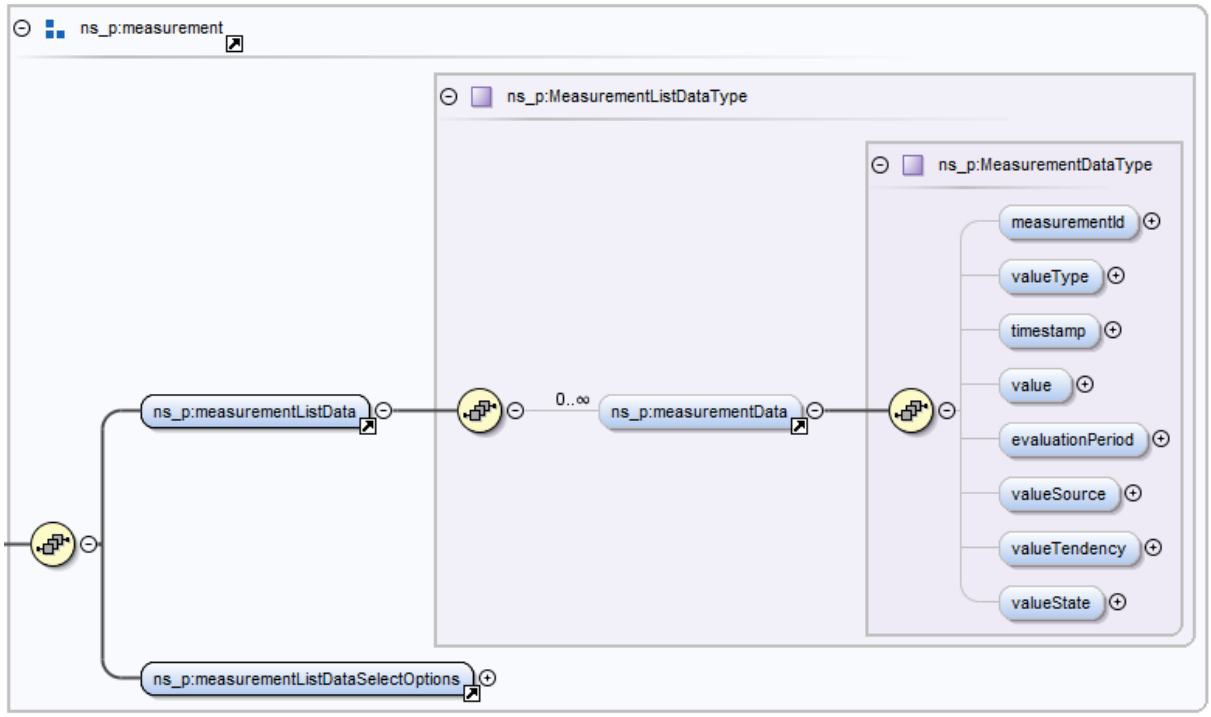


Figure 114: Measurement function-group overview

5.3.15.2 measurementListData

5.3.15.2.1 General

The *measurementListData* command is used to communicate measurement values. Typically, a sensor sends this command as notification as soon as it changes more than a specific threshold. Of course, it may be readable, too.



7585

7586 *Figure 115: measurementListData function overview*

7587

7588 *5.3.15.2.2 Detailed description of elements*

Element	Type	Description
measurementId	*Identifier "MeasurementIdType" (see* Table 339*).*	Enables the identification of different sensors on one SPINE feature.
valueType	*Union "*MeasurementValueTypeType*": - Enum (see* Table 231*):* MeasurementValueTypeEnumType - EnumExtendType (see section 3.10.1.5)	It is possible to model different types of measurement values. For example “value”, “averageValue”, “minValue”, etc. (see the SPINE data model for all possible value types).
timestamp	*Common data type "AbsoluteOrRelativeTimeType". See section 3.10.2.3.*	The time of creation of a measurement value element (i.e. the time it was measured, calculated, ...). Some sensors and communications technologies may already provide this information natively. In other cases the technology interface may set timestamp to the time of creation of a measurementData instance.
value	*Common data type "ScaledNumberType". See section 3.10.1.8.*	The measurement value itself, i.e. the magnitude according to “valueType”.
evaluationPeriod	*Common data type "TimePeriodType". See section 3.10.2.1.*	Some *valueType* values can be detailed or even require information about the time period. For example, an *averageValue* is

		usually computed from measured values of a finite time period. For smart devices, the start and end time of the *evaluationPeriod* can be filled in directly. For bare ("simple") sensors, such "derived"/"accompanying" values (average, min, max, ...) could be calculated (if desired) by the technology interface component together with the corresponding *evaluationPeriod*.
valueSource	*Union "*MeasurementValueSourceType*":* - *Enum (see* Table 232*):* MeasurementValueSourceEnumType - EnumExtendType (see section 3.10.1.5)	Whether element "value" is measured, calculated or empirically obtained can be denoted with this element.
valueTendency	*Union "*MeasurementValueTendencyType*":* - *Enum (see* Table 233*):* MeasurementValueTendencyEnumType - EnumExtendType (see section 3.10.1.5)	This elements states whether the tendency is *rising*, *stable* or *falling*.
valueState	*Union "*MeasurementValueStateType*":* - *Enum (see* Table 234*):* MeasurementValueStateEnumType - EnumExtendType (see section 3.10.1.5)	A measurand may have a state.

Table 230: measurementListData function detailed description of elements

Enumeration **MeasurementValueTypeEnumType**:

Value	Description
value	The normal measurement value.
averageValue	The average measurement value over the last time, stated in the element evaluationPeriod.
minValue	The minimum measurement value of the last time, stated in the element evaluationPeriod.
maxValue	The maximum measurement value of the last time, stated in the element evaluationPeriod.
standardDeviation	Used to quantify the amount of variation or dispersion of a set of values.

Table 231: Enumeration MeasurementValueTypeEnumType

Enumeration **MeasurementValueSourceEnumType**:

Value	Description
measuredValue	The measurement value was measured with a dedicated sensor. The value is rather exact.
calculatedValue	The measurement value was calculated with the values from some other parameters (e.g. water temperature, outside temperature, etc.) that have a calculable impact on the measurand. This value is not as exact as a measured value, but it differs not too much from the real value.

empiricalValue	The measurement value is only an empirical one that can differ a lot from the real value. It can rely on some other parameters that have a known value, but there is no exactly calculable relation between them.

Table 232: Enumeration MeasurementValueSourceEnumType

Enumeration **MeasurementValueTendencyEnumType**:

Value	Description
rising	The average measurement value is currently rising.
stable	The average measurement value is currently stable.
falling	The average measurement value is currently falling.

Table 233: Enumeration MeasurementValueTendencyEnumType

Enumeration **MeasurementValueStateEnumType**:

Value	Description
normal	The measurement value could be measured / calculated / empirically determined as expected.
outOfRange	The measurement value is currently out of range of the sensor (as defined by “valueRangeMin” and “valueRangeMax”, see section 5.3.15.3.2).
error	An error occurred while determining the measurement value.

Table 234: Enumeration MeasurementValueStateEnumType

5.3.15.2.3 Available selectors

- measurementId
- valueType
- timestampInterval

5.3.15.2.4 Examples

5.3.15.2.4.1 Read-reply

A *measurementListData* command with a measured value, sent as reply with only one entry (device is not storing history values):

```
<measurementListData>
    <measurementData>
        <measurementId>1</measurementId>
        <valueType>value</valueType>
        <timestamp>2013-07-15T12:00:00.0Z</timestamp>
        <value>
            <number>255</number>
            <scale>-1</scale>
        </value>
        <valueSource>measuredValue</valueSource>
        <valueTendency>rising</valueTendency>
        <valueState>normal</valueState>
    </measurementData>
</measurementListData>
```


5.3.15.2.4.2 Notify

A *measurementListData* command with a calculated average value (no other value type supported by the device), sent as (complete) notification:

```
<measurementListData>
    <measurementData>
        <measurementId>1</measurementId>
        <valueType>averageValue</valueType>
        <timestamp>2013-07-15T12:13:14.5Z</timestamp>
        <value>
            <number>23</number>
        </value>
        <evaluationPeriod>
            <startTime>2013-07-15T11:00:00.0Z</startTime>
            <endTime>2013-07-15T12:00:00.0Z</endTime>
        </evaluationPeriod>
        <valueSource>calculatedValue</valueSource>
        <valueTendency>stable</valueTendency>
        <valueState>normal</valueState>
    </measurementData>
</measurementData>
```

5.3.15.2.4.3 Read-reply

In this example the requested node supports only values of type "value" and offers this reply:

```
<measurementListData>
    <measurementData>
        <measurementId>1</measurementId>
        <valueType>value</valueType>
        <timestamp>2013-07-15T12:10:00.0Z</timestamp>
        <value>
            <number>257</number>
            <scale>-1</scale>
        </value>
        <valueSource>measuredValue</valueSource>
        <valueTendency>falling</valueTendency>
        <valueState>normal</valueState>
    </measurementData>
    <measurementData>
        <measurementId>1</measurementId>
        <valueType>value</valueType>
        <timestamp>2013-07-15T12:20:00.0Z</timestamp>
        <value>
            <number>261</number>
            <scale>-1</scale>
        </value>
        <valueSource>measuredValue</valueSource>
        <valueTendency>stable</valueTendency>
        <valueState>normal</valueState>
    </measurementData>
</measurementListData>
```

5.3.15.3 measurementConstraintsListData

5.3.15.3.1 General

Some constraints concerning the possible values of a measurand are modelled within this function.

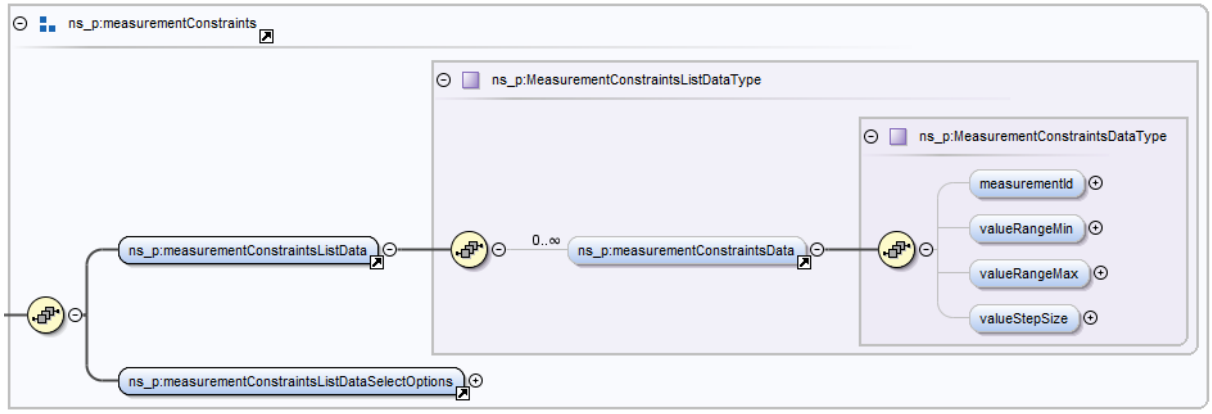


Figure 116: measurementConstraintsListData function overview

5.3.15.3.2 Detailed description of elements

Element	Type	Description
measurementId	*Identifier "MeasurementIdType" (see* Table 339*).*	Enables the identification of different sensors on one SPINE feature.
valueRangeMin	*Common data type "ScaledNumberType". See section 3.10.1.8.*	The minimum possible measurement value measurable by the feature.
valueRangeMax	*Common data type "ScaledNumberType". See section 3.10.1.8.*	The maximum possible measurement value measurable by the feature.
valueStepSize	*Common data type "ScaledNumberType". See section 3.10.1.8.*	The minimum step size between two different measurement values.

Table 235: measurementConstraintsListData function detailed description of elements

5.3.15.3.3 Available selectors

- measurementId

5.3.15.3.4 Examples

5.3.15.3.4.1 Read-reply

A *measurementConstraintsListData* instance from a temperature sensor, sent as reply to a standard read command:

```
<measurementConstraintsListData>
    <measurementConstraintsData>
        <measurementId>1</measurementId>
        <valueRangeMin>
            <number>-1</number>
            <scale>1</scale>
        </valueRangeMin>
        <valueRangeMax>
            <number>4</number>
            <scale>1</scale>
```



```
            </valueRangeMax>
            <valueStepSize>
                <number>1</number>
                <scale>-1</scale>
            </valueStepSize>
        </measurementConstraintsData>
</measurementConstraintsListData>
```

5.3.15.4 measurementDescriptionListData

5.3.15.4.1 General

Descriptive information on a specific measurement type is modelled with a *measurementDescriptionListData* function. This information is rather constant in comparison to measurement values. Typically, it is requested once at the beginning, when a node (an external display, e.g.) is configured to present or use a specific measurement value. As usual, for such scenarios it is recommended the the server notifies changes to all relevant communication partners, because a server cannot expect clients to periodically request this information.

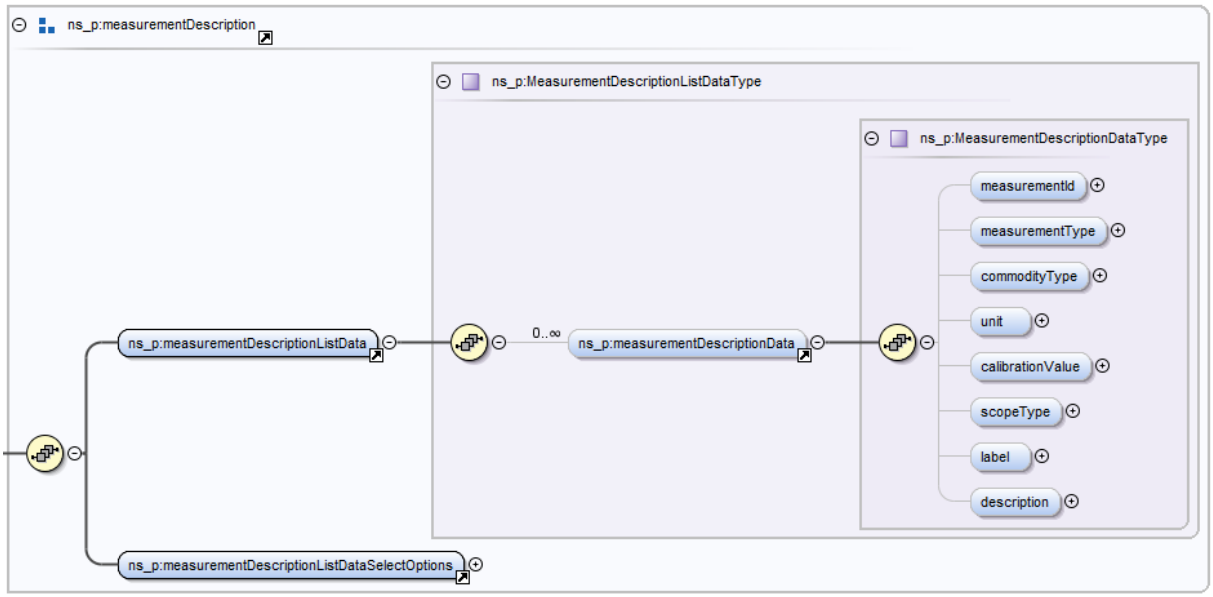


Figure 117: measurementDescriptionListData function overview

5.3.15.4.2 Detailed description of elements

Element	Type	Description
measurementId	*Identifier "MeasurementIdType" (see* Table 339*).*	Enables the identification of different measurands on one SPINE feature.
measurementType	*Union* "MeasurementTypeType"*:* - *Enum (see* Table 237*):* MeasurementTypeEnumType - EnumExtendType (see section 3.10.1.5)	To identify which type of measurand is measured (for example “temperature”).

commodityType	*Common data type "CommodityTypeType". See section 3.10.1.9.*	If a measurand of a commodity is measured, the type of commodity is stated here.
unit	*Common data type "UnitOfMeasurementType". See section 3.10.1.17.*	The unit, which is used for the value. It is always related to the normal value. Other *valueType* values (e.g. *averageDerivativeValue*) can have different units (derived from *unit*).
calibrationValue	*Common data type "ScaledNumberType". See section 3.10.1.8.*	Some measurement sensors need a calibration value (e.g. set during production phase). It is automatically added to the internal measured value. I.e. the value stated in the *value* element in the *measurementListData* function is the already corrected value.
scopeType	*Common data type "ScopeTypeType". See section 3.10.1.21.*	Specifies a more detailed meaning of the measurand (e.g. outsideTemperature).
label	*Common data type "LabelType". See section 3.10.1.2.*	A (short) label can be useful to identify or group specific sensors.
description	*Common data type "DescriptionType". See section 3.10.1.3.*	A (longer) description of a measurement device can be deposited here.

7719 *Table 236: measurementDescriptionListData function detailed description of elements*

7720 Enumeration **MeasurementTypeEnumType**:

Value	**Description**
acceleration	The acceleration is measured.
angle	An angle is measured.
angularVelocity	The angular velocity is measured.
area	An area is measured.
atmosphericPressure	The atmospheric pressure is measured.
capacity	The capacity is measured.
concentration	A concentration of a substance is measured.
count	A count is measured.
current	The current is measured.
density	The density is measured.
distance	A distance is measured.
electricField	The electric field is measured.
energy	The energy is measured.
force	The force is measured.
frequency	The frequency is measured.
harmonicDistortion	The harmonic distortion is measured.
heat	The heat is measured.
heatFlux	The heat flux is measured.
illuminance	The illuminance is measured.
impulse	An impulse is measured.
level	A level is measured.
magneticField	The magnetic field is measured.
mass	A mass is measured.

massFlow	The mass flow is measured.
particles	The quantity of particles is measured.
percentage	The percentage is measured.
power	The power is measured.
powerFactor	The power factor is measured.
pressure	The pressure is measured.
radonActivity	The radon activity is measured.
relativeHumidity	The relative humidity is measured.
resistance	The resistance is measured.
solarRadiation	The solar radiation is measured.
speed	The speed is measured.
temperature	The temperature is measured.
time	The time is measured.
torque	The torque is measured.
unknown	An unknown measurand is measured.
velocity	The velocity is measured.
voltage	The voltage is measured.
volume	The volume is measured.
volumetricFlow	The volumetric flow is measured.

Table 237: Enumeration MeasurementTypeEnumType

5.3.15.4.3 Available selectors

- measurementId
- measurementType
- commodityType
- scopeType

5.3.15.4.4 Examples

5.3.15.4.4.1 Read-reply

A *measurementDescriptionListData* instance from a temperature sensor, sent as reply to a standard read command:

```
<measurementDescriptionListData>
    <measurementDescriptionData>
        <measurementId>1</measurementId>
        <measurementType>temperature</measurementType>
        <commodityType>air</commodityType>
        <unit>degC</unit>
        <calibrationValue>
            <number>-48</number>
            <scale>-1</scale>
        </calibrationValue>
        <scopeType>outsideTemperature</scopeType>
        <label>Outside_Temp.</label>
        <description>Outside Air Temperature</description>
    </measurementDescriptionData>
</measurementDescriptionListData>
```


5.3.15.5 measurementThresholdRelationListData

5.3.15.5.1 General

The relation between a measurand and dedicated thresholds (see section 5.3.26) is modelled with this function.

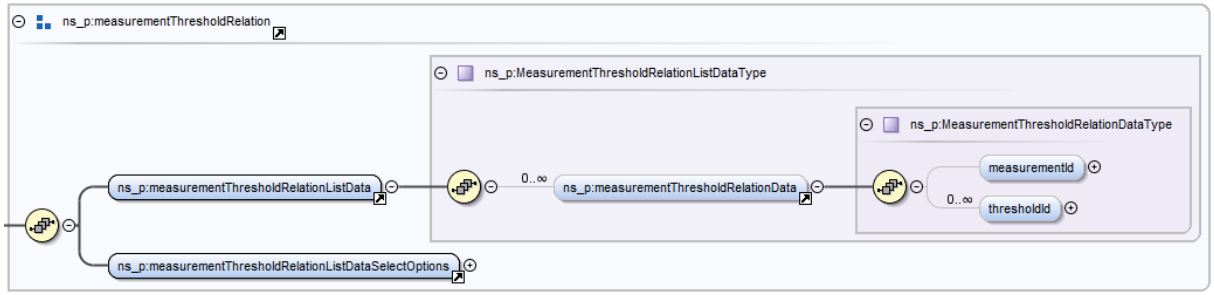


Figure 118: measurementThresholdRelationListData function overview

5.3.15.5.2 Detailed description of elements

Element	Type	Description
measurementId	*Identifier "MeasurementIdType" (see* Table 339*).*	Enables the identification of different sensors on one SPINE address.
thresholdId (list)	*Identifier "ThresholdIdType" (see* Table 339*).*	Related threshold identifier(s).

Table 238: measurementThresholdRelationListData function detailed description of elements

5.3.15.5.3 Available selectors

- measurementId
- thresholdId

5.3.15.5.4 Examples

5.3.15.5.4.1 Read-reply

A *measurementThresholdRelationListData* sent as a reply to a standard read command:

```
<measurementThresholdRelationListData>
    <measurementThresholdRelationData>
        <measurementId>1</measurementId>
        <thresholdId>2<thresholdId>
        <thresholdId>3<thresholdId>
        <thresholdId>6<thresholdId>
    </measurementThresholdRelationData>
</measurementThresholdRelationListData>
```


5.3.16 Messaging

5.3.16.1 Introduction

Messaging is basically intended for simple textual information transfer for displays and users. The textual information is not intended to be used for any automatisms. It should be used for textual information that are for example event based or otherwise will change over time.

Some devices may keep a list of messages rather than (or in addition to) a single message.

Figure 119: Messaging function-group overview

5.3.16.2 messagingListData

5.3.16.2.1 General

The messagingListData function contains the neccessary elements to model a simple text message.

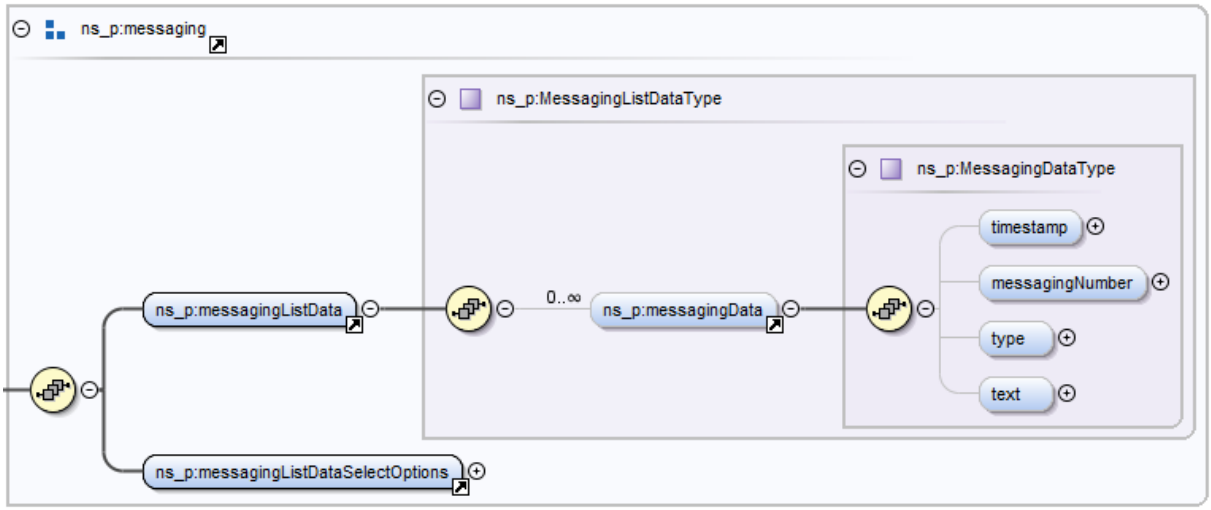


Figure 120: messagingListData function overview

5.3.16.2.2 Detailed description of elements

Element	Type	Description
timestamp	*Common data type "AbsoluteOrRelativeTimeType". See section 3.10.2.3.*	The time when the message was generated.
messagingNumber	*Identifier "MessagingNumberType" (see* Table 339*).*	An identifier for one specific message. If a message is marked as obsolete (see below) this number can be used to identify the original message.
type	*Union "*MessagingTypeType*":* *- Enum (see* Table 240*):* MessagingTypeEnumType	The type of the message.

	- EnumExtendType (see section 3.10.1.5)	
text	*Simple type "MessagingDataTextType" (restriction of xs:string)*	The message itself.

Table 239: messagingListData function detailed description of elements

Enumeration **MessagingTypeEnumType**:

Value	Description
logging	Used for messages that are stored in a log file.
information	Messages that are presented to the customer on a display; lower priority.
warning	Messages that are presented to the customer on a display; medium priority.
alarm	Messages that are presented to the customer on a display; high priority.
emergency	Messages that are presented to the customer on a display; very high priority.
obsolete	Used to mark previously sent messages as obsolete (so that they are not displayed anymore).

Table 240: Enumeration MessagingTypeEnumType

5.3.16.2.3 Available selectors

- timestampInterval
- messagingNumber

5.3.16.2.4 Examples

5.3.16.2.4.1 Notify

A washing machine is configured to send messages to an external display. After a firmware update it sends this (complete) notification:

```
<messagingListData>
    <messagingData>
        <timestamp>2006-05-04T18:13:51.0Z</timestamp>
        <messagingNumber>72</messagingNumber>
        <type>information</type>
        <text>Update completed.</text>
    </messagingData>
</messagingListData>
```

Other possible reasons for messages to a display: Notification of end of washing cycle; notification of problem with water supply; etc.

5.3.16.2.4.2 Notify

An alarm device keeps a list of its events of the last month. It is configured to send new messages to an external display every 2 minutes (the “filter/partial” part to announce the “restricted function exchange is simplified by “…” just to improve the readability”):

```
...
<messagingListData>
    <messagingData>
```



```
            <timestamp>2006-05-04T18:13:51.0Z</timestamp>
            <messagingNumber>2014</messagingNumber>
            <type>information</type>
            <text>Window 1st floor closed.</text>
        </messagingData>
        <messagingData>
            <timestamp>2006-05-04T18:14:23.0Z</timestamp>
            <messagingNumber>2015</messagingNumber>
            <type>information</type>
            <text>Alarm engaged.</text>
        </messagingData>
    </messagingListData>
```

5.3.16.2.4.3 Read-reply

The alarm device is able to provide its complete message list upon request.

A user selects the alarm device on an external display and queries its messages. The alarm device replies with a proper messagingListData. Extract with just the first and the last message:

```
<messagingListData>
    <messagingData>
        <timestamp>2006-04-05T03:23:17.0Z</timestamp>
        <messagingNumber>1716</messagingNumber>
        <type>alarm</type>
        <text>Window ground floor opened.</text>
    </messagingData>
    ...
    <messagingData>
        <timestamp>2006-05-04T18:14:23.0Z</timestamp>
        <messagingNumber>2015</messagingNumber>
        <type>information</type>
        <text>Alarm engaged.</text>
    </messagingData>
</messagingListData>
```

5.3.16.2.4.4 Read-reply

The alarm device supports the selector *timestampInterval*. A user selects the alarm device on an external display and queries its messages of the last hour. The display sends a read classifier together with the corresponding filter:

```
<function>messagingListData</function>
<filter>
    <cmdControl>
        <partial/>
    </cmdControl>
    <messagingListDataSelectors>
        <timestampInterval>
            <startTime>-PT1H</startTime>
            <endTime>PT0S</endTime>
        </timestampInterval>
    </messagingListDataSelectors>
</filter>
<messagingListData/>
```


The alarm device replies with a proper messagingListData. Extract with just the first and the last message:

```
<function>messagingListData</function>
<filter>
    <cmdControl>
        <partial/>
    </cmdControl>
</filter>
<messagingListData>
    <messagingData>
        <timestamp>2006-05-04T17:55:19.0Z</timestamp>
        <messagingNumber>2012</messagingNumber>
        <type>information</type>
        <text>Front door closed.</text>
    </messagingData>
    ...
    <messagingData>
        <timestamp>2006-05-04T18:14:23.0Z</timestamp>
        <messagingNumber>2015</messagingNumber>
        <type>information</type>
        <text>Alarm engaged.</text>
    </messagingData>
</messagingListData>
```

5.3.16.2.4.5 Read-reply

The alarm device supports the select option *messagingNumber*. A user selects the alarm device on an external display and queries the messages with messagingNumber "2011". The display sends a read classifier together with the select options:

```
<function>messagingListData</function>
<filter>
    <cmdControl>
        <partial/>
    </cmdControl>
    <messagingListDataSelectors>
        <messagingNumber>2011</messagingNumber>
    </messagingListDataSelectors>
</filter>
<messagingListData/>
```

The alarm device replies with a proper messagingListData:

```
<function>messagingListData</function>
<filter>
    <cmdControl>
        <partial/>
    </cmdControl>
</filter>
<messagingListData>
    <messagingData>
        <timestamp>2006-05-04T16:10:12.0Z</timestamp>
        <messagingNumber>2011</messagingNumber>
        <type>information</type>
        <text>Alarm disengaged.</text>
    </messagingData>
</messagingListData>
```


5.3.17 NetworkManagement

5.3.17.1 Introduction

This class permits to model certain "knowledge" of other devices as well as the "modification" of this knowledge. In terms of a communications technology these models can be used to add remote nodes (devices, endpoints) to the own (SPINE) network, to remove them from the network, etc.

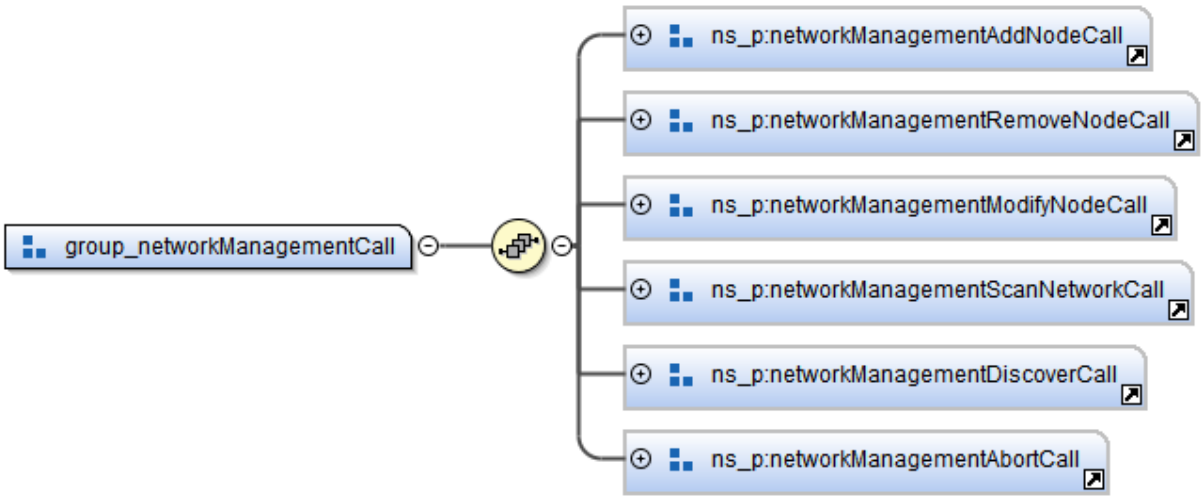


Figure 121: NetworkManagement function-group overview (call)

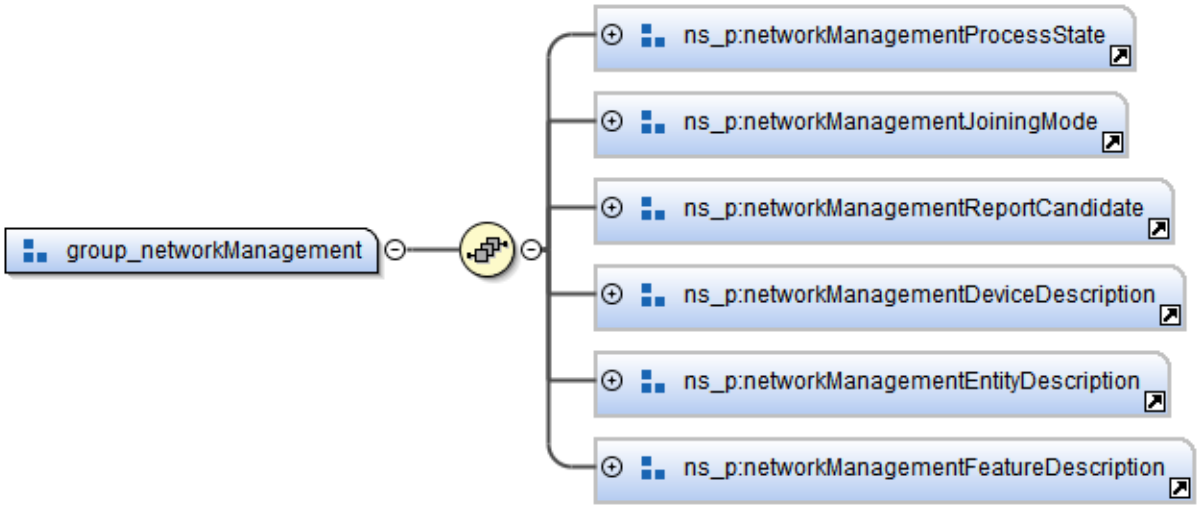


Figure 122: NetworkManagement function-group overview (data)

5.3.17.1.1 SPINE address components, hierarchy, terminology

A SPINE address can contain these address elements:

1. device
2. entity
3. feature

The element “entity” can appear multiple times (i.e. as list). This is used to model an entity hierarchy, starting with a root entity as first item of the list. The consecutive list of “entity” elements finally denotes a specific entity.

Altogether, the address elements are used to determine the proper functional component. Each device can contain several entities. Each entity can contain several features. Each addressable component is in general called a “node”.

5.3.17.1.2 Network Management Responsible

A so-called "Network Management Responsible" is defined in this class. This is an instance responsible to perform SPINE network management tasks for other nodes. The Network Management Responsible itself is considered a node with node address as well. A brief example explains this: A sophisticated "SPINE gateway" permits software applications to exchange SPINE XMLs with an (internal) interface that provides access to devices of different communications technologies. For each communications technology there is an (internal) implementation that realizes the translation between SPINE XML (from/to the aforementioned internal interface) and the dedicated communications technology interface/protocol (i.e. it makes use of the so-called "mapping"). We assume this "mapping implementation" is responsible to "find" devices accessible via its communications technology. In general, the implementation is responsible to control whether data can be exchanged between software applications and devices or not. This means these implementations are responsible for network management activities concerning their attached or accessible devices.

Please note this example is just to explain the term "Network Management Responsible". The architecture sketched in the example shall not be considered to be the only permitted architecture.

5.3.17.1.3 Use of “energy management gateway” in examples

Some communications technologies follow a “centralised” approach, whereas others are “de-centralised”. The class “NetworkManagement” does not impose which approach has to be followed. In the interest of simplification, subsequent examples often assume an “energy management gateway” which is capable to connect with other SPINE devices. However, please be aware that these examples are neither a mandatory feature of an “energy management gateway”, nor will a communications architecture be derived from it within this class description.

5.3.17.2 networkManagement...Call

The *networkManagement...Calls* are used to add, modify or remove nodes (usually on Gateways) or to tell a node, that it shall perform a network scan, discover a specific node or abort the current task.

5.3.17.3 networkManagementAddNodeCall

5.3.17.3.1 General

This function can be used to add another node (device) to the own SPINE network.

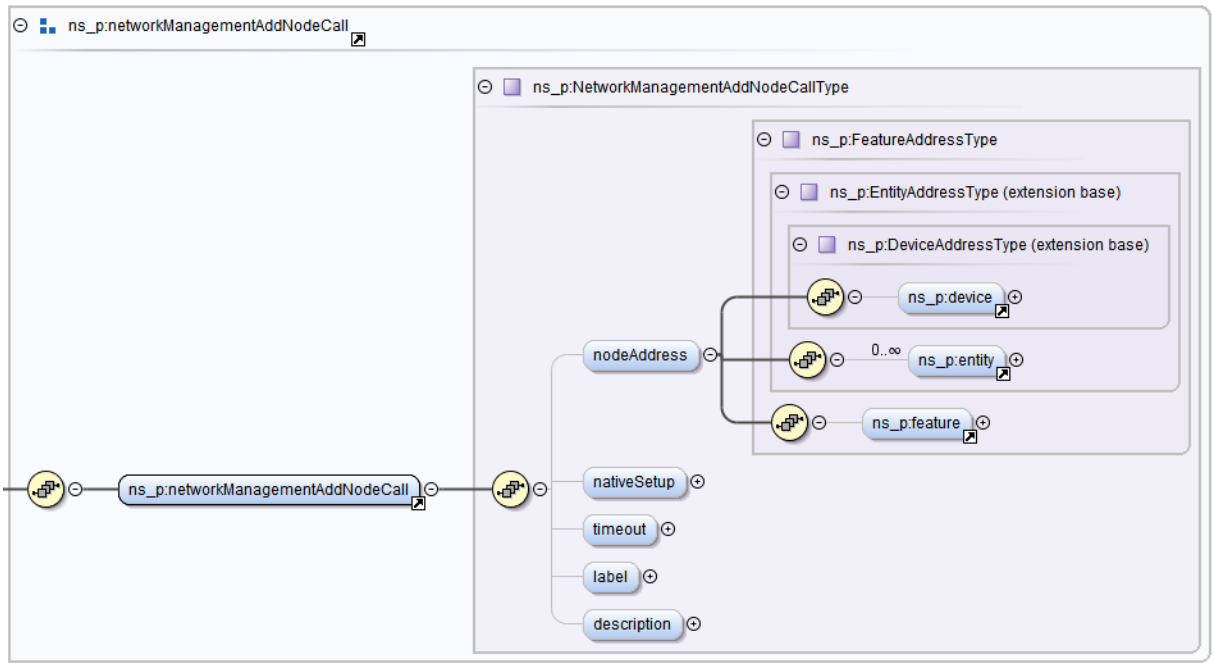


Figure 123: networkManagementAddNodeCall function overview

5.3.17.3.2 Detailed description of elements

Element	Type	Description
nodeAddress	*Common data type "FeatureAddressType". See section 3.10.3.6.*	The node address of the node that shall be added. Contains elements for *device*, *entity* and *feature*.
nodeAddress.device	*See section 3.10.3.6.*	Device address part.
nodeAddress.entity (list)	*See section 3.10.3.6.*	Entity (list) address part.
nodeAddress.feature	*See section 3.10.3.6.*	Feature address part.
nativeSetup	*Simple type "NetworkManagementNativeSetupType" (restriction of xs:string)*	Technology dependent or implementation dependent configuration to identify (and maybe configure) the new node.
timeout	xs:duration (W3C standard type)	The procedure of adding the node shall be aborted latest after this duration.
label	*Common data type "LabelType". See section 3.10.1.2.*	A short label of the new node.
description	*Common data type "DescriptionType". See section 3.10.1.3.*	Descriptive information on the new node.

Table 241: networkManagementAddNodeCall function element description

5.3.17.3.3 Available selectors

None.

5.3.17.3.4 Examples

5.3.17.3.4.1 Call

The user interface of an energy management gateway permits adding a display that is accessible via a specific communications technology denoted here as "CP". In this example the native address of the display in "CP" is known in advance ("21.6.5.3.11", e.g.). Furthermore it is assumed that "CP" is a rather low-level protocol and the energy management gateway's interface to "CP" has to be configured in order to communicate with the display properly (configuration of profiles, e.g.). There is information available how to translate this configuration into "nativeSetup" properly. We assume the vendor of the energy management gateway provided a sophisticated user interface that simplifies this procedure for the user. Thus, the user just selects the technology interface to "CP", enters the native address and the label and then selects the right profile for a display for the interface to "CP". The user interface creates the subsequent XML and provides it to the Network Management Responsible of "CP":

```
<networkManagementAddNodeCall>
    <nativeSetup>21.6.5.3.11;R-2.7;at-42.344;pqtl-
276381626831726378123568l3</nativeSetup>
    <timeout>PT40S</timeout>
    <label>Display</label>
</networkManagementAddNodeCall>
```

In this example the *nodeAddress* is unset to let the Network Management Responsible of “CP” assign a proper address. Upon the successful completion of the process the communications technology interface provides a proper *networkManagementProcessStateData* notification to the user interface. It furthermore creates proper *networkManagementDeviceDescriptionListData*, *networkManagementEntityDescriptionListData* and *networkManagementFeatureDescriptionListData* notifications for internal purposes (information for further applications installed on the energy management gateway: notification of a new node).

5.3.17.4 networkManagementRemoveNodeCall

5.3.17.4.1 General

This function can be used to remove a node from the own SPINE network.

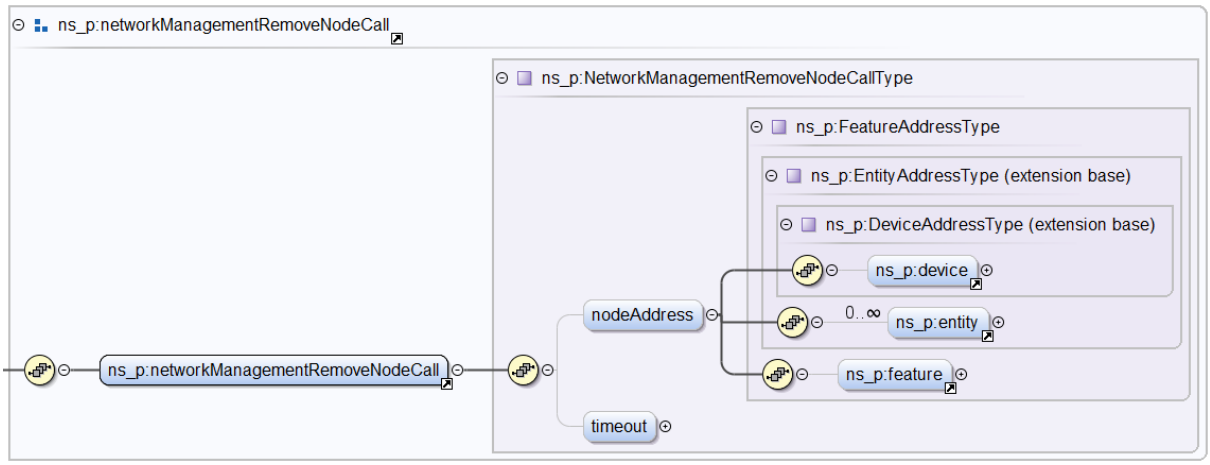


Figure 124: networkManagementRemoveNodeCall function overview

8020

8021 *5.3.17.4.2 Detailed description of elements*

Element	Type	Description
nodeAddress	*Common data type "FeatureAddressType". See section 3.10.3.6.*	The node address of the node that will be removed. Contains elements for *device*, *entity* and *feature*.
nodeAddress.device	*See section 3.10.3.6.*	Device address part.
nodeAddress.entity (list)	*See section 3.10.3.6.*	Entity (list) address part.
nodeAddress.feature	*See section 3.10.3.6.*	Feature address part.
timeout	xs:duration (W3C standard type)	The procedure of removing the node will be aborted latest after this duration.

8022 *Table 242: networkManagementRemoveNodeCall function element description*

8023

8024 *5.3.17.4.3 Available selectors*

8025 None.

8026

8027 *5.3.17.4.4 Examples*

8028 *5.3.17.4.4.1 Call*

8029 We assume a display belongs to the SPINE network of an energy management gateway and is
8030 assigned the device address "d:_i:46925_CP_pqtl-27638162683172637812356813" (as a result of
8031 the example in 5.3.17.3.4, e.g.). The user interface of the energy management gateway permits
8032 removing the display from the gateway's network. The user selects the display on the user interface
8033 and requests its removal. The user interface creates the subsequent XML and provides it to the
8034 Network Management Responsible of this node:

```
8035 <networkManagementRemoveNodeCall>
8036     <nodeAddress>
8037         <device>d:_i:46925_CP\_pqtl-27638162683172637812356813</device>
8038     </nodeAddress>
8039     <timeout>PT30S</timeout>
8040 </networkManagementRemoveNodeCall>
```

8041 Because the whole *device* will be removed (and not a specific *entity* or *feature*) only the *device*-part
8042 of the address is to be filled.

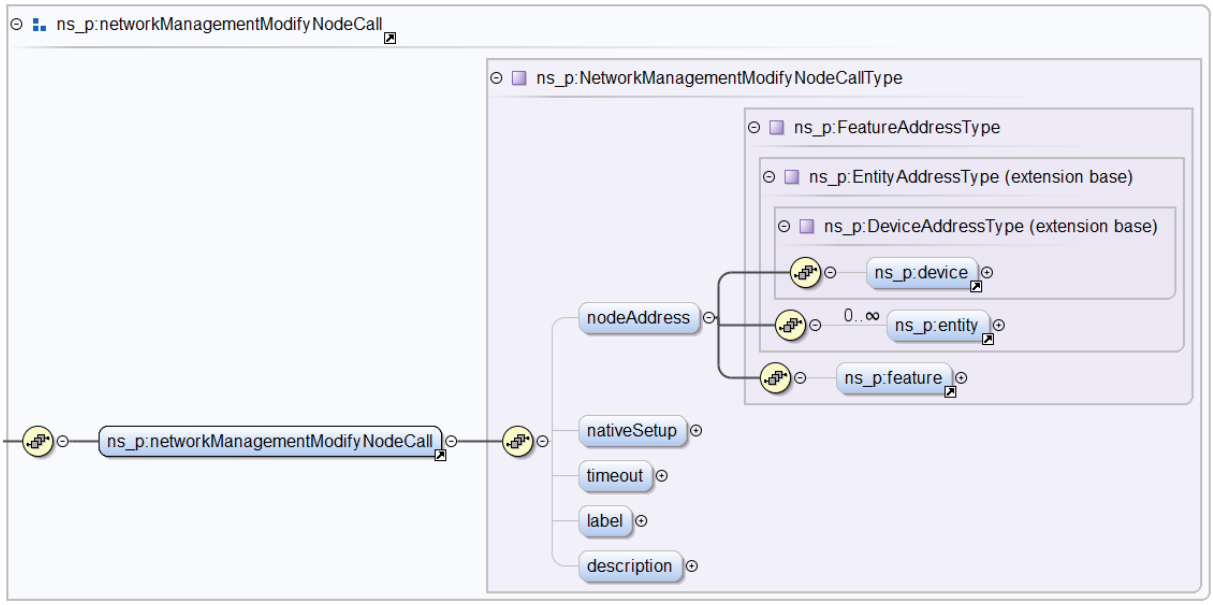
8043 Upon the successful completion of the process the Network Management Responsible provides a
8044 proper *networkManagementProcessStateData* notification to the user interface. It furthermore
8045 creates a proper *networkManagementDeviceDescriptionListData* notification for internal purposes
8046 (information for further applications installed on the energy management gateway: notification of a
8047 removed node).

8048

8049 ***5.3.17.5 networkManagementModifyNodeCall***

8050 *5.3.17.5.1 General*

8051 This function can be used to modify an already "present" node of the own SPINE network. Here,
8052 "present" implies the node has a node address.



8053

8054 *Figure 125: networkManagementModifyNodeCall function overview*

8055

8056 *5.3.17.5.2 Detailed description of elements*

Element	Type	Description
nodeAddress	*Common data type "FeatureAddressType". See section 3.10.3.6.*	The node address of the node that will be modified. Contains elements for *device*, *entity* and *feature*.
nodeAddress.device	*See section 3.10.3.6.*	Device address part.
nodeAddress.entity (list)	*See section 3.10.3.6.*	Entity (list) address part.
nodeAddress.feature	*See section 3.10.3.6.*	Feature address part.
nativeSetup	*Simple type "NetworkManagementNativeSetupType" (restriction of xs:string)*	Similar to *nativeSetup* of *networkManagementAddNodeCall*. But for *networkManagementModifyNodeCall* this element is not used to identify which node has to be modified (see element *nodeAddress* for this purpose).
timeout	xs:duration (W3C standard type)	The procedure of modifying the node will be aborted latest after this duration.
label	*Common data type "LabelType". See section 3.10.1.2.*	A short label of the node.

description	*Common data type "DescriptionType". See section 3.10.1.3.*	Descriptive information on the node.

Table 243: networkManagementModifyNodeCall function element description

5.3.17.5.3 Available selectors

None.

5.3.17.5.4 Examples

5.3.17.5.4.1 Call

We assume a display belongs to the SPINE network of an energy management gateway and is assigned the device address "d:_i:46925_CP_pqtl-27638162683172637812356813" (as a result of the example in 5.3.17.3.4, e.g.). The user interface of the energy management gateway permits modifying the display's label. The user selects the display on the user interface and enters the new label "Kitchen display". The user interface creates the subsequent XML and provides it to its own interface to the Network Management Responsible of this node:

```
<networkManagementModifyNodeCall>
    <nodeAddress>
        <device>d:_i:46925_CP\_pqtl-27638162683172637812356813</device>
    </nodeAddress>
    <nativeSetup>21.6.5.3.11;R-2.7;at-42.344;pqtl-
27638162683172637812356813</nativeSetup>
    <timeout>PT15S</timeout>
    <label>Kitchen display</label>
    <description></description>
</networkManagementModifyNodeCall>
```

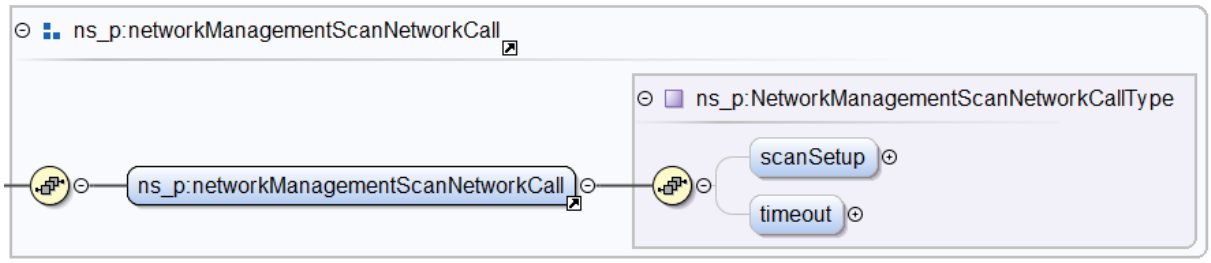
Because the label of the physical device will be changed (and not of a specific *entity* or *feature*) only the *device*-part of the address is to be filled.

Upon the successful completion of the process the Network Management Responsible provides a proper *networkManagementProcessStateData* notification to the user interface. It furthermore creates a proper *networkManagementDeviceDescriptionListData* notification for internal purposes (information for further applications installed on the energy management gateway: notification of a modified node).

5.3.17.6 networkManagementScanNetworkCall

5.3.17.6.1 General

Some communications technologies provide measures to scan which other devices are accessible. The function *networkManagementScanNetworkCall* can be used to trigger such an action, if supported by technology and implementation. This function is not intended to add discovered devices directly to the own SPINE network. Instead, it is intended just to get a report. This report can be modelled by networkManagementReportCandidateData. Of course, these reports may then be used to add discovered nodes.



8096

8097 *Figure 126: networkManagementScanNetworkCall function overview*

8098

8099 *5.3.17.6.2 Detailed description of elements*

Element	Type	Description
scanSetup	*Simple type "NetworkManagementScanSetupType" (restriction of xs:string)*	Technology dependent or implementation dependent configuration for the scanning procedure. This might contain parameters to search for specific types of devices, e.g.
timeout	xs:duration (W3C standard type)	The procedure of scanning will be aborted latest after this duration.

8100 *Table 244: networkManagementScanNetworkCall function element description*

8101

8102 *5.3.17.6.3 Available selectors*

8103 None.

8104

8105 *5.3.17.6.4 Examples*

8106 *5.3.17.6.4.1 Call*

8107 An energy management gateway includes (among others) a specific communications technology
8108 (denoted here as "CP"). The implementation provides scanning other devices. The user selects "CP"
8109 in the gateway's user interface and triggers "Scan". The user interface then creates the subsequent
8110 XML and provides it to the Network Management Responsible of "CP":

```
8111 <networkManagementScanNetworkCall>
8112     <timeout>PT1M</timeout>
8113 </networkManagementScanNetworkCall>
```

8114 In this example "CP" is assumed to not require "scanSetup". During the scan the Network
8115 Management Responsible reports discovered devices as described above. Upon the successful
8116 completion of the process the Network Management Responsible also provides a proper
8117 *networkManagementProcessStateData* notification to the user interface.

8118

5.3.17.7 networkManagementDiscoverCall

5.3.17.7.1 General

If a node changed its functionality, it can announce the changes with all details to all nodes interested in that information. This approach can generate a lot of unneeded traffic as not every recipient requires knowledge of all changes. Hence, the *networkManagementDiscoverCall* can be sent instead to those nodes to tell them, which address to discover (again). Then the nodes can still decide to query for details on the changes.

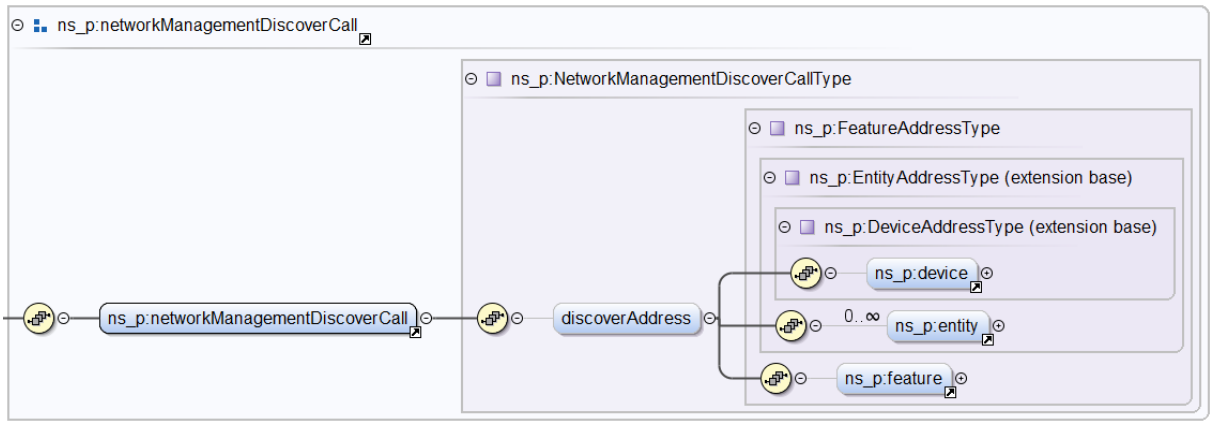


Figure 127: networkManagementDiscoverCall function overview

5.3.17.7.2 Detailed description of elements

Element	Type	Description
discoverAddress	*Common data type "FeatureAddressType". See section 3.10.3.6.*	The SPINE address of the node that will be discovered in detail. Contains elements for *device*, *entity* and *feature*.
discoverAddress.device	*See section 3.10.3.6.*	Device address part.
discoverAddress.entity (list)	*See section 3.10.3.6.*	Entity (list) address part.
discoverAddress.feature	*See section 3.10.3.6.*	Feature address part.

Table 245: networkManagementDiscoverCall function element description

5.3.17.7.3 Available selectors

None.

5.3.17.7.4 Examples

5.3.17.7.4.1 Call

If the functionality of a node has changed, it can send a *networkManagementDiscoverCall* to ask other nodes to discover the changed node:

```
<networkManagementDiscoverCall>
    <discoverAddress>
        <device>d:_i:46925_CP\_pqtl-276381626831726378123568l3</device>
        <entity>3</entity>
```



```
    </discoverAddress>
</networkManagementDiscoverCall>
```

5.3.17.8 networkManagementAbortCall

5.3.17.8.1 General

This function can be used to model the request to abort all ongoing processes triggered by *networkManagementAddNodeCall*, *networkManagementRemoveNodeCall*, *networkManagementModifyNodeCall*, *networkManagementScanNetworkCall* or *networkManagementDiscoverCall*.

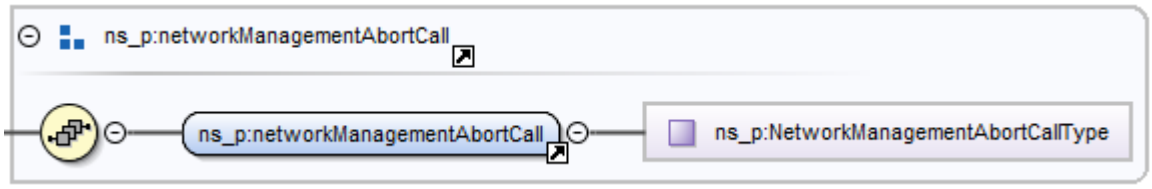


Figure 128: networkManagementAbortCall function overview

5.3.17.8.2 Detailed description of elements

In this version of the specification the function networkManagementAbortCall does not contain child elements.

5.3.17.8.3 Available selectors

None.

5.3.17.8.4 Examples

5.3.17.8.4.1 Call

We assume a user triggered a scan of the communications technology "CP" as described in 5.3.17.6.4. Furthermore, we assume the implementation provides the possibility to abort this process. 25 seconds after the user initiated the scan he decides to abort the process and selects the "Abort" button on the gateway's user interface. The user interface then creates the subsequent XML and provides it to the Network Management Responsible of "CP":

```
<networkManagementAbortCall/>
```

Upon the abortion of the process the Network Management Responsible provides a proper *networkManagementProcessStateData* notification to the user interface.

5.3.17.9 networkManagement...Data

The *networkManagement...Data* functions inform about the process of actions triggered with the aforementioned calls, deliver or change the current joining mode or report new candidates (found

via the *networkManagementScanNetworkCall*). The device, entity and feature descriptions deliver detailed information about nodes (in the meaning of a node-graph including all devices, entities and features) and specific information about entities and features.

5.3.17.10 networkManagementProcessStateData

5.3.17.10.1 General

This function can be used to model the state of a process triggered by *networkManagementAddNodeCall*, *networkManagementRemoveNodeCall*, *networkManagementModifyNodeCall*, *networkManagementScanNetworkCall* or *networkManagementDiscoverCall*.

The underlying revision of the SPINE data model supports only notification of this function (i.e. read, write, or reply mechanisms are not supported).

Please note this function does not support the identification of a specific process. This should be considered carefully in case an implementation supports parallel processes.

It is advised to notify an instance of *networkManagementProcessStateData* only to the originator of the related network management process. Considering the architecture example of section 5.3.17.1.2 this would mean only the software application that triggered a certain process would be notified by the "Network Management Responsible".

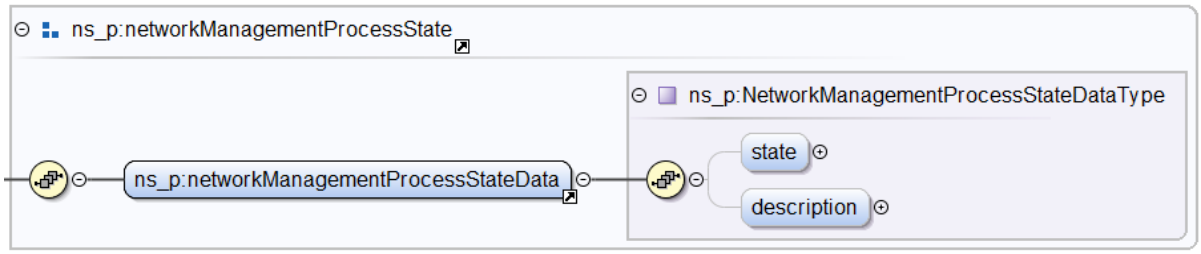


Figure 129: networkManagementProcessStateData function overview

5.3.17.10.2 Detailed description of elements

Element	Type	Description
state	*Enum (see* Table 247*):* NetworkManagementProcessStateStateType	The state of the process.
description	*Common data type "DescriptionType". See section 3.10.1.3.*	Descriptive information.

Table 246: networkManagementProcessStateData function element description

Enumeration **NetworkManagementProcessStateStateType**:

Value	Description
succeeded	The process completed successfully.
failed	The process failed.
aborted	The process was aborted.

Table 247: Enumeration NetworkManagementProcessStateStateType

8201

8202 *5.3.17.10.3 Available selectors*
8203 None.

8204

8205 *5.3.17.10.4 Examples*

8206 *5.3.17.10.4.1 Notify*
8207 Section 5.3.17.3.4 described the creation of a *networkManagementProcessStateData* notification as
8208 a result of the process of adding a display. This XML could look like this:

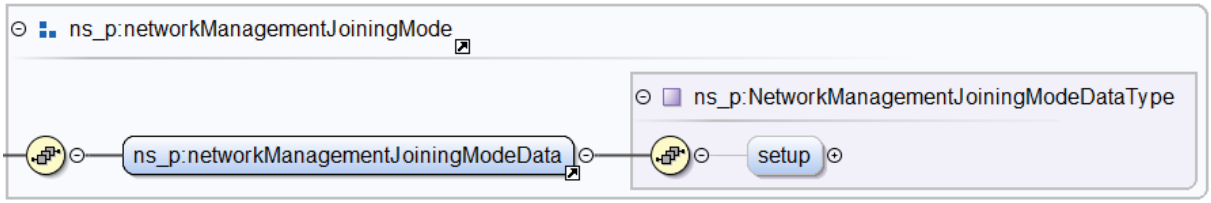
8209 `<networkManagementProcessStateData>`
8210 `  <state>succeeded</state>`
8211 `  <description>Added node.</description>`
8212 `</networkManagementProcessStateData>`

8213

8214 ***5.3.17.11 networkManagementJoiningModeData***

8215 *5.3.17.11.1 General*
8216 Some communications technologies or implementations permit a temporary or permanent
8217 acceptance of other devices or at least a report of discovery. I.e. upon an incoming joining request of
8218 another device the local device may add it automatically to its list of accepted devices. In terms of
8219 the SPINE network management this means a local device would be added automatically to the own
8220 SPINE network. As mentioned before, the new device also might just be reported instead of
8221 automatically added. The function *networkManagementJoiningModeData* permits to model the
8222 configuration of the local node's joining behaviour.

8223 Please note joining does not apply to a "scan network" process (see 5.3.17.6). But an "add node"
8224 process initiated on another device might lead to an incoming joining request on the local device,
8225 e.g. Depending on the joining mode configuration this request can be skipped (rejected), reported
8226 (see 5.3.17.12) or automatically added (see 5.3.17.13.4.2).



8227

8228 *Figure 130: networkManagementJoiningModeData function overview*

8229

8230 *5.3.17.11.2 Detailed description of elements*

Element	Type	Description
setup	*Simple type "NetworkManagementSetupType" (restriction of xs:string)*	Technology dependent or implementation dependent configuration of the joining behaviour. In addition to the kind of joining procedure this could also contain parameters for

		a temporary enabling only, acceptance of specific devices only, etc., e.g.

Table 248: networkManagementJoiningModeData function element description

5.3.17.11.3 Available selectors

None.

5.3.17.11.4 Examples

5.3.17.11.4.1 Write

An energy management gateway includes (among others) a specific communications technology (denoted here as "CP"). The implementation provides so-called pairing with another device. In this example it will not be discussed which kind of joining procedure applies (adding the new device or just reporting it). But we assume "CP" requires some proprietary parameters.

The user chooses "CP" in the gateway's user interface and selects "Enable pairing". The user interface then creates the subsequent XML and provides it together with a *write* classifier to the Network Management Responsible of "CP" which configures its joining behaviour:

```
<networkManagementJoiningModeData>
    <setup>pairingOn;T=40;v=130.255</setup>
</networkManagementJoiningModeData>
```

5.3.17.12 networkManagementReportCandidateData

5.3.17.12.1 General

As already described in 5.3.17.6.1 and 5.3.17.11.1 some processes might require the report of a potential (but not yet accepted/added) device. The function *networkManagementReportCandidateData* can be used to model such a report.

Please note the reported data strongly depends on the implementation and possibilities of the related communications technology.

The underlying revision of the SPINE data model supports only notifications of this function (i.e. read, write and reply mechanisms are not supported).

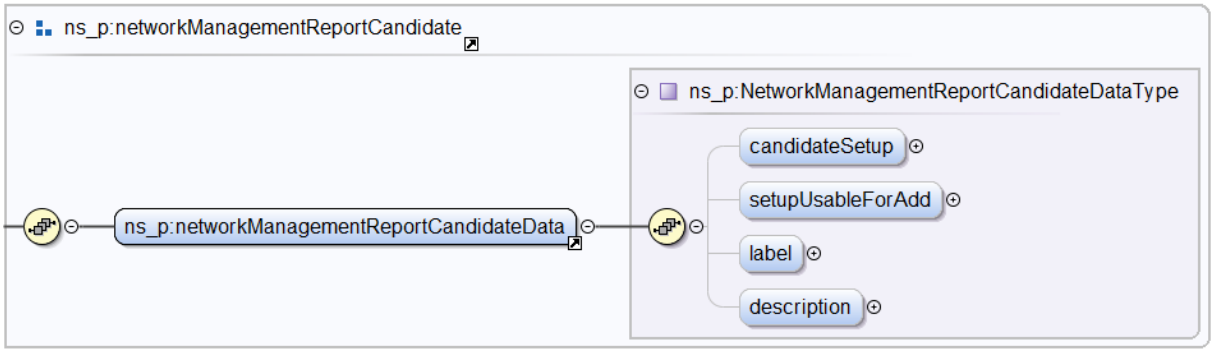


Figure 131: networkManagementReportCandidateData function overview

8260

8261 *5.3.17.12.2 Detailed description of elements*

Element	Type	Description
candidateSetup	*Simple type "NetworkManagementCandidateSetupType" (restriction of xs:string)*	Technology dependent or implementation dependent information on the reported node. This element is intended to provide information usable for an "add node" process. Whether its content could be copied unchanged into an element *nativeSetup* of *networkManagementAddNodeCall* depends on the element *setupUsableForAdd*.
setupUsableForAdd	xs:boolean (W3C standard type)	Only if this element is set to *true* the content of *candidateSetup* could be used for *nativeSetup* of *networkManagementAddNodeCall*.
label	*Common data type "LabelType". See section 3.10.1.2.*	A short label of the reported node.
description	*Common data type "DescriptionType". See section 3.10.1.3.*	Descriptive information on the reported node.

8262 *Table 249: networkManagementReportCandidateData function element description*

8263

8264 *5.3.17.12.3 Available selectors*

8265 None.

8266

8267 *5.3.17.12.4 Examples*

8268 *5.3.17.12.4.1 Notify*

8269 We assume the "scan network" example of 5.3.17.6.4 occurs on a device with implementation and
8270 communications technology capable of providing sufficient details of reported devices. Among
8271 others, the Network Management Responsible provides this XML as notification to the user interface
8272 due to a discovered device:

```
8273 <networkManagementReportCandidateData>
8274     <candidateSetup>21.6.5.3.11;R-2.7;at-42.344;pqtl-
8275 27638162683172637812356813</candidateSetup>
8276     <setupUsableForAdd>true</setupUsableForAdd>
8277     <label>Display</label>
8278     <description>Device type: Display/Generic</description>
8279 </networkManagementReportCandidateData>
```

8280

5.3.17.13 networkManagementDeviceDescriptionListData

5.3.17.13.1 General

This function can be used to model SPINE network management relevant detailed information about the “device level” of a node.

It is recommended to notify an instance of *networkManagementDeviceDescriptionListData* to all "parts" requiring knowledge of "any" added/removed/modified device. Considering the architecture example of section 5.3.17.1.2 this could mean all those software applications that want to react dynamically on such changes could be notified by the "Network Management Responsible". Thus, each of these applications could verify whether the reported change affects its own tasks.

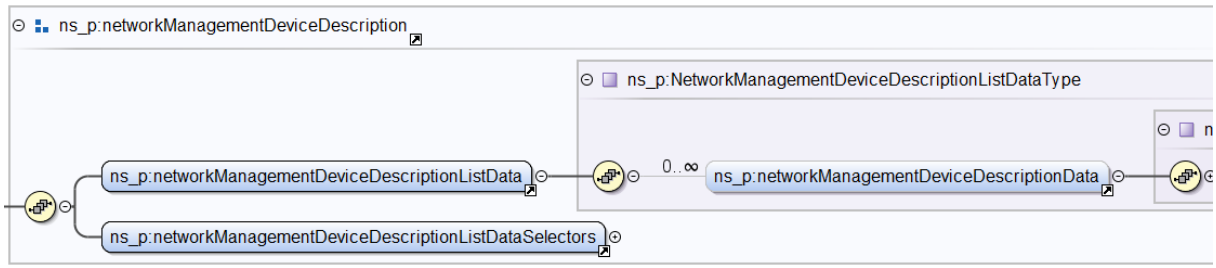
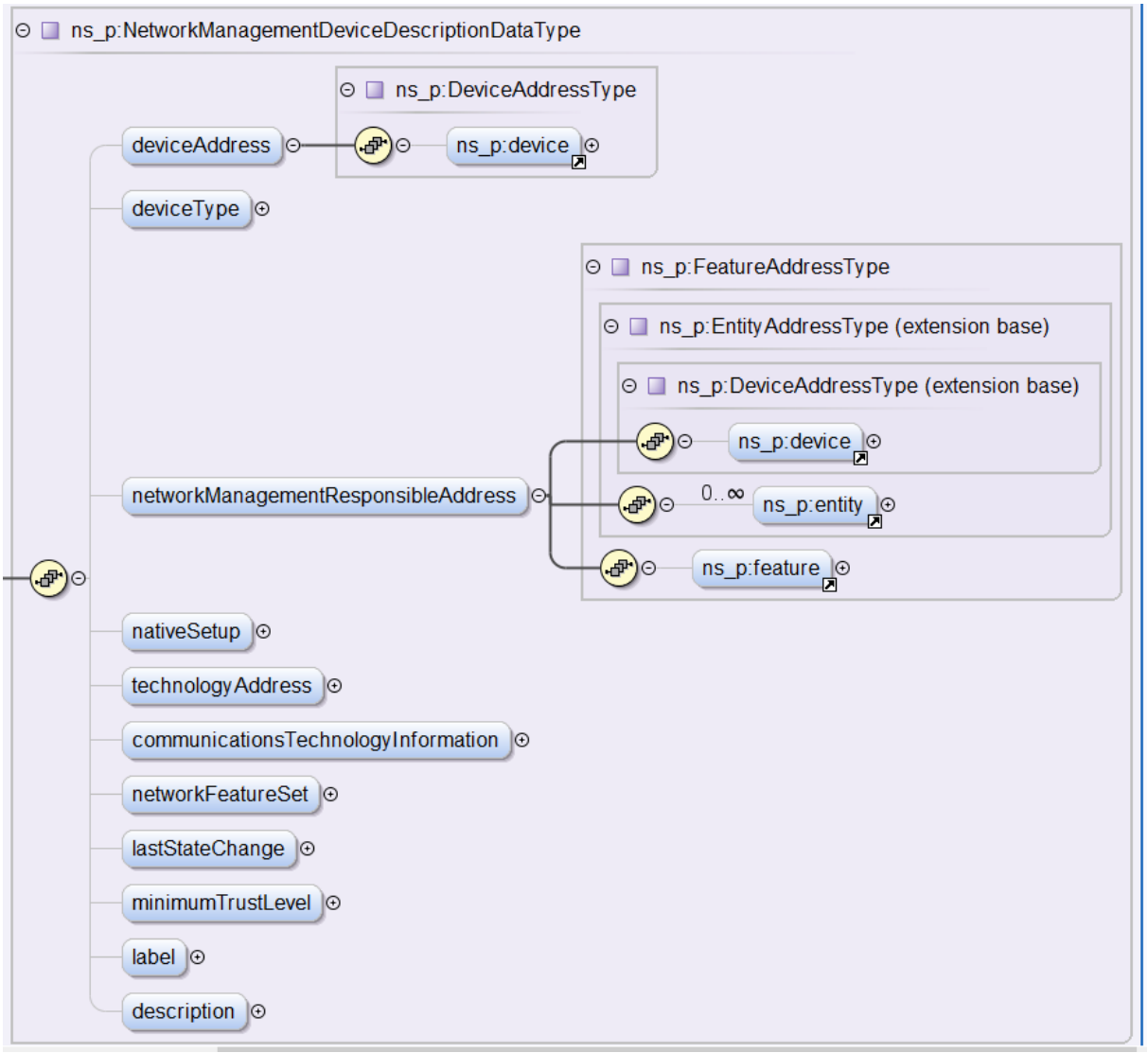


Figure 132: networkManagementDeviceDescriptionListData function overview, part 1



8292

8293 *Figure 133: networkManagementDeviceDescriptionListData function overview, part 2*

8294

8295 *5.3.17.13.2 Detailed description of elements*

Element	Type	Description
deviceAddress	*Common data type "DeviceAddressType". See section 3.10.3.4.*	The node address of the node. Contains elements for *device*.
deviceAddress.device	*See section 3.10.3.4.*	Device address part.
deviceType	*Common data type "DeviceTypeType". See section 3.10.1.25.*	The type of the device. E.g. Washer, Dryer, ...
networkManagementResponsibleAddress	*Common data type "FeatureAddressType" . See section 3.10.3.6.*	The address of the so-called “Network Management Responsible” (see section 5.3.17.1.2) of *deviceAddress*. Contains elements for *device*, *entity* and *feature*.
networkManagementResponsibleAddress.device	*See section 3.10.3.6.*	Device address part.
networkManagementResponsibleAddress.entity (list)	*See section 3.10.3.6.*	Entity (list) address part.

networkManagementResponsibleAddress.feature	*See section 3.10.3.6.*	Feature address part.
nativeSetup	*Simple type "NetworkManagementNativeSetupType" (restriction of xs:string)*	Technology dependent or implementation dependent configuration to identify (and maybe configure) the device.
technologyAddress	*Simple type "NetworkManagementTechnologyAddressType" (restriction of xs:string)*	The address the device has in its own communications technology.
communicationsTechnologyInformation	*Simple type "NetworkManagementCommunicationsTechnologyInformationType" (restriction of xs:string)*	Technology dependent or implementation dependent information on the device with focus on communications aspects. *communicationsTechnologyInformation* is intended to be rather "read only" (but may change dependent on configuration, e.g.).
networkFeatureSet	*Enum (see* Table 251*):* NetworkManagementFeatureSetType	Specifies which network capabilities the device has (simple, smart, router, ...).
lastStateChange	*Enum (see* Table 252*):* NetworkManagementStateChangeType	This element can contain the last change of its network management state (*added*, *removed*, etc.).
minimumTrustLevel	*Simple type "NetworkManagementMinimumTrustLevelType" (restriction of xs:string)*	Contains the lowest trust level needed, to be allowed to communicate with this node.
label	*Common data type "LabelType". See section 3.10.1.2.*	A short label of the device.
description	*Common data type "DescriptionType". See section 3.10.1.3.*	Descriptive information on the device.

Table 250: networkManagementDeviceDescriptionData function element description

Enumeration **NetworkManagementFeatureSetType**:

Value	Description
gateway	Connects to devices with another communications technology; keeps a list of its “attached devices” and provides routing to them.
router	Reserved for future use.
smart	Keeps a list of devices of interest for its communication and may permit routing to them.
simple	Can only communicate with directly reachable devices (i.e. no communication via an intermediate SPINE device). Does not provide routing to other devices.

Table 251: Enumeration NetworkManagementFeatureSetType

Enumeration **NetworkManagementStateChangeType**:

Value	Description
added	The device was added. See also 5.3.17.3.
removed	The device was removed. See also 5.3.17.4.
modified	The device was modified. See also 5.3.17.5.

Table 252: Enumeration NetworkManagementStateChangeType

5.3.17.13.3 Available selectors

- deviceAddress.device

5.3.17.13.4 Examples

5.3.17.13.4.1 Notify (change of a "FridgeFreezer")

After the information of a device has changed, the updated device information is notified to all subscribed instances (the "filter/partial" part to announce the "restricted function exchange is simplified by "..." just to improve the readability"):

```
...
<networkManagementDeviceDescriptionListData>
    <networkManagementDeviceDescriptionData>
        <deviceAddress>
            <device>d:_i:46925_CP\_irto-746298673562109563824385 62</device>
        </deviceAddress>
        <deviceType>FridgeFreezer</deviceType>
        <technologyAddress>21.6.5.3.11</technologyAddress>
<communicationsTechnologyInformation>CP;addr=21.6.5.3.11;shortAddr=114</com
municationsTechnologyInformation>
        <networkFeatureSet>smart</networkFeatureSet>
    </networkManagementDeviceDescriptionData>
</networkManagementDeviceDescriptionListData>
```

5.3.17.13.4.2 Notify (added "Display")

We assume a user triggered adding a display as described in section 5.3.17.3.4. That section mentions the notification of a proper *networkManagementDeviceDescriptionListData* instance. In this example it is assumed the Network Management Responsible of "CP" has the device address "d:_i:46925_CP_A873C6BC09EF4E50" and the display has been assigned the node address "d:_i:46925_CP_pqtl-276381626831726378123568 13". Thus, the subsequent XML could be created (the "filter/partial" part to announce the "restricted function exchange is simplified by "..." just to improve the readability"):

```
...
<networkManagementDeviceDescriptionListData>
    <networkManagementDeviceDescriptionData>
        <deviceAddress>
            <device>d:_i:46925_CP\_pqtl-276381626831726378123568 13</device>
        </deviceAddress>
        <deviceType>Generic</deviceType>
        <networkManagementResponsibleAddress>
            <device>d:_i:46925_CP\_A873C6BC09EF4E50</device>
            <entity>0</entity>
        </networkManagementResponsibleAddress>
```



```
        <nativeSetup>21.6.5.3.11;R-2.7;at-42.344;pqtl-
276381626831726378123568l3</nativeSetup>
        <lastStateChange>added</lastStateChange>
        <minimumTrustLevel>8</minimumTrustLevel>
        <label>Display</label>
    </networkManagementDeviceDescriptionData>
</networkManagementDeviceDescriptionListData>
```

5.3.17.13.4.3Read-reply (from gateway with full device list information)

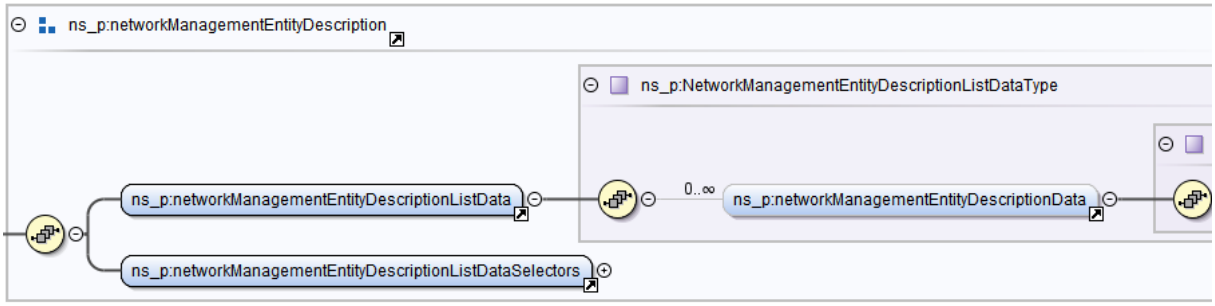
We assume a user added a display as described in sections 5.3.17.3.4 and 5.3.17.13.4.2. In this example, “CP” provides a resource with aggregated information on all “device description details” of its attached devices and itself. We also assume only the display is currently attached via "CP". The user wants to see the list of devices from the described resource. He selects this option on the user interface. The user interface then creates a proper request to the Network Management Responsible’s aggregating resource of "CP" and finally gets a proper response where the Network Management Responsible itself and the display are described:

```
<networkManagementDeviceDescriptionListData>
    <networkManagementDeviceDescriptionData>
        <deviceAddress>
            <device>d:_i:46925_CP\_A873C6BC09EF4E50</device>
        </deviceAddress>
        <networkFeatureSet>gateway</networkFeatureSet>
        <lastStateChange>modified</lastStateChange>
        <minimumTrustLevel>8</minimumTrustLevel>
        <label>CP mapping</label>
    </networkManagementDeviceDescriptionData>
    <networkManagementDeviceDescriptionData>
        <deviceAddress>
            <device>d:_i:46925_CP\_pqtl-276381626831726378123568l3</device>
        </deviceAddress>
        <deviceType>Generic</deviceType>
        <networkManagementResponsibleAddress>
            <device>d:_i:46925_CP\_A873C6BC09EF4E50</device>
            <entity>0</entity>
        </networkManagementResponsibleAddress>
        <nativeSetup>21.6.5.3.11;R-2.7;at-42.344;pqtl-
276381626831726378123568l3</nativeSetup>
        <lastStateChange>added</lastStateChange>
        <minimumTrustLevel>8</minimumTrustLevel>
        <label>Display</label>
    </networkManagementDeviceDescriptionData>
</networkManagementDeviceDescriptionListData>
```

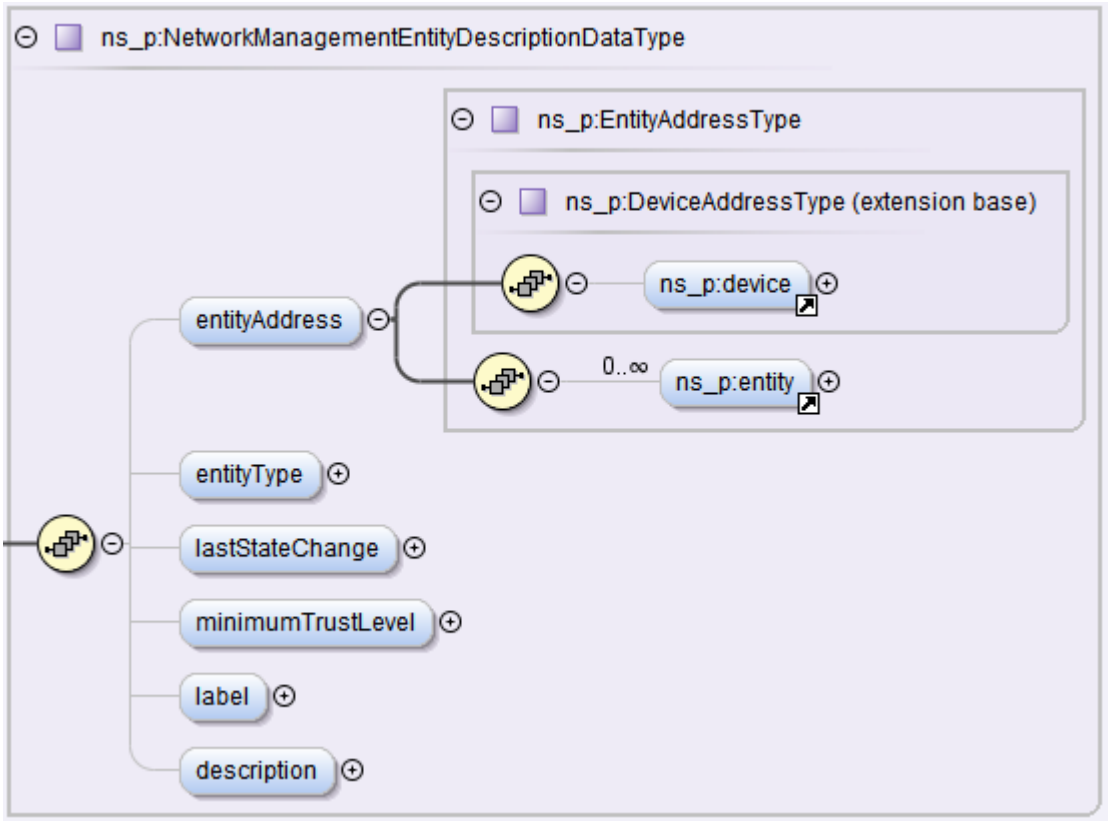
5.3.17.14networkManagementEntityDescriptionListData

5.3.17.14.1General

In addition to the *networkManagementDeviceDescriptionListData* function, this functions can be used to model SPINE network management relevant detailed information about an “entity level” of a node.

8391 

8392 *Figure 134: networkManagementEntityDescriptionListData function overview, part 1*

8393 

8394 *Figure 135: networkManagementEntityDescriptionListData function overview, part 2*

8395

8396 *5.3.17.14.2 Detailed description of elements*

Element	Type	Description
entityAddress	*Common data type "EntityAddressType". See section 3.10.3.5.*	The node address of the node. Contains elements for *device* and *entity*.
entityAddress.device	*See section 3.10.3.5.*	Device address part.
entityAddress.entity (list)	*See section 3.10.3.5.*	Entity (list) address part.
entityType	*Common data type "EntityTypeType". See section 3.10.1.27.*	The type of the entity. E.g. Battery, Fan, ...
lastStateChange	*Enum (see* Table 252*):* NetworkManagement StateChangeType	This element can contain the last change of its network management state (*added*, *removed*, etc.).

minimumTrustLevel	*Simple type "NetworkManagementMinimumTrustLevelType" (restriction of xs:string)*	Contains the lowest trust level needed, to be allowed to communicate with this node.
label	*Common data type "LabelType". See section 3.10.1.2.*	A short label of the entity.
description	*Common data type "DescriptionType". See section 3.10.1.3.*	Descriptive information on the entity.

Table 253: networkManagementEntityDescriptionData function element description

5.3.17.14.3 Available selectors

- entityAddress

5.3.17.14.4 Examples

5.3.17.14.4.1 Notify

After updating information on its entities, a device notifies the corresponding information to all subscribed instances (the “filter/partial” part to announce the “restricted function exchange is simplified by “...” just to improve the readability”):

```
...
<networkManagementEntityDescriptionListData>
    <networkManagementEntityDescriptionData>
        <nodeAddress>
            <device>d:_i:46925_CP\_irto-746298673562109563824385 62</device>
            <entity>1</entity>
        </nodeAddress>
        <entityType>Fridge</entityType>
        <label>Big fridge</label>
    </networkManagementEntityDescriptionData>
    <networkManagementEntityDescriptionData>
        <nodeAddress>
            <device>d:_i:46925_CP\_irto-746298673562109563824385 62</device>
            <entity>2</entity>
        </nodeAddress>
        <entityType>Freezer</entityType>
        <label>Small freezer compartment</label>
    </networkManagementEntityDescriptionData>
</networkManagementEntityDescriptionListData>
```

5.3.17.15 networkManagementFeatureDescriptionListData

5.3.17.15.1 General

In addition to the *networkManagementDeviceDescriptionListData* function, this functions adds feature-specific information which is only needed for nodes on address hierarchy level *feature*.

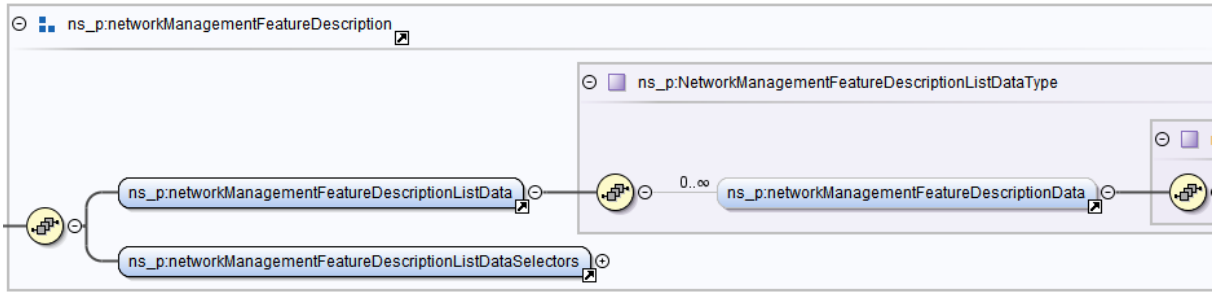


Figure 136: networkManagementFeatureDescriptionListData function overview, part 1

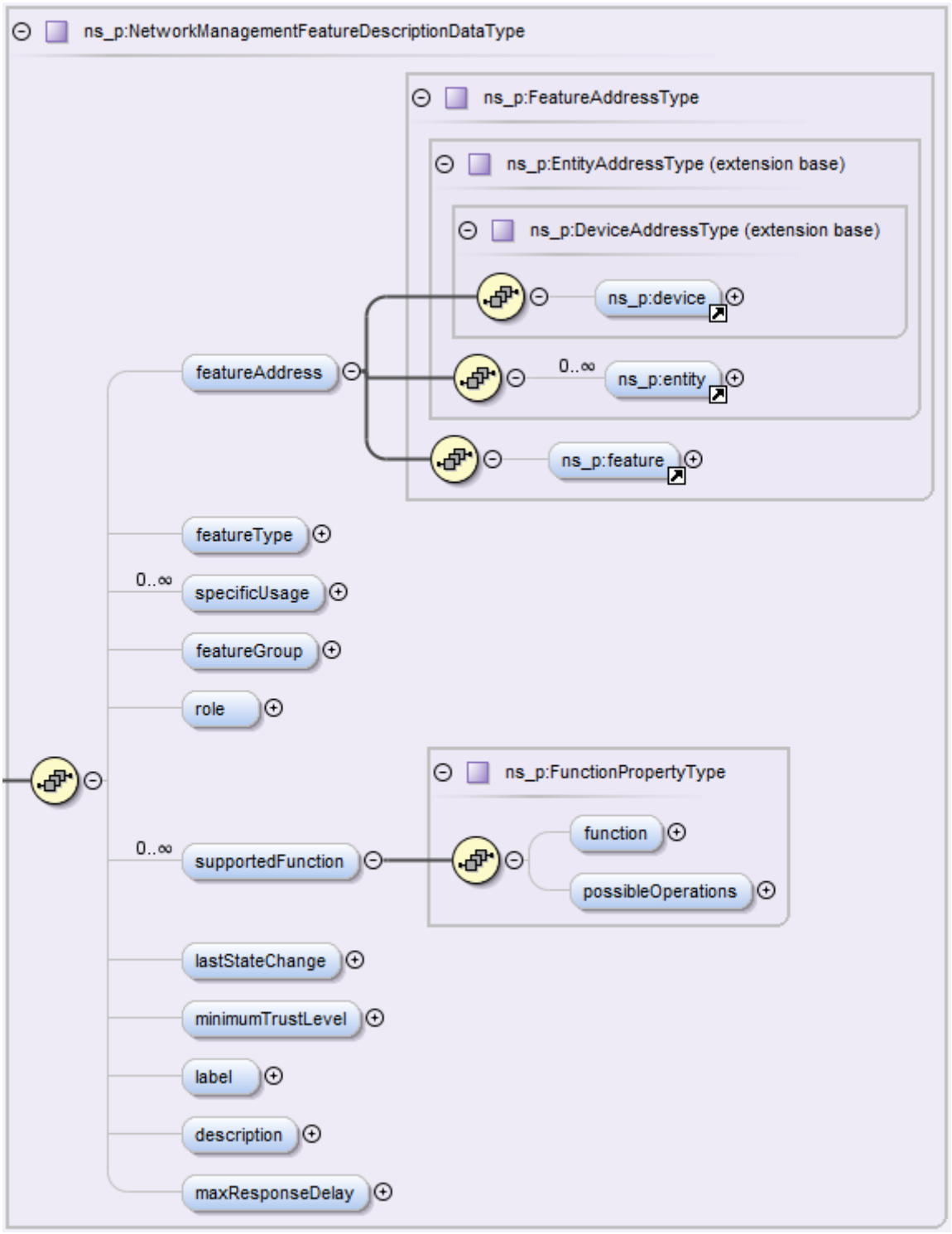
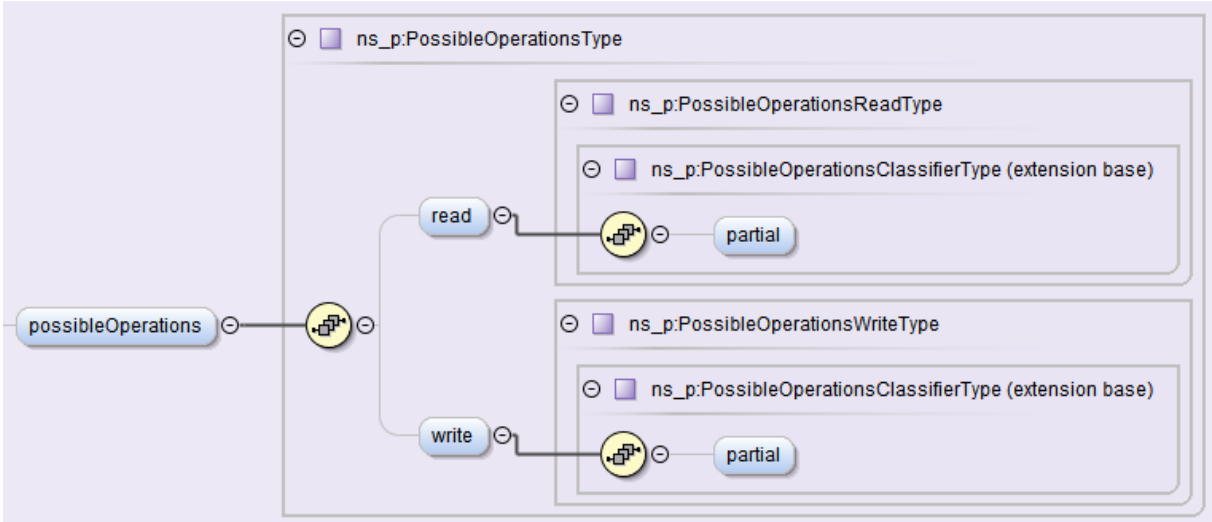


Figure 137: networkManagementFeatureDescriptionListData function overview, part 2



8435

8436 *Figure 138: networkManagementFeatureDescriptionListData function overview, part 3*

8437

8438 *5.3.17.15.2 Detailed description of elements*

Element	Type	Description
featureAddress	*Common data type "FeatureAddressType" . See section 3.10.3.6.*	The node address of the node. Contains elements for *device*, *entity* and *feature*.
featureAddress.device	*See section 3.10.3.6.*	Device address part.
featureAddress.entity (list)	*See section 3.10.3.6.*	Entity (list) address part.
featureAddress.feature	*See section 3.10.3.6.*	Feature address part.
featureType	*Common data type "FeatureTypeType". See section 3.10.1.29.*	The type of the feature. E.g. ActuatorSwitch, Sensing, ...
specificUsage (list)	*Common data type "FeatureSpecificUsageType". See section 3.10.1.31.*	Each occurrence of “specificUsage” denotes a specific usage (see section 2.1.5) that is implemented for this Feature Type. Please consider the proper definition of "featureType" for possible values.
featureGroup	*Common data type "FeatureGroupType". See section 3.10.1.24.*	Information to “link” to related features of this entity. See section 2.1.4 for details.
role	*Common data type "RoleType". See section 3.10.1.23.*	The functional role of this feature (i.e. “client” or “server” or “special”).
supportedFunction (list)	*Common data type "FunctionPropertyType". See section 3.10.1.41.*	Each occurrence contains information on a function that is supported on this feature with the given role. Note: This element is most useful for features with the role “server” or “special” to express which functionality is offered for clients. But it can also be used if the role is “client” in order to express client functionality.

supportedFunction.function	*See section 3.10.1.41.*	Contains the name of a supported function.
supportedFunction.possibleOperations	*See section 3.10.1.41.*	Can be used to express some details on operations supported by the given function (which also means the element “function” has to be set properly) if the role is “server” or “special”.
supportedFunction.possibleOperations.read	*See section 3.10.1.41.*	If present, it denotes the feature can receive “read” commands for the given function and creates proper replies.
supportedFunction.possibleOperations.read.partial	*See section 3.10.1.41.*	If present, it denotes the feature permits the creation of somehow “curtailed” responses (in contrast to full copies of its function instance).
supportedFunction.possibleOperations.write	*See section 3.10.1.41.*	If present, it denotes the feature can receive “write” commands for the given function and adjusts its own function instance accordingly.
supportedFunction.possibleOperations.write.partial	*See section 3.10.1.41.*	If present, it denotes the feature accepts also somehow “curtailed” “write” commands (in contrast to a full replacement of its own data).
lastStateChange	*Enum (see* Table 252*):* NetworkManagementStateChangeType	This element can contain the last change of its network management state (*added*, *removed*, etc.).
minimumTrustLevel	*Simple type "NetworkManagementMinimumTrustLevelType" (restriction of xs:string)*	Contains the lowest trust level needed, to be allowed to communicate with this node.
label	*Common data type "LabelType". See section 3.10.1.2.*	A short label of the feature.
description	*Common data type "DescriptionType". See section 3.10.1.3.*	Descriptive information on the feature.
maxResponseDelay	*Common data type "MaxResponseDelayType". See section 3.10.1.42.*	The duration the Server usually needs to generate a proper reply.

8439 *Table 254: networkManagementFeatureDescriptionListData function element description*

8440

8441 *5.3.17.15.3Available selectors*

8442 - featureAddress

8443

5.3.17.15.4 Examples

5.3.17.15.4.1 Notify

After the functionality of an entity has changed, the new feature information is notified to all bound instances (the “filter/partial” part to announce the “restricted function exchange is simplified by “…” just to improve the readability”):

```
...
<networkManagementFeatureDescriptionListData>
    <networkManagementFeatureDescriptionData>
        <featureAddress>
            <device>d:_i:46925_CP\_pqtl-276381626831726378123568l3</device>
            <entity>1</entity>
            <feature>1</feature>
        </featureAddress>
        <featureType>ActuatorLevel</featureType>
        <role>server</role>
        <supportedFunction>
            <function>actuatorLevelData</function>
            <possibleOperations>
                <read/>
                <write/>
            </possibleOperations>
        </supportedFunction>
        <supportedFunction>
            <function>actuatorLevelDescriptionData</function>
            <possibleOperations>
                <read/>
            </possibleOperations>
        </supportedFunction>
        <lastStateChange>modified</lastStateChange>
        <label>Illumination</label>
        <maxResponseDelay>PT20S</maxResponseDelay>
    </networkManagementFeatureDescriptionData>
</networkManagementFeatureDescriptionListData>
```

5.3.18 OperatingConstraints

5.3.18.1 Introduction

This class is designed to model some constraints of an appliance (or functionality) with regards to its energy consumption or production. I.e. this class is NOT intended to control a device. Rather, it enables an appliance to express what needs to be considered especially for the case it permits being controlled by another device. An example is a central energy management system that uses the class DirectControl to control the appliance; in this case the energy management system needs to be aware of the appliance’s operating constraints.

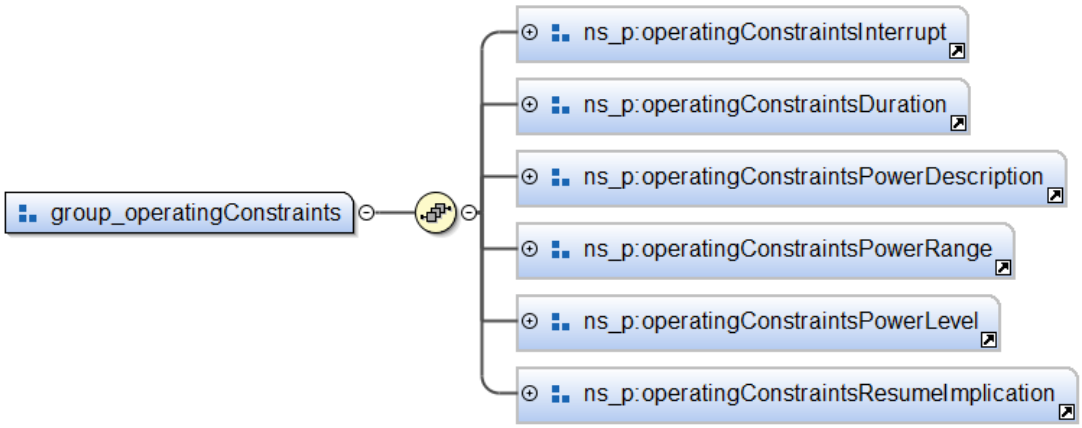


Figure 139: OperatingConstraints function-group overview

In general the class OperatingConstraints can be used with an without relation to the class PowerSequences. As can be seen in the following functions this is usually achieved with the (optional) element “sequenceId”. In fact, most functions of this class are list-based, hence provide a possibility to model also constraints per power sequence.

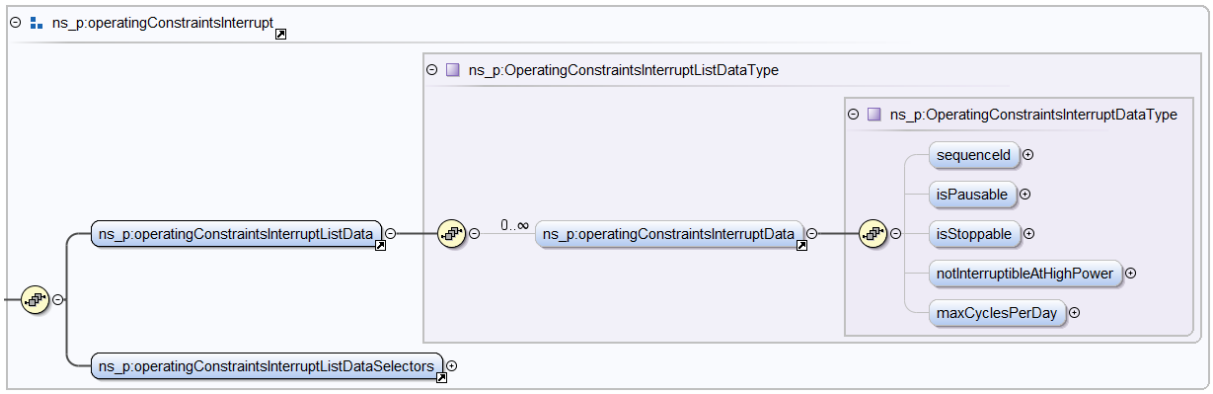
Without a relation to a power sequence the OperatingConstraints functions usually relate to “now”, i.e. the current state of the functionality. With the relation to a power sequence the corresponding constraints apply for the time of the related power sequence. Care should be taken if some constraints depend on the power sequence and other constraints are rather “global”, i.e. independent from a power sequence. In every case the use of a feature with such a function should be clearly defined in order to avoid ambiguities. The following brief recommendations may be helpful to achieve this:

1. Avoid a mixture of operatingConstraintsInterruptData items with and without sequenceId within operatingConstraintsInterruptListData.
2. Within operatingConstraintsInterruptListData NO child element of operatingConstraintsInterruptData should depend on sequenceId in some operatingConstraintsInterruptData items and be independent of sequenceId in other operatingConstraintsInterruptData items.

5.3.18.2 operatingConstraintsInterruptListData

5.3.18.2.1 General

This function models basic constraints for an interrupt of a device's current task.



8509

8510 *Figure 140: operatingConstraintsInterruptListData function overview*

8511

8512 ### *5.3.18.2.2 Detailed description of elements*

Element	Type	Description
sequenceId	*Identifier "PowerSequenceIdType" (see* Table 339*).*	If interrupt constraints are related to a power sequence, the appropriate *sequenceId* is stated in this element.
isPausable	xs:boolean (W3C standard type)	States whether a functionality or task is pausable (true) or not (false).
isStoppable	xs:boolean (W3C standard type)	States whether a functionality or task is stoppable (true) or not (false).
notInterruptibleAtHighPower	xs:boolean (W3C standard type)	States whether a functionality or task is NOT interruptible while it is consuming (or producing) energy (true) or whether it can be interrupted nevertheless (false).
maxCyclesPerDay	xs:unsignedInt (W3C standard type)	If a functionality or task may be started only a maximum number of times per day, this value is denoted here.

8513 *Table 255: operatingConstraintsInterruptListData function element description*

8514

8515 ### *5.3.18.2.3 Available selectors*

8516 - sequenceId

8517

8518 ### *5.3.18.2.4 Examples*

8519 #### *5.3.18.2.4.1 Read-reply*

8520 We assume an energy management system wants to control an appliance. We also assume the
8521 appliance expresses details on its process with a power sequence (with seqenceId “1” in this
8522 example). The energy management system queries the appliance’s operatingConstraintsInterrupt
8523 information and gets the following response:

```
8524 <operatingConstraintsInterruptListData>
8525     <operatingConstraintsInterruptData>
8526         <sequenceId>1</sequenceId>
8527         <isPausable>true</isPausable>
```



```
            <notPausableAtHighPower>true</notPausableAtHighPower>
        </operatingConstraintsInterruptData>
</operatingConstraintsInterruptListData>
```

5.3.18.3 operatingConstraintsDurationListData

5.3.18.3.1 General

This function models constraints on the minimum/maximum duration of active ("on") and pause ("off") phases. Example: A compressor can be damaged if it is started and then switched off too early.

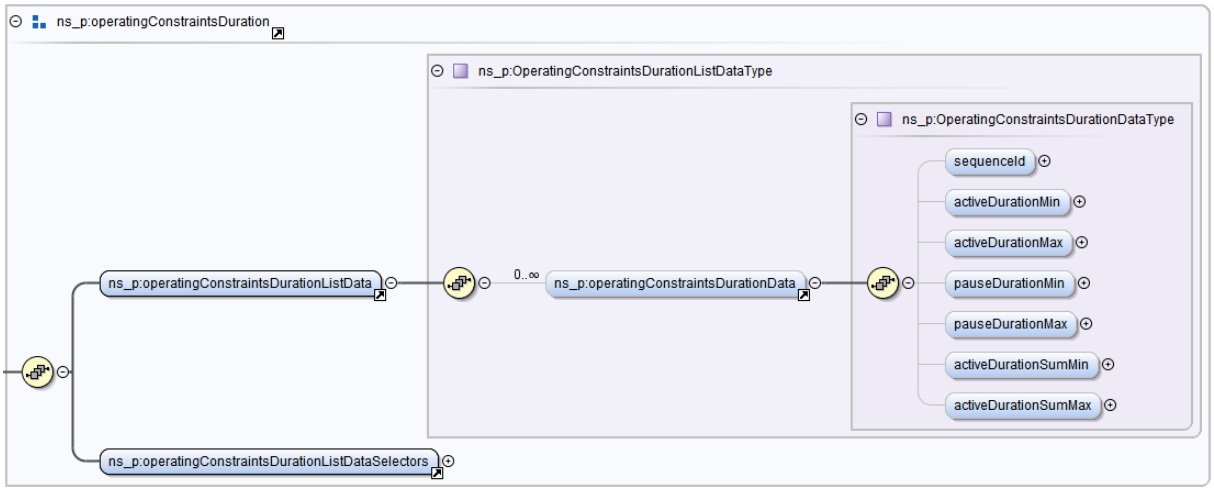


Figure 141: operatingConstraintsDurationListData function overview

5.3.18.3.2 Detailed description of elements

Element	Type	Description
sequenceId	*Identifier "PowerSequenceIdType" (see* Table 339*).*	If duration constraints are related to a power sequence, the appropriate *sequenceId* is stated in this element.
activeDurationMin	xs:duration (W3C standard type)	This is the minimum duration a device has to run without interruption.
activeDurationMax	xs:duration (W3C standard type)	This is the maximum duration a device can run without interruption.
pauseDurationMin	xs:duration (W3C standard type)	This is the minimum duration a device has to pause after the end of an activity.
pauseDurationMax	xs:duration (W3C standard type)	This is the maximum duration a device can pause after the end of an activity.
activeDurationSumMin	xs:duration (W3C standard type)	This is the minimum duration a device has to run in total for a given task (summation of all active times).
activeDurationSumMax	xs:duration (W3C standard type)	This is the maximum duration a device can run in total for a given task (summation of all active times).

Table 256: operatingConstraintsDurationListData function element description

5.3.18.3.3 Available selectors

- sequenceId

5.3.18.3.4 Examples

5.3.18.3.4.1 Read-reply

This example continues the example of section 5.3.18.2.4.1. The energy management system queries for the appliance's duration constraints and gets the following response:

```
<operatingConstraintsDurationListData>
    <operatingConstraintsDurationData>
        <sequenceId>1</sequenceId>
        <activeDurationMin>PT10M</activeDurationMin>
        <activeDurationMax>PT1H</activeDurationMax>
        <pauseDurationMin>PT5M30S</pauseDurationMin>
        <pauseDurationMax>P1D</pauseDurationMax>
        <activeDurationSumMin>PT3H15M</activeDurationSumMin>
        <activeDurationSumMax>PT5H30M</activeDurationSumMax>
    </operatingConstraintsDurationData>
</operatingConstraintsDurationListData>
```

5.3.18.4 operatingConstraintsPowerDescriptionListData

5.3.18.4.1 General

This function models basic information relevant for *operatingConstraintsPowerRangeListData* and *operatingConstraintsPowerLevelListData*.

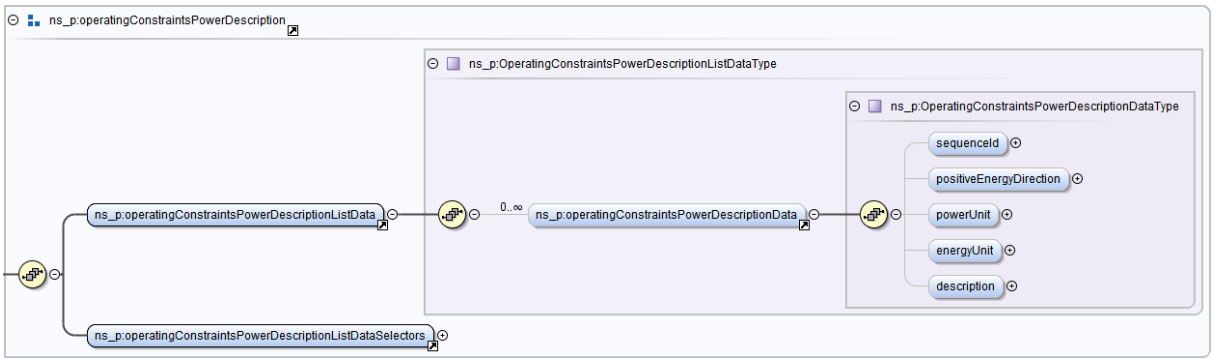


Figure 142: operatingConstraintsPowerDescriptionListData function overview

5.3.18.4.2 Detailed description of elements

Element	Type	Description
sequenceId	*Identifier "PowerSequenceIdType" (see* Table 339*).*	If function elements are related to a power sequence, the appropriate *sequenceId* is stated in this element.
positiveEnergyDirection	*Common data type "EnergyDirectionType". See section 3.10.1.13.*	The *positiveEnergyDirection* states whether energy consumption or production is counted as positive value.

powerUnit	*Common data type "UnitOfMeasurementType". See section 3.10.1.17.*	The unit for all power values in operatingConstraintsPowerRangeListData and operatingConstraintsPowerLevelListData.
energyUnit	*Common data type "UnitOfMeasurementType". See section 3.10.1.17.*	The unit for all energy values in operatingConstraintsPowerRangeListData and operatingConstraintsPowerLevelListData.
description	*Common data type "DescriptionType". See section 3.10.1.3.*	Descriptive information.

Table 257: operatingConstraintsPowerDescriptionListData function element description

5.3.18.4.3 Available selectors

- sequenceId

5.3.18.4.4 Examples

5.3.18.4.4.1 Read-reply

This example continues the example of section 5.3.18.2.4.1. The energy management system queries for the appliance's description on constraints and gets the following response:

```
<operatingConstraintsPowerDescriptionListData>
    <operatingConstraintsPowerDescriptionData>
        <sequenceId>1</sequenceId>
        <positiveEnergyDirection>consume</positiveEnergyDirection>
        <powerUnit>W</powerUnit>
        <energyUnit>Wh</energyUnit>
        <description>Constraints for sequence 1</description>
    </operatingConstraintsPowerDescriptionData>
</operatingConstraintsPowerDescriptionListData>
```

5.3.18.5 operatingConstraintsPowerRangeListData

5.3.18.5.1 General

This function can be used to model a power/energy range(!) a device can operate. An alternative model for power/energy constraints is operatingConstraintsPowerLevelListData.

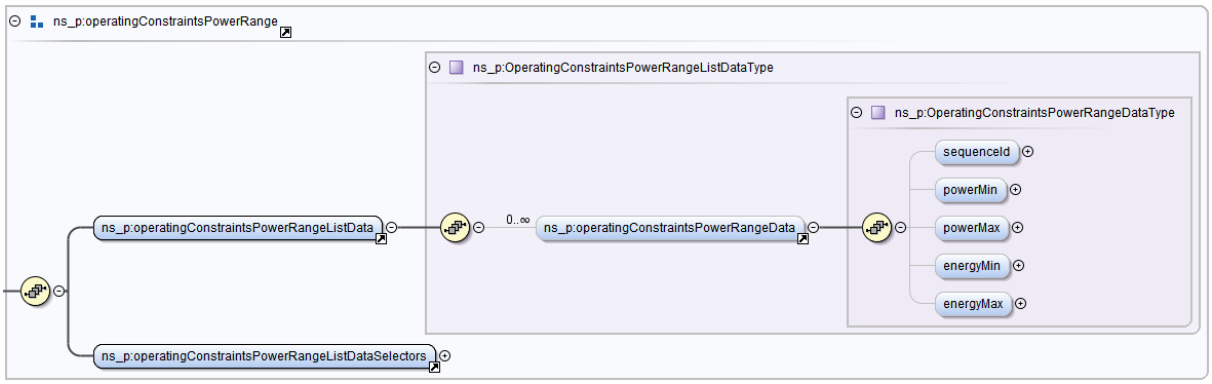


Figure 143: operatingConstraintsPowerRangeListData function overview

5.3.18.5.2 Detailed description of elements

Element	Type	Description
sequenceId	*Identifier "PowerSequenceIdType" (see* Table 339*).*	If the power range is related to a power sequence, the appropriate *sequenceId* is stated in this element.
powerMin	*Common data type "ScaledNumberType". See section 3.10.1.8.*	The minimum power a device can consume/produce.
powerMax	*Common data type "ScaledNumberType". See section 3.10.1.8.*	The maximum power a device can consume/produce.
energyMin	*Common data type "ScaledNumberType". See section 3.10.1.8.*	The minimum energy a device can consume/produce.
energyMax	*Common data type "ScaledNumberType". See section 3.10.1.8.*	The maximum energy a device can consume/produce.

Table 258: operatingConstraintsPowerRangeListData function element description

5.3.18.5.3 Available selectors

- sequenceId

5.3.18.5.4 Examples

5.3.18.5.4.1 Read-reply

This example continues the example of section 5.3.18.4.4.1. The energy management system queries for the appliance's power range constraints and gets the following response:

```
<operatingConstraintsPowerRangeListData>
    <operatingConstraintsPowerRangeData>
        <sequenceId>1</sequenceId>
        <powerMin>
            <number>0</number>
        </powerMin>
        <powerMax>
            <number>2</number>
            <scale>3</scale>
        </powerMax>
        <energyMin>
            <number>5</number>
```



```
            <scale>2</scale>
        </energyMin>
        <energyMax>
            <number>21</number>
            <scale>2</scale>
        </energyMax>
    </operatingConstraintsPowerRangeData>
</operatingConstraintsPowerRangeListData>
```

5.3.18.6 operatingConstraintsPowerLevelListData

5.3.18.6.1 General

This function can be used to model a list of possible power values a device can operate. An alternative model for power constraints is operatingConstraintsPowerRangeListData, where a full range (not only discrete levels) can be modelled.

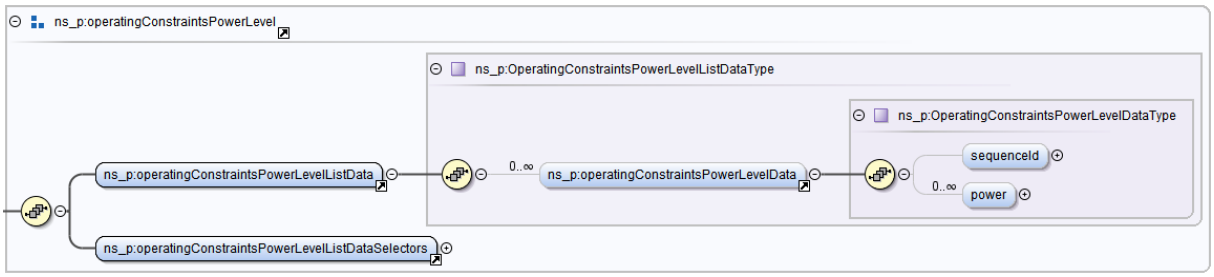


Figure 144: operatingConstraintsPowerLevelListData function overview

5.3.18.6.2 Detailed description of elements

Element	Type	Description
sequenceId	*Identifier "PowerSequenceIdType" (see* Table 339*).*	If the power levels are related to a power sequence, the appropriate *sequenceId* is stated in this element.
power (list)	*Common data type "ScaledNumberType". See section 3.10.1.8.*	Each “power” item contains a possible power level.

Table 259: operatingConstraintsPowerLevelListData element description

5.3.18.6.3 Available selectors

- sequenceId

5.3.18.6.4 Examples

5.3.18.6.4.1 Read-reply

This example continues the example of section 5.3.18.4.4.1. The energy management system queries for the appliance’s power level constraints and gets the following response:

```
<operatingConstraintsPowerLevelListData>
    <operatingConstraintsPowerLevelData>
```



```
            <sequenceId>1</sequenceId>
            <power>
                <number>5</number>
                <scale>2</scale>
            </power>
            <power>
                <number>15</number>
                <scale>2</scale>
            </power>
            <power>
                <number>25</number>
                <scale>2</scale>
            </power>
        </operatingConstraintsPowerLevelData>
</operatingConstraintsPowerLevelListData>
```

5.3.18.7 operatingConstraintsResumeImplicationListData

5.3.18.7.1 General

Pausing a device might be permitted according to above mentioned (duration) constraints. Nevertheless, it can be expected that it has implications if a device's task is paused temporarily and then resumed to continue the task. One possible implication is an additional mechanical stress which reduces the device's lifetime and could be interpreted as additional cost. Such additional costs or additional energy consumption can be modelled with this function in order to be considered for energy management. However, these are typically empirical (thus rather constant) data, therefore should not be considered to be precise.

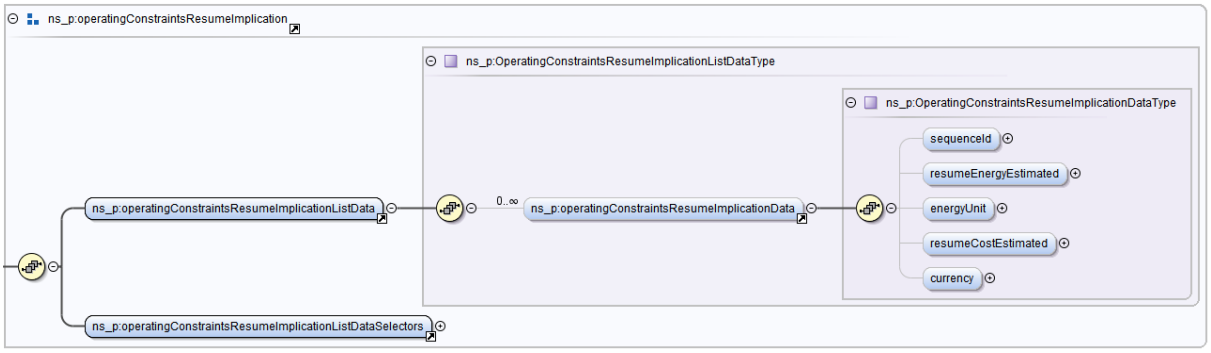


Figure 145: operatingConstraintsResumeImplicationListData function overview

5.3.18.7.2 Detailed description of elements

Element	Type	Description
sequenceId	*Identifier "PowerSequenceIdType" (see* Table 339*).*	If the resume implications are related to a power sequence, the appropriate *sequenceId* is stated in this element.
resumeEnergyEstimated	*Common data type "ScaledNumberType". See section 3.10.1.8.*	Additional energy a device consumes before resuming to its normal operation (after a pause).
energyUnit	*Common data type "UnitOfMeasurement*	Unit for the energy stated in the element resumeEnergyEstimated.

	Type". See section 3.10.1.17.	
resumeCostEstimated	*Common data type "ScaledNumberType". See section 3.10.1.8.*	Additional costs for the resumption of a device to its normal operation.
currency	*Common data type "CurrencyType". See section 3.10.1.19.*	Currency for the price stated in the element resumeCostEstimated.

Table 260: operatingConstraintsResumeImplicationListData function element description

5.3.18.7.3 Available selectors

- sequenceId

5.3.18.7.4 Examples

5.3.18.7.4.1 Read-reply

This example continues the example of section 5.3.18.2.4.1. The energy management system queries for the appliance's resume implications and gets the following response:

```
<operatingConstraintsResumeImplicationListData>
    <operatingConstraintsResumeImplicationData>
        <sequenceId>1</sequenceId>
        <resumeEnergyEstimated>
            <number>5</number>
            <scale>1</scale>
        </resumeEnergyEstimated>
        <energyUnit>Wh</energyUnit>
        <resumeCostEstimated>
            <number>2</number>
            <scale>-2</scale>
        </resumeCostEstimated>
        <currency>EUR</currency>
    </operatingConstraintsResumeImplicationData>
</operatingConstraintsResumeImplicationListData>
```

5.3.19 PowerSequences

5.3.19.1 Introduction

This class primarily permits to model curves of power and energy over time. This is considered in the sub-class PowerTimeSlot (see 5.3.19.2). Secondarily, the class provides definitions for the modelling of power scheduling including alternative plans.

5.3.19.1.1 Basic concepts

Throughout this class two terms are of major interest: *Sequence* and *Slot*. A sequence can be used to describe a "task". It is the most "coarse" view in this class, i.e. a sequence represents all single steps of a whole task. The single steps are represented by slots. Slots are the most "fine" view in this class with regards to the time resolution of a task.

Each sequence and slot can be associated an identifier or number, respectively. The slot numbers of two sequences should be considered independent from each other. I.e. slot number 7 of sequence 1 describes a different slot than slot number 7 of sequence 2. This basically means a slot is only "uniquely" identified in combination with a sequence ID. As formal notation this leads to the following relation:

- 1 sequence has 0..m slots
- 1 slot belongs to 0..1 sequence
- 1 pair of { sequenceId, slotNumber } uniquely identifies 1 slot

These relations are quite general as they permit modelling sequences with no slot and also the use of slots without sequences. However, for the most applications it can be expected that constraints are required where a sequence has at least 1 slot and a slot belongs to exactly one sequence.

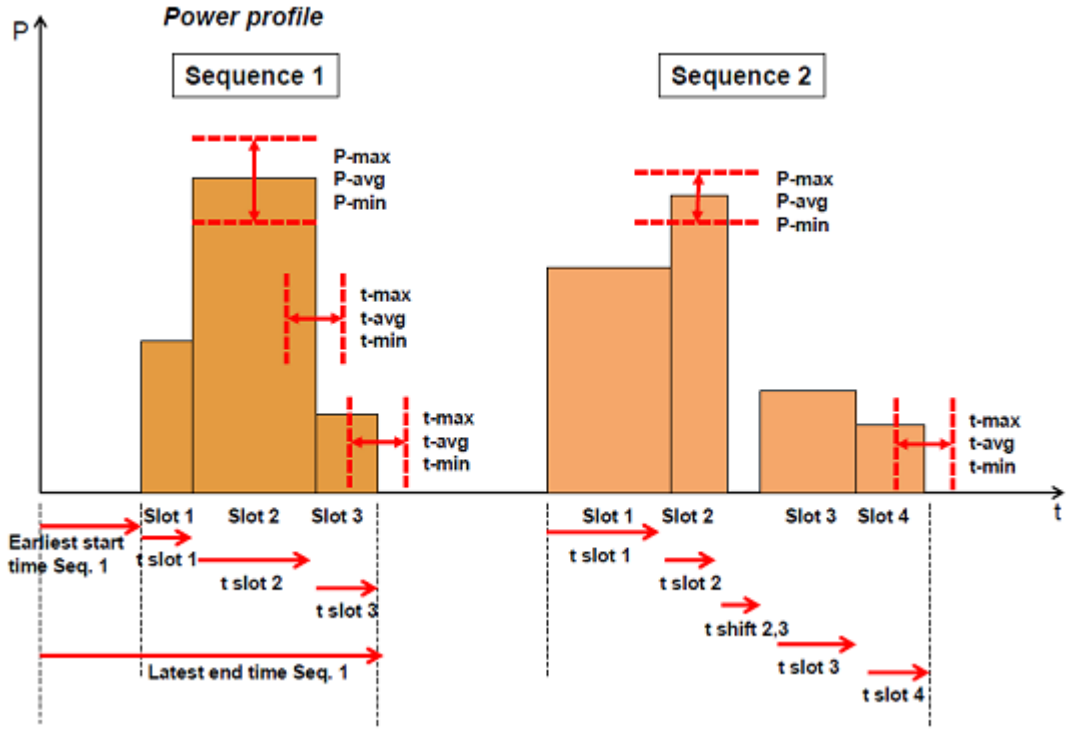
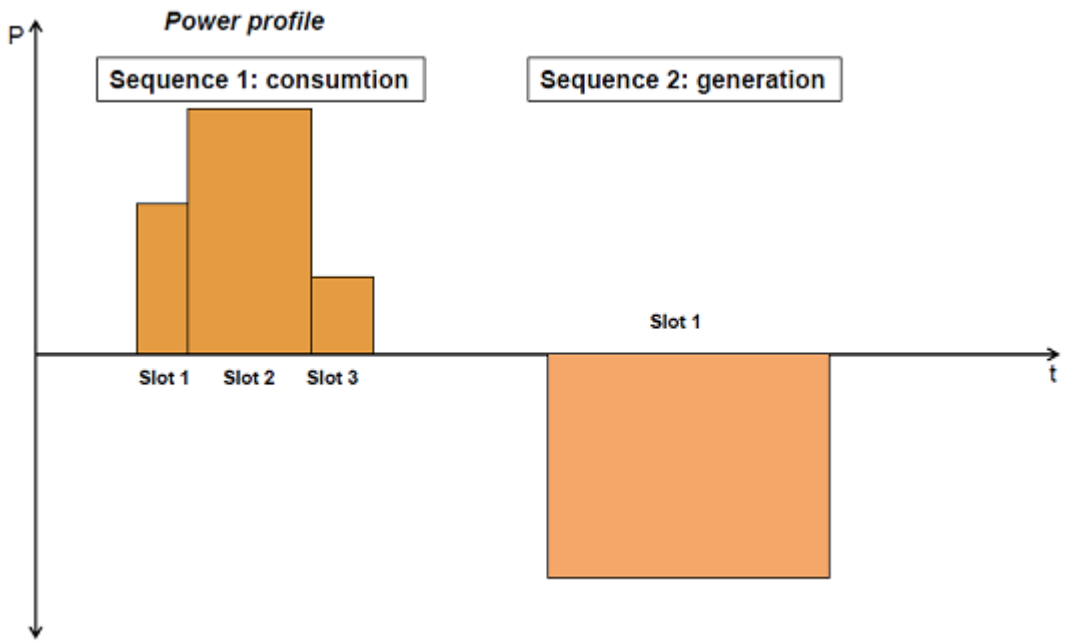


Figure 146: Concept overview of power sequences: Power P over time t; sequences are constituted of slots

Irrespective of whether the Smart Device is a consumer or producer it needs to announce a kind of expected energy consumption or generation (power profile) to allow energy allocation within smart premises.

For example, a heat pump can ask for allocating 2 sequences per day, once in the morning, once in the afternoon, each sequence with different phases of power consumption. Alternatively, a smart device can offer multiple sequences as alternatives to each other (for the same time period, e.g.). An example for such alternative plans can be a battery pack offering charging or discharging. A customer energy management (CEM) system may then choose between these possibilities. Some more details on alternative plans will be described later on.

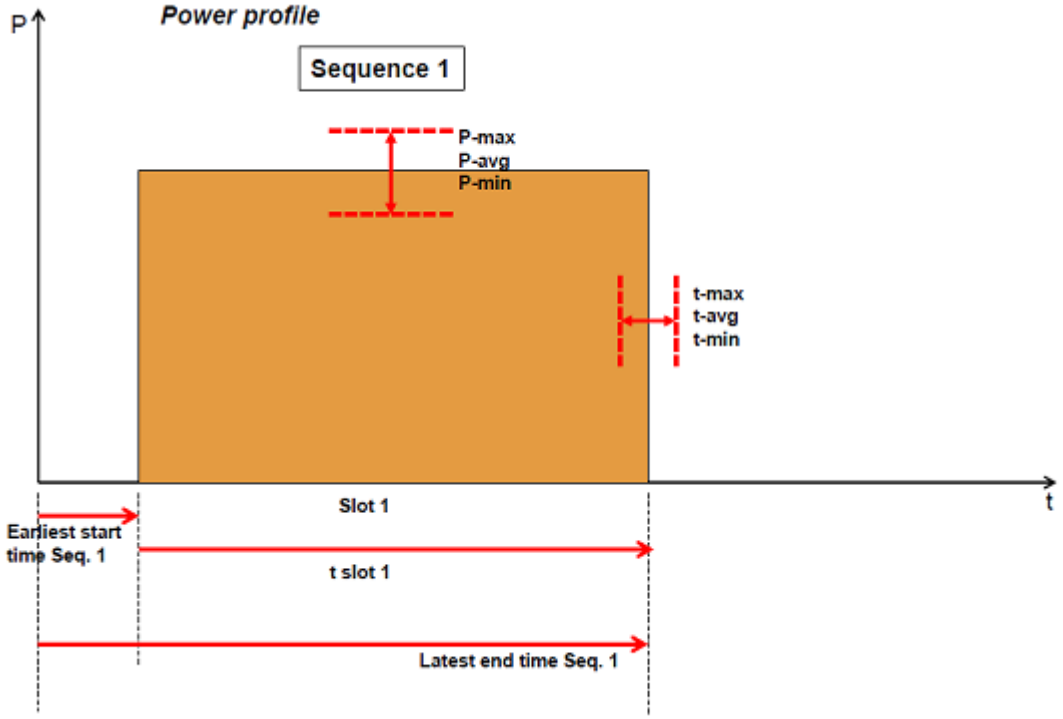
Power sequences may exist for energy consumption or production.



8734

8735 *Figure 147: Example of a device that first consumes and later on generates power. In this case "consumption" is assigned a*
8736 *positive value.*

8737 The easiest way to express the expected energy consumption / generation is one sequence with one
8738 timeframe with one power value over timeslot.



8739

8740 *Figure 148: Example of a sequence with only one slot with some possible parameter boundaries.*

8741

5.3.19.1.2 Repetition concept

In its most basic form, a sequence can execute its explicitly announced slots one after the other and is completed when the last of these slots was completed. However, the PowerSequences also permits a “repetition concept” where a sequence is executed a given number of times.

As example we consider a sequence with three explicitly announced slots. As each slot represents a “phase” of the sequence, we denote the three phases “A”, “B”, and “C”. In this example the sequence shall be executed two times. This finally means that this sequence of phases is executed: “A”, “B”, “C”, “A”, “B”, “C”.

5.3.19.1.3 Alternative sequences

As mentioned above, the class permits the modelling of “alternative plans”. This means two or more sequences may be offered by an appliance to an energy management system (e.g.) to select none or exactly one of the alternatives for the execution. Among others, the so-called “state” of a power sequence can be useful to distinguish the selected sequence from the remaining alternatives (see section 5.3.19.9).

Some examples describe subsequent tasks (an appliance executes first sequence 1 and some time later it executes sequence 2, e.g.). The PowerSequences class permits as well the combination of alternatives and sequential plans. I.e. alternatives are grouped together and the groups are considered sequentially. Section 5.3.19.7 is useful to model groups of alternatives.

5.3.19.1.4 Overview on the class modules

The SPINE data model of this class follows the design principle of modular definitions. As a result the underlying revision of the SPINE data model contains almost no relation to slots within the sub-class PowerSequence. Thus, in theory it is possible to use PowerSequence without PowerTimeSlot. However, this is not the primary use case of PowerSequence and is not considered further in this document.

PowerTimeSlot, on the other hand, uses *sequenceId* in many functions, e.g. Therefore, using PowerTimeSlot without PowerSequences is not considered in detail.

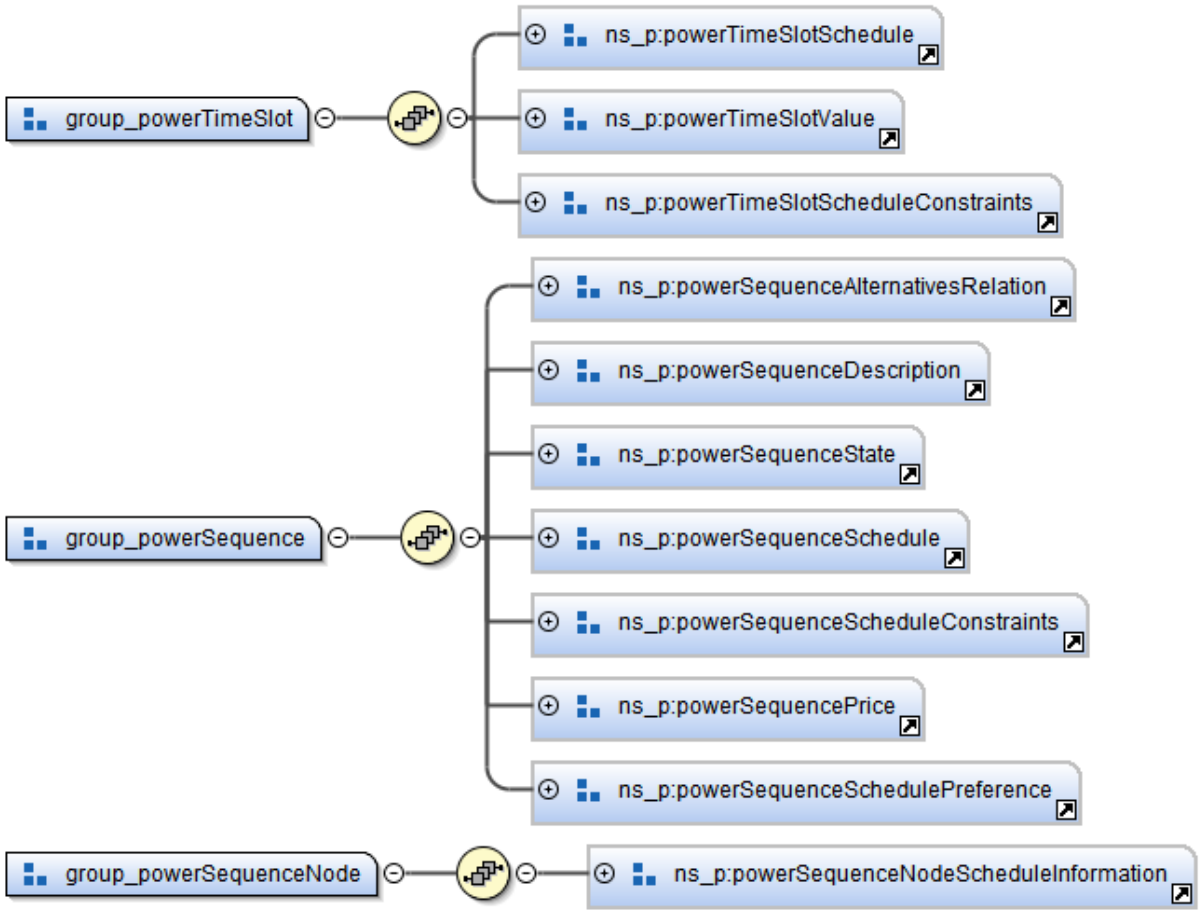


Figure 149: PowerSequences function-group overview (data)

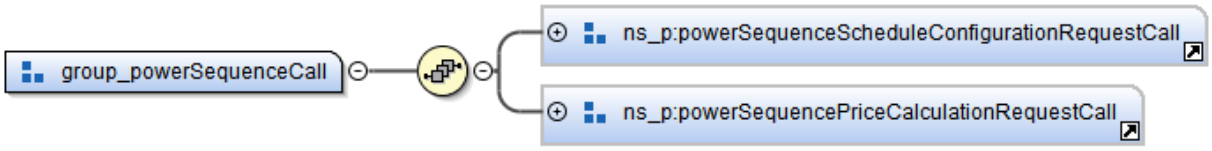


Figure 150: PowerSequences function-group overview (call)

5.3.19.2 Sub-class PowerTimeSlot – Power values, facets, curves

The root of this sub-class is the consideration of single power or energy values. Each value can be "enriched" with miscellaneous facets like a timestamp, durations, dependency to other values, information on its "precision", etc. The combination of the single values or facets into lists permits modelling power curves over time, or modelling of "variants" at a given time, to give two examples.

With regards to the time and dependency facets it is possible to model scheduling of periods or whole curves. Constraints on scheduling can be modelled, too.

Please note this class basically supports modelling curves for "any purpose". One purpose might be the announcement of a device's planned energy consumption. Another purpose might be the presentation of a calculated or potential power availability. Esp. information from the *PowerSequence* class may provide further information on the use of slots and curves.

5.3.19.3 powerTimeSlotScheduleListData

5.3.19.3.1 General

This function can be used to model the schedule of a single slot. This information is intended to be used as actual information rather than schedule constraints (see 5.3.19.5). Together with a proper “write” classifier it can also be used to model the configuration of a slot schedule.

As already mentioned in 5.3.19.2 the "purpose" or applicability of a slot might depend on further information.

The function distinguishes between slots with "absolute" and "differential" schedule. The "absolute" schedule applies the element "timePeriod" (see below) and should be considered the most relevant information and the most common use. It can be used for curves with known or "already decided" time information, e.g. As timePeriod's sub-elements make use of the type AbsoluteOrRelativeTimeType it is possible to express all slot's time information with UTC or relative to "now" (or from a different point of view: "absolute" with "now" as reference time). A differential schedule can be used to model theoretical curves, e.g. The latter is useful for curves having no absolute schedule yet.

Of course, the slots of a curve shall not overlap in time. On the other hand, subsequent slots may well be separated by a gap in time. Mathematically spoken the slots of a sequence shall be disjoint and may be non-contiguous with regards to their schedule.

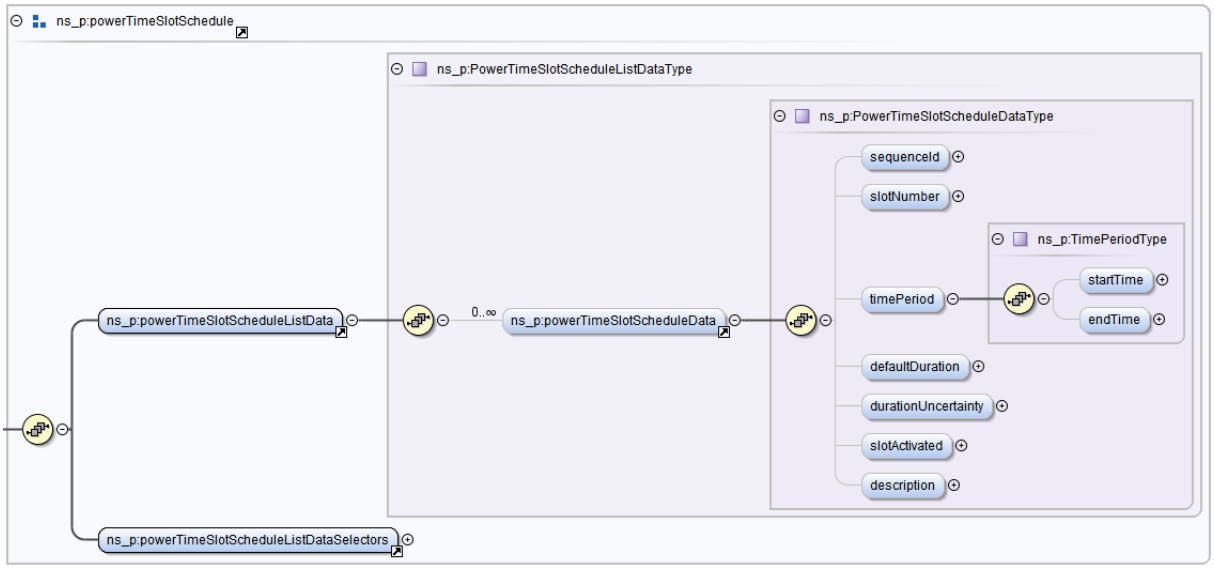


Figure 151: powerTimeSlotScheduleListData function overview

5.3.19.3.2 Detailed description of elements

Element	Type	Description
sequenceId	*Identifier "PowerSequenceIdType" (see* Table 339*).*	The sequence identifier.
slotNumber	*Identifier "PowerTimeSlotNumb*	The number of the slot.

	erType" (see Table 339*).*	
timePeriod	*Common data type "TimePeriodType". See section 3.10.2.1.*	Time information on the slot.
timePeriod.startTime	*See section 3.10.2.1.*	Start time of the slot.
timePeriod.endTime	*See section 3.10.2.1.*	End time of the slot.
defaultDuration	xs:duration (W3C standard type)	The duration of this slot.
durationUncertainty	xs:duration (W3C standard type)	The defaultDuration may have some uncertainty. The actual duration then may be in the range from *defaultDuration - durationUncertainty* to *defaultDuration + durationUncertainty*.
slotActivated	xs:boolean (W3C standard type)	This element permits modelling the state of an “optional slot”. Whether a slot is activated or not is specified here. The value “true” denotes the slot is used, wherease “false” means it is not used. See also element “optionalSlot” of function “powerTimeSlotScheduleConstraintsListData” (see Table 264).
description	*Common data type "DescriptionType". See section 3.10.1.3.*	Descriptive information on this slot.

Table 261: powerTimeSlotScheduleListData function element description

5.3.19.3.3 Available selectors

- sequenceId
- slotNumber

5.3.19.3.4 Examples

5.3.19.3.4.1 Read-reply

An energy management gateway was notified by a fridge about an upcoming task, referenced by sequence 2. The energy management gateway queries for powerTimeSlotScheduleListData schedule details on sequence 2 and receives a proper reply from the fridge:

```
<powerTimeSlotScheduleListData>
    <powerTimeSlotScheduleData>
        <sequenceId>2</sequenceId>
        <slotNumber>1</slotNumber>
        <timePeriod>
            <startTime>2006-05-04T16:15:00.0Z</startTime>
            <endTime>2006-05-04T16:22:00.0Z</endTime>
        </timePeriod>
        <defaultDuration>PT7M</defaultDuration>
```



```
            <durationUncertainty>PT0S</durationUncertainty>
            <slotActivated>true</slotActivated>
            <description>Super freeze</description>
        </powerTimeSlotScheduleData>
        <powerTimeSlotScheduleData>
            <sequenceId>2</sequenceId>
            <slotNumber>2</slotNumber>
            <timePeriod>
                <startTime>2006-05-04T16:25:00.0Z</startTime>
                <endTime>2006-05-04T16:29:30.0Z</endTime>
            </timePeriod>
            <defaultDuration>PT4M30S</defaultDuration>
            <durationUncertainty>PT5S</durationUncertainty>
            <slotActivated>true</slotActivated>
            <description>Freeze</description>
        </powerTimeSlotScheduleData>
</powerTimeSlotScheduleListData>
```

5.3.19.4 powerTimeSlotValueListData

5.3.19.4.1 General

This function can be used to model miscellaneous values associated to a single slot. The most essential values are certainly the "real" power and the "real" energy. In addition it is possible to model value facets like average, skewness, minimum, etc.

Please note the units of energy and power values are modelled in the PowerSequence sub-class.

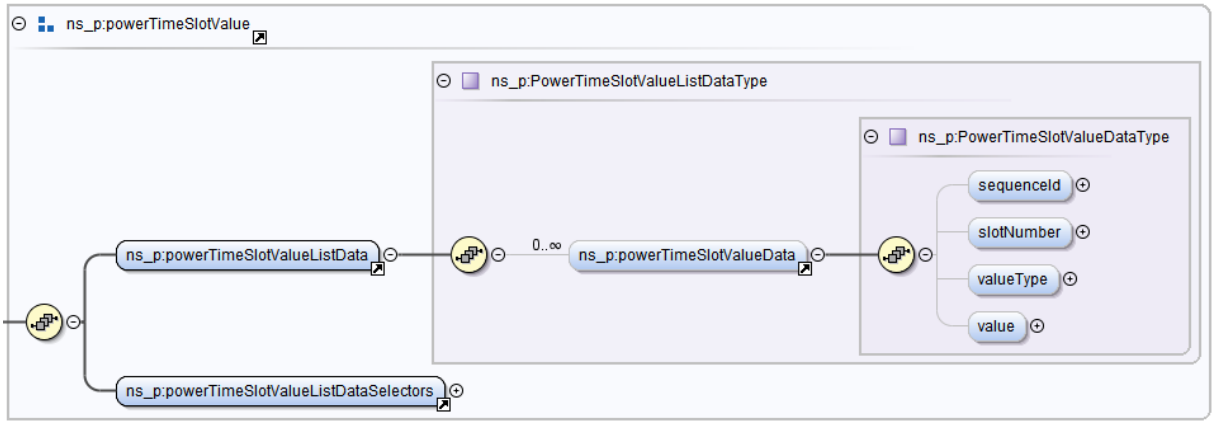


Figure 152: powerTimeSlotValueListData function overview

5.3.19.4.2 Detailed description of elements

Element	Type	Description
sequenceId	*Identifier "PowerSequenceIdType" (see* Table 339*).*	The sequence identifier.
slotNumber	*Identifier "PowerTimeSlotNumberType" (see* Table 339*).*	The number of the slot.
valueType	*Union "*PowerTimeSlotValueTypeType*":* *- Enum (see* Table 263*):* PowerTimeSlotValueTypeEnumType - EnumExtendType (see section 3.10.1.5)	The kind of the value ("power", "powerMin", "energy", ...).

value	*Common data type "ScaledNumberType". See section 3.10.1.8.*	The magnitude of the value denoted by valueType.

Table 262: powerTimeSlotValueListData function element description

Enumeration **PowerTimeSlotValueTypeEnumType**:

Value	Description
power	Power
powerMin	Lower bound of power; minimum
powerMax	Upper bound of power; maximum
powerExpectedValue	Expected value (or “expectation” in terms of probability theory) of power; may be used to express the estimation of an average value.
powerStandardDeviation	Standard deviation (in terms of probability theory) of power
powerSkewness	Skewness (in terms of probability theory) of power
energy	Energy
energyMin	Lower bound of energy; minimum
energyMax	Upper bound of energy; maximum
energyExpectedValue	Expected value (or “expectation” in terms of probability theory) of energy; may be used to express the estimation of an average value.
energyStandardDeviation	Standard deviation (in terms of probability theory) of energy
energySkewness	Skewness (in terms of probability theory) of energy

Table 263: Enumeration PowerTimeSlotValueTypeEnumType

5.3.19.4.3 Available selectors

- sequenceId
- slotNumber
- valueType

5.3.19.4.4 Examples

5.3.19.4.4.1 Read-reply

We continue with the example of section 5.3.19.3.4.1. The energy management gateway now queries for all values of sequence 2 and receives a proper reply from the fridge:

```
<powerTimeSlotValueListData>
    <powerTimeSlotValueData>
        <sequenceId>2</sequenceId>
        <slotNumber>1</slotNumber>
        <valueType>power</valueType>
        <value>
            <number>3</number>
            <scale>2</scale>
        </value>
    </powerTimeSlotValueData>
    <powerTimeSlotValueData>
        <sequenceId>2</sequenceId>
        <slotNumber>1</slotNumber>
        <valueType>powerMax</valueType>
        <value>
            <number>315</number>
        </value>
    </powerTimeSlotValueData>
```



```
        <powerTimeSlotValueData>
            <sequenceId>2</sequenceId>
            <slotNumber>1</slotNumber>
            <valueType>energyExpectedValue</valueType>
            <value>
                <number>5</number>
            </value>
        </powerTimeSlotValueData>
        <powerTimeSlotValueData>
            <sequenceId>2</sequenceId>
            <slotNumber>2</slotNumber>
            <valueType>power</valueType>
            <value>
                <number>12</number>
                <scale>1</scale>
            </value>
        </powerTimeSlotValueData>
        <powerTimeSlotValueData>
            <sequenceId>2</sequenceId>
            <slotNumber>2</slotNumber>
            <valueType>powerMax</valueType>
            <value>
                <number>14</number>
                <scale>1</scale>
            </value>
        </powerTimeSlotValueData>
        <powerTimeSlotValueData>
            <sequenceId>2</sequenceId>
            <slotNumber>2</slotNumber>
            <valueType>energyExpectedValue</valueType>
            <value>
                <number>2</number>
            </value>
        </powerTimeSlotValueData>
    </powerTimeSlotValueListData>
```

5.3.19.5 powerTimeSlotScheduleConstraintsListData

5.3.19.5.1 General

This function can be used to model constraints of a slot's schedule. It is useful in case a device supports the configuration or modification of a schedule. In this case the constraints can be used to determine the conditions a configuration needs to fulfil.

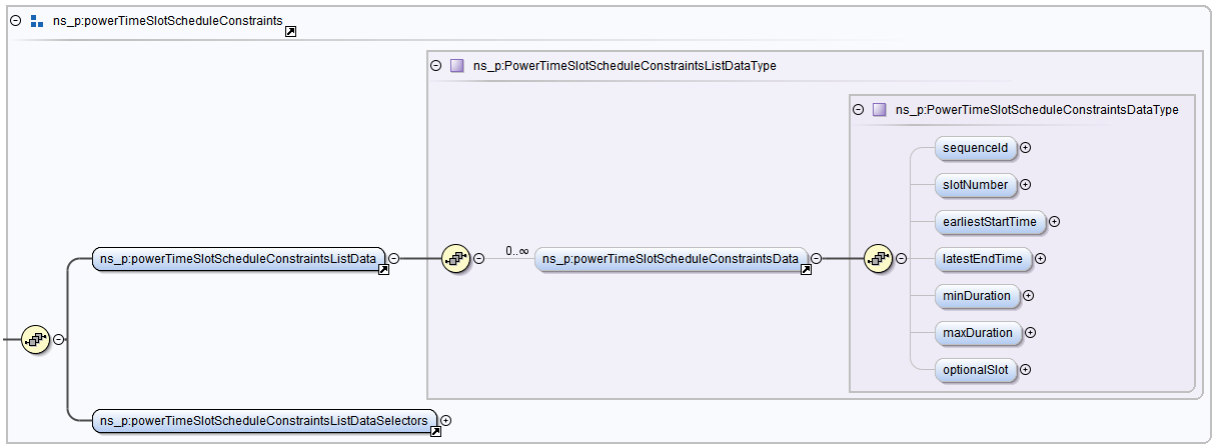


Figure 153: powerTimeSlotScheduleConstraintsListData function overview

5.3.19.5.2 Detailed description of elements

Element	Type	Description
sequenceId	*Identifier "PowerSequenceIdType" (see* Table 339*).*	The sequence identifier.
slotNumber	*Identifier "PowerTimeSlotNumberType" (see* Table 339*).*	The number of the slot.
earliestStartTime	*Common data type "AbsoluteOrRelativeTimeType". See section 3.10.2.3.*	The earliest start time of the slot (UTC or relative to "now").
latestEndTime	*Common data type "AbsoluteOrRelativeTimeType". See section 3.10.2.3.*	The end time of the slot has to be not later than this time (UTC or relative to "now").
minDuration	xs:duration (W3C standard type)	This is the minimum duration the slot can run.
maxDuration	xs:duration (W3C standard type)	This is the maximum duration the slot has to run.
optionalSlot	xs:boolean (W3C standard type)	A slot is optional if this element is set to “true”. See also element “slotActivated” of function “powerTimeSlotScheduleListData” (see Table 261) for the state of the slot (activated or deactivated).

Table 264: powerTimeSlotScheduleConstraintsListData function element description

5.3.19.5.3 Available selectors

- sequenceId
- slotNumber

5.3.19.5.4 Examples

5.3.19.5.4.1 Read-reply

We continue with the examples of section 5.3.19.3.4.1 and section 5.3.19.4.4. The energy management gateway wants to figure out whether some slots of sequence 2 can be shifted somehow. It sends a request for powerTimeSlotScheduleConstraintsListData of sequence 2 and receives a proper reply from the fridge:

```
<powerTimeSlotScheduleConstraintsListData>
    <powerTimeSlotScheduleConstraintsData>
        <sequenceId>2</sequenceId>
        <slotNumber>1</slotNumber>
        <earliestStartTime>2006-05-04T16:15:00.0Z</earliestStartTime>
        <latestEndTime>2006-05-04T18:00:00.0Z</latestEndTime>
        <minDuration>PT1H10M</minDuration>
        <maxDuration>PT1H30M</maxDuration>
        <optionalSlot>false</optionalSlot>
    </powerTimeSlotScheduleConstraintsData>
    <powerTimeSlotScheduleConstraintsData>
        <sequenceId>2</sequenceId>
        <slotNumber>2</slotNumber>
        <earliestStartTime>2006-05-04T19:00:00.0Z</earliestStartTime>
        <latestEndTime>2006-05-04T19:45:00.0Z</latestEndTime>
        <minDuration>PT15M</minDuration>
        <maxDuration>PT20M</maxDuration>
        <optionalSlot>false</optionalSlot>
    </powerTimeSlotScheduleConstraintsData>
</powerTimeSlotScheduleConstraintsListData>
```

5.3.19.6 Sub-class PowerSequence

The PowerSequence sub-class delivers all information about the power sequences themselves. This covers descriptions, states, constraints for scheduling, scheduling and price.

5.3.19.7 powerSequenceAlternativesRelationListData

5.3.19.7.1 General

As already described in section 5.3.19.1.3 for some use cases it is necessary to consider power sequences representing alternative tasks, i.e. tasks where at maximum one can finally be chosen. In order to model such a group of alternative sequences the powerSequenceAlternativesRelation can be used. It permits to model the relation

- 1 alternatives group has 0..m sequences

though practical use cases should have at least one sequence in an alternatives group.

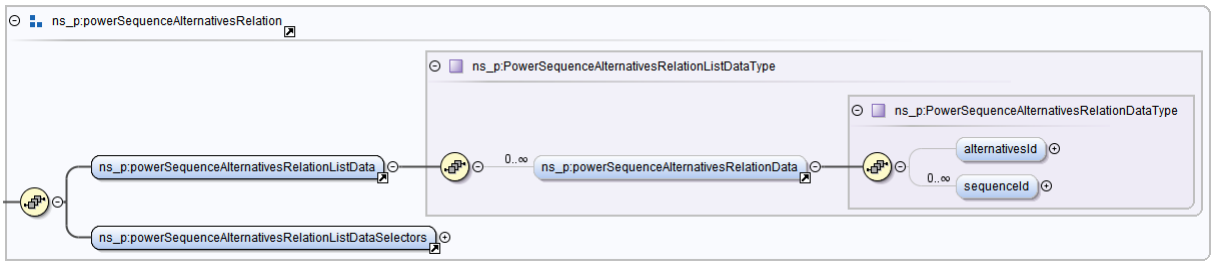


Figure 154: powerSequenceAlternativesRelationListData function overview

5.3.19.7.2 Detailed description of elements

Element	Type	Description
alternativesId	*Identifier "AlternativesIdType" (see* Table 339*).*	The identifier for the group of alternative sequences.
sequenceId (list)	*Identifier "PowerSequenceIdType" (see* Table 339*).*	A sequence identifier that belongs to the given alternatives group.

Table 265: powerSequenceAlternativesRelationListData function element description

5.3.19.7.3 Available selectors

- alternativesId
- sequenceId

5.3.19.7.4 Examples

5.3.19.7.4.1 Notify

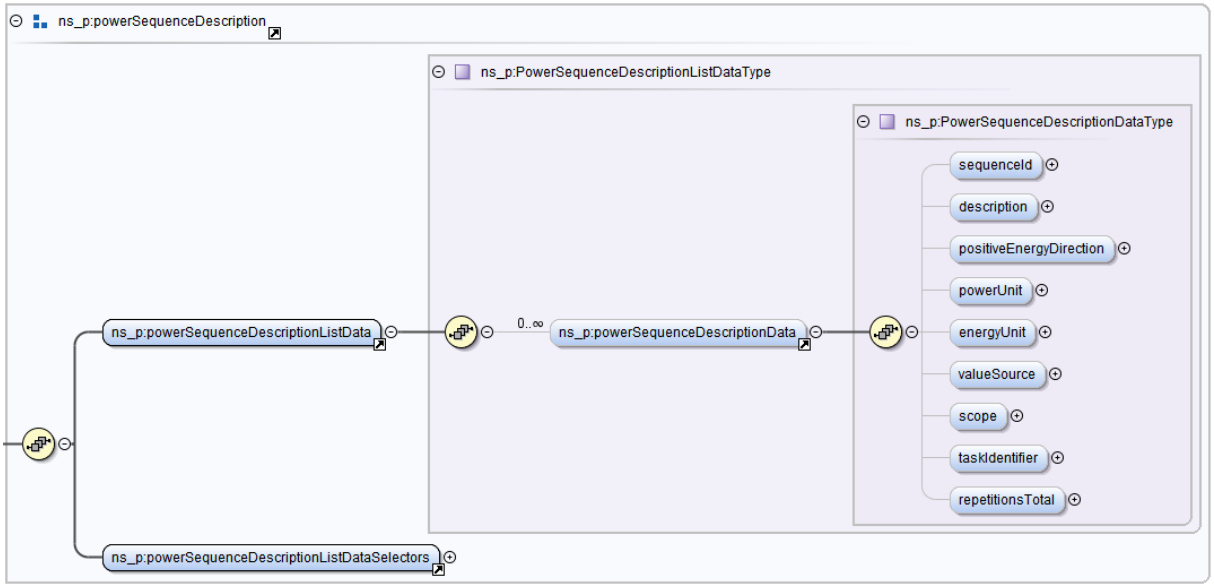
A heating system announces some proposals for the afternoon and the evening: For the afternoon group 1 contains sequences 1 and 2 to choose from and for the evening group 2 permits to select sequence 3, 4, or 5. It sends this (complete) notification:

```
<powerSequenceAlternativesRelationListData>
    <powerSequenceAlternativesRelationData>
        <alternativesId>1</alternativesId>
        <sequenceId>1</sequenceId>
        <sequenceId>2</sequenceId>
    </powerSequenceAlternativesRelationData>
    <powerSequenceAlternativesRelationData>
        <alternativesId>2</alternativesId>
        <sequenceId>3</sequenceId>
        <sequenceId>4</sequenceId>
        <sequenceId>5</sequenceId>
    </powerSequenceAlternativesRelationData>
</powerSequenceAlternativesRelationListData>
```


9007 ***5.3.19.8 powerSequenceDescriptionListData***

9008 *5.3.19.8.1 General*

9009 The *powerSequenceDescriptionData* function models descriptive (typically non-changing) information
9010 on sequences (one for each available sequence of a node).



9011

9012 *Figure 155: powerSequenceDescriptionListData function overview*

9013

9014 *5.3.19.8.2 Detailed description of elements*

Element	Type	Description
sequenceId	*Identifier "PowerSequenceIdType" (see* Table 339*).*	The sequence identifier.
description	*Common data type "DescriptionType". See section 3.10.1.3.*	Descriptive information on this sequence.
positiveEnergyDirection	*Common data type "EnergyDirectionType". See section 3.10.1.13.*	The *positiveEnergyDirection* states whether energy consumption or production will be counted as positive value: If the element is set to “consume”: energy values are positive during consumption and negative during production of energy. If the element is set to “produce”: energy values are negative during consumption and positive during production of energy.
powerUnit	*Common data type "UnitOfMeasurementType". See section 3.10.1.17.*	This is the unit for all values related to power.
energyUnit	*Common data type "UnitOfMeasurementType". See section 3.10.1.17.*	This is the unit for all values concerning energy consumption and production.
valueSource	*Union* "MeasurementValueSourceType*":*	This element enables to express whether the values of the given sequence are based upon measurement or calculation, e.g.

	- *Enum (see* Table 232*):* MeasurementValueSourceEnumType - EnumExtendType (see section 3.10.1.5)	
scope	*Union "*PowerSequenceScopeType*":* - *Enum (see* Table 267*):* PowerSequenceScopeEnumType - EnumExtendType (see section 3.10.1.5)	With this element it is possible to distinguish between sequences intended for different scopes (forecast, measurement, etc.). Note: Sequences with scope "forecast" or "measurement" typically originate from a device executing a task, whereas sequences with scope "recommendation" rather denote a configuration into a device that shall execute a task.
taskIdentifier	xs:unsignedInt (W3C standard type)	This "technical identifier" denotes which kind of task or condition applies to the power sequence. A device can provide identifiers in order to enable energy management applications recognize common (i.e. "typical" or frequent) tasks of a device. Note: This element should not be confused with “taskId” of the class TaskManagement.
repetitionsTotal	xs:unsignedInt (W3C standard type)	If the sequence applies the “repetition concept”, this element denotes the total number of executions of the sequence. See also element “activeRepetitionNumber” of function “powerSequenceStateData”.

Table 266: powerSequenceDescriptionListData function element description

Enumeration **PowerSequenceScopeEnumType**:

Value	**Description**
forecast	The sequence's values reflect the best guess of a device's task which typically begins in the future.
measurement	The sequence's values reflect the measurement of a device's task.
recommendation	The value "recommendation" can be used to model a "target power curve", i.e. a power profile a device SHOULD follow for a given task (if possible).

Table 267: Enumeration PowerSequenceScopeEnumType

5.3.19.8.3 Available selectors

- sequenceId

5.3.19.8.4 Examples

5.3.19.8.4.1 Read-reply

One is interested in the description of all available power sequences. A query for a device's powerSequenceDescriptionListData may lead to the following reply:

```
<powerSequenceDescriptionListData>
    <powerSequenceDescriptionData>
        <sequenceId>1</sequenceId>
        <description>Washing programme 1</description>
        <positiveEnergyDirection>consume</positiveEnergyDirection>
        <powerUnit>W</powerUnit>
        <energyUnit>Wh</energyUnit>
        <scope>forecast</scope>
        <taskIdentifier>7</taskIdentifier>
    </powerSequenceDescriptionData>
    <powerSequenceDescriptionData>
        <sequenceId>2</sequenceId>
        <description>Washing programme 2</description>
        <positiveEnergyDirection>consume</positiveEnergyDirection>
        <powerUnit>W</powerUnit>
        <energyUnit>Wh</energyUnit>
        <scope>forecast</scope>
        <taskIdentifier>8</taskIdentifier>
    </powerSequenceDescriptionData>
</powerSequenceDescriptionListData>
```

5.3.19.9 powerSequenceStateListData

5.3.19.9.1 General

Information on the current state of a power sequence can be modelled with the *powerSequenceStateData* function.

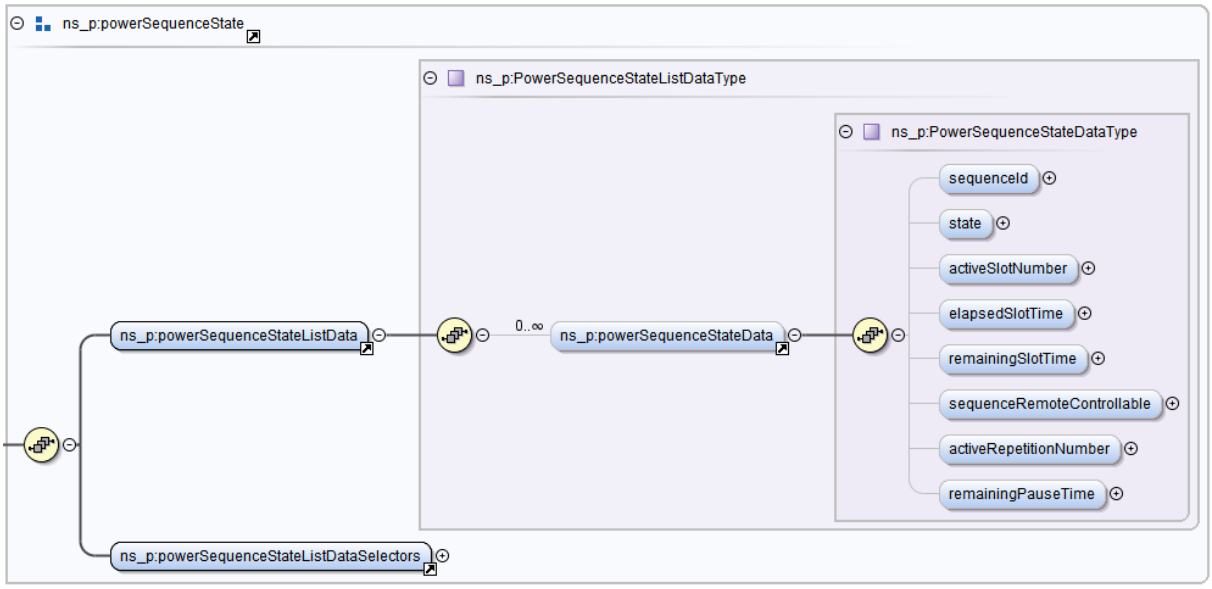


Figure 156: powerSequenceStateListData function overview

9054 *5.3.19.9.2 Detailed description of elements*

Element	Type	Description
sequenceId	*Identifier "PowerSequenceIdType" (see* Table 339*).*	The sequence identifier.
state	*Union "*PowerSequenceStateType*":* *- Enum (see* Table 269*):* PowerSequenceStateEnumType - EnumExtendType (see section 3.10.1.5)	The current state of the power sequence.
activeSlotNumber	*Identifier "PowerTimeSlotNumberType" (see* Table 339*).*	Contains the currently active slot.
elapsedSlotTime	xs:duration (W3C standard type)	Contains the time the slot has already been in “running” state (this also means the value remains constant during a “paused” state).
remainingSlotTime	xs:duration (W3C standard type)	Contains the time the slot still needs to be in “running” state (this also means the value remains constant during a “paused” state).
sequenceRemoteControllable	xs:boolean (W3C standard type)	If this element is set to “false” the sequence is executed autonomously and cannot be modified by SPINE commands received external nodes. Note: Consider also the element “nodeRemoteControllable” of function “powerSequenceNodeScheduleInformationData” in Table 274!
activeRepetitionNumber	xs:unsignedInt (W3C standard type)	If the sequence applies the “repetition concept”, this element denotes the current number of the sequence execution.
remainingPauseTime	xs:duration (W3C standard type)	Denotes the duration the current slot (element “activeSlotNumber”) permits being paused.

9055 *Table 268: powerSequenceStateListData function element description*

9056 Enumeration **PowerSequenceStateEnumType**:

Value	Description
running	The sequence is currently executed.
paused	The sequence is currently paused.
scheduled	The sequence is not executed at the moment but will be executed (i.e. gain state “running”) by the power sequences server at the scheduled time.
scheduledPaused	Similar to the value “scheduled”, but at the scheduled time the sequence will be set into the state “paused”.
pending	The sequence is “recommended” (in terms of “preferred”) by the power sequences server but will not be executed autonomously by the server. Remark: This state is most appropriate for devices announcing that a specific task (power sequence) is going to become "valid" but still lacks something / requires configuration or advice. It is most suitable in conjunction with the function powerSequenceScheduleConfigurationRequestCall.

inactive	The sequence denotes an “optional task”. I.e. it is not invalid but it is also not running/scheduled/... so far. However, this sequence may become running or scheduled(Paused) by appropriate means.
completed	The sequence has been completed.
invalid	The sequence is invalid and shall not be considered anymore.

Table 269: Enumeration PowerSequenceStateEnumType

5.3.19.9.3 Available selectors

- sequenceId

5.3.19.9.4 Examples

5.3.19.9.4.1 Notify

A power sequences server executes sequence “2” and just completed slot number 1. Now, it begins wih slot number 2 and notifies this change (the “filter/partial” part to announce the “restricted function exchange is simplified by “...” just to improve the readability”):

```
...
<powerSequenceStateListData>
    <powerSequenceStateData>
        <sequenceId>2</sequenceId>
        <activeSlotNumber>2</activeSlotNumber>
        <elapsedSlotTime>PT0M</elapsedSlotTime>
        <remainingSlotTime>PT20M</remainingSlotTime>
    </powerSequenceStateData>
</powerSequenceStateListData>
```

5.3.19.10 powerSequenceScheduleListData

5.3.19.10.1 General

The scheduling of a sequence can be modelled with this function.

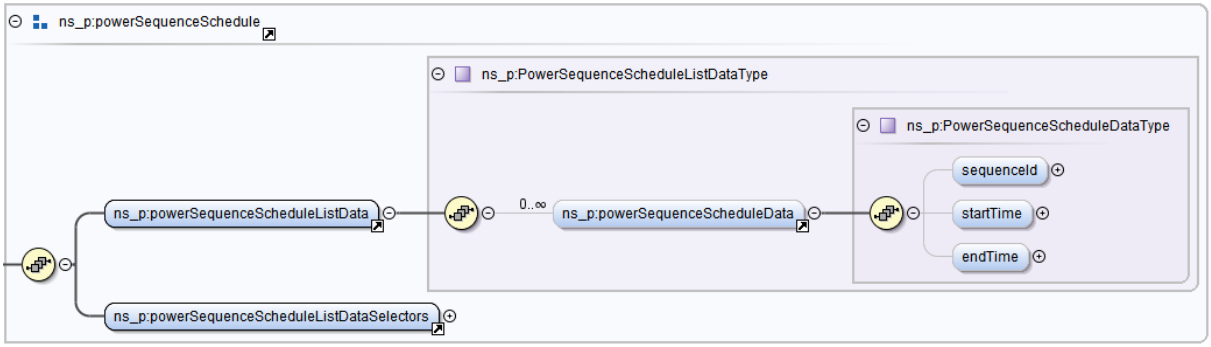


Figure 157: powerSequenceScheduleListData function overview

5.3.19.10.2 Detailed description of elements

Element	Type	Description
sequenceId	*Identifier "PowerSequenceIdType" (see* Table 339*).*	The sequence identifier.

startTime	*Common data type "AbsoluteOrRelativeTimeType". See section 3.10.2.3.*	The start time of the sequence.
endTime	*Common data type "AbsoluteOrRelativeTimeType". See section 3.10.2.3.*	The end time of the sequence.

Table 270: powerSequenceScheduleListData function element description

5.3.19.10.3 Available selectors

- sequenceId

5.3.19.10.4 Examples

5.3.19.10.4.1 Read-reply

An Energy Management Software wants to get information on the current schedule of a specific sequence (2). It sends a proper request and receives this response:

```
<powerSequenceScheduleListData>
    <powerSequenceScheduleData>
        <sequenceId>2</sequenceId>
        <startTime>2013-07-13T18:11:00.0Z</startTime>
        <endTime>2013-07-14T16:55:00.0Z</endTime>
    </powerSequenceScheduleData>
</powerSequenceScheduleListData>
```

5.3.19.11 powerSequenceScheduleConstraintsListData

5.3.19.11.1 General

A power sequence of an appliance may have time-related constraints that are to be considered by an energy management software during demand optimization calculations. Otherwise the appliance could reject a schedule change requested by the energy management software.

This section deals with overall schedule constraints of a sequence. Schedule constraints of slots are discussed in section 5.3.19.5. An application may make use of both kinds of constraints together. Of course, contradictory information from constraints and schedules should be avoided.

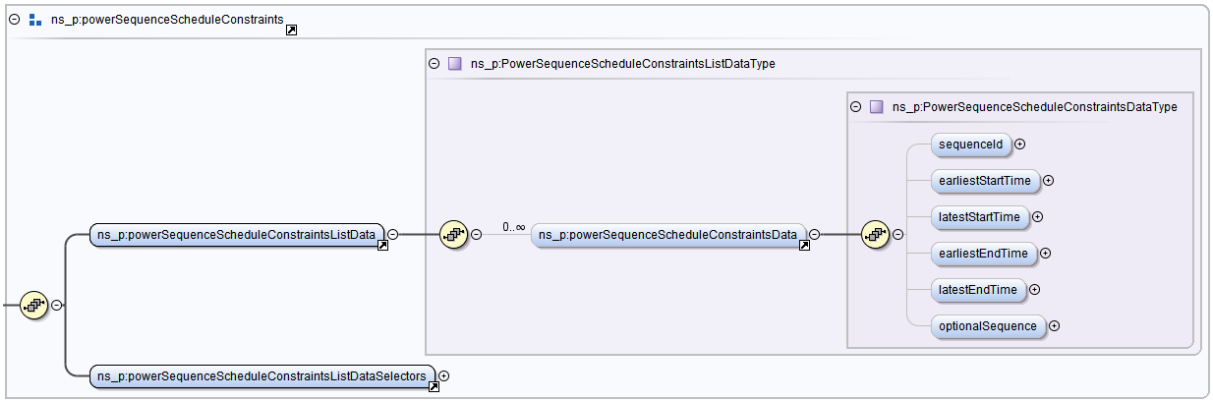


Figure 158: powerSequenceScheduleConstraintsListData function overview

5.3.19.11.2 Detailed description of elements

Element	Type	Description
sequenceId	*Identifier "PowerSequenceIdType" (see* Table 339*).*	The sequence identifier.
earliestStartTime	*Common data type "AbsoluteOrRelativeTimeType". See section 3.10.2.3.*	The earliest time the sequence can start.
latestStartTime	*Common data type "AbsoluteOrRelativeTimeType". See section 3.10.2.3.*	The latest time the sequence can start.
earliestEndTime	*Common data type "AbsoluteOrRelativeTimeType". See section 3.10.2.3.*	The earliest time the sequence can end.
latestEndTime	*Common data type "AbsoluteOrRelativeTimeType". See section 3.10.2.3.*	The latest time the sequence can end.
optionalSequence	xs:boolean (W3C standard type)	For specific use cases a sequence can be tagged explicitely as “optional task”. In this case this element can be set to *true*.

Table 271: powerSequenceScheduleConstraintsListData function element description

Please note that start and end time constraints just model start and end boundaries of a power sequence. I.e. in general the duration of a sequence can be shorter than the period given by the start/end time constraints.

5.3.19.11.3 Available selectors

- sequenceId

5.3.19.11.4 Examples

5.3.19.11.4.1 Read-reply

Before an Energy Management Software re-schedules a specific sequence (2) of an appliance it queries the appliance for the proper constraints. The appliance sends this response:

```
<powerSequenceScheduleConstraintsListData>
    <powerSequenceScheduleConstraintsData>
        <sequenceId>2</sequenceId>
        <earliestStartTime>2013-07-14T12:00:00.0Z</earliestStartTime>
        <latestStartTime>2013-07-14T12:45:00.0Z</latestStartTime>
        <earliestEndTime>2013-07-14T16:00:00.0Z</earliestEndTime>
        <latestEndTime>2013-07-14T18:00:00.0Z</latestEndTime>
    </powerSequenceScheduleConstraintsData>
</powerSequenceScheduleConstraintsListData>
```


5.3.19.12 powerSequencePriceListData

5.3.19.12.1 General

The execution of a power sequence may cause some cost or earnings due to the consumption or production of energy. This means a power sequence can be assigned a price. In fact, as the availability of energy may vary with time, the price of a power sequence could as well depend upon its time of execution (which may esp. be of interest if the scheduling of power sequences shall consider cost saving requirements). Therefore, the *powerSequencePriceData* function permits modelling a price per assumed start time of the sequence. Please note assumed start times may differ from the "final" start time.

How an appliance gains knowledge of the price associated to one of its sequences shall not be discussed in detail. Just a few possibilities shall be sketched briefly: Configuration (the sequence has a fixed price which is stored in the appliance), local calculation (the appliance obtains price information and applies it to the power consumption), or query to an external node (see *powerSequencePriceCalculationRequestCall* function in section 5.3.19.18).

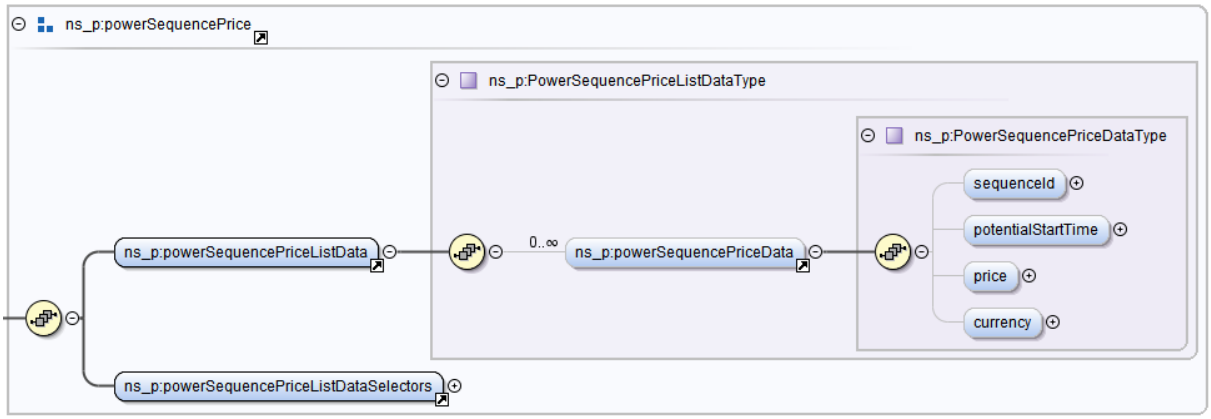


Figure 159: powerSequencePriceListData function overview

5.3.19.12.2 Detailed description of elements

Element	Type	Description
sequenceId	*Identifier "PowerSequenceIdType" (see* Table 339*).*	The sequence identifier.
potentialStartTime	*Common data type "AbsoluteOrRelativeTime Type". See section 3.10.2.3.*	The price information belongs to the start time given with this element. Please note this can differ from the final start time of a sequence (in case the sequence has been shifted in time but the price has not been updated accordingly, e.g.). Of course, in this case the price denoted in this instance can differ from the "real" price.
price	*Common data type "ScaledNumberType". See section 3.10.1.8.*	The price of the sequence for the given potential start time.
currency	*Common data type "CurrencyType". See section 3.10.1.19.*	The currency of the price.

Table 272: powerSequencePriceListData function element description

5.3.19.12.3 Available selectors

- sequenceId
- potentialStartTimeInterval

5.3.19.12.4 Examples

5.3.19.12.4.1 Write

An energy management system got information about power sequence 2 of an appliance. The energy management system has also knowledge of the energy price and can calculate a proper cost. It submits the calculated price for a potential start time to the appliance and the appliance can then display the cost to the user:

```
<powerSequencePriceListData>
    <powerSequencePriceData>
        <sequenceId>2</sequenceId>
        <potentialStartTime>2013-07-14T12:35:00.0Z</potentialStartTime>
        <price>
            <number>95</number>
            <scale>-2</scale>
        </price>
        <currency>EUR</currency>
    </powerSequencePriceData>
</powerSequencePriceListData>
```

5.3.19.13 powerSequenceSchedulePreferenceListData

5.3.19.13.1 General

A “smart energy management” typically includes the consideration of a user’s preference for the kind of energy consumption optimisation. Some may prefer the optimisation to reduce the cost whereas others may prefer first of all the use of “green energy”. This kind of preference can be modelled using *powerSequenceSchedulePreference*, where the preference can be assigned per sequence.

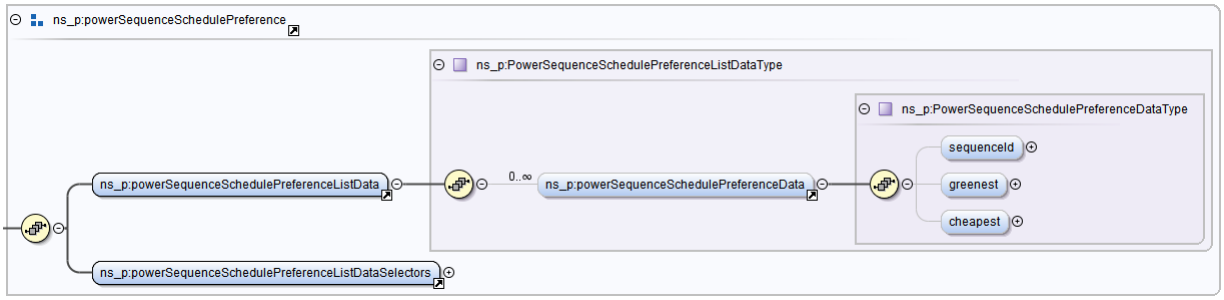


Figure 160: powerSequenceSchedulePreferenceListData function overview

5.3.19.13.2 Detailed description of elements

Element	Type	Description

sequenceId	*Identifier "PowerSequenceIdType" (see* Table 339*).*	The sequence identifier.
greenest	xs:boolean (W3C standard type)	For this sequence an optimisation towards the minimum usage of NON-renewable energy is preferred.
cheapest	xs:boolean (W3C standard type)	For this sequence an optimisation that minimises the user's energy bill is preferred.

Table 273: powerSequenceSchedulePreferenceListData function element description

5.3.19.13.3 Available selectors

- sequenceId

5.3.19.13.4 Examples

5.3.19.13.4.1 Read-reply

A "smart washing machine" is connected with an energy management system. At the washing machine a user can press a button to enable the preference for renewable energy. Once the washing machine announced an upcoming task, the energy management system queries for the user's preference and receives this response:

```
<powerSequenceSchedulePreferenceListData>
    <powerSequenceSchedulePreferenceData>
        <sequenceId>3</sequenceId>
        <greenest>true</greenest>
    </powerSequenceSchedulePreferenceData>
</powerSequenceSchedulePreferenceListData>
```

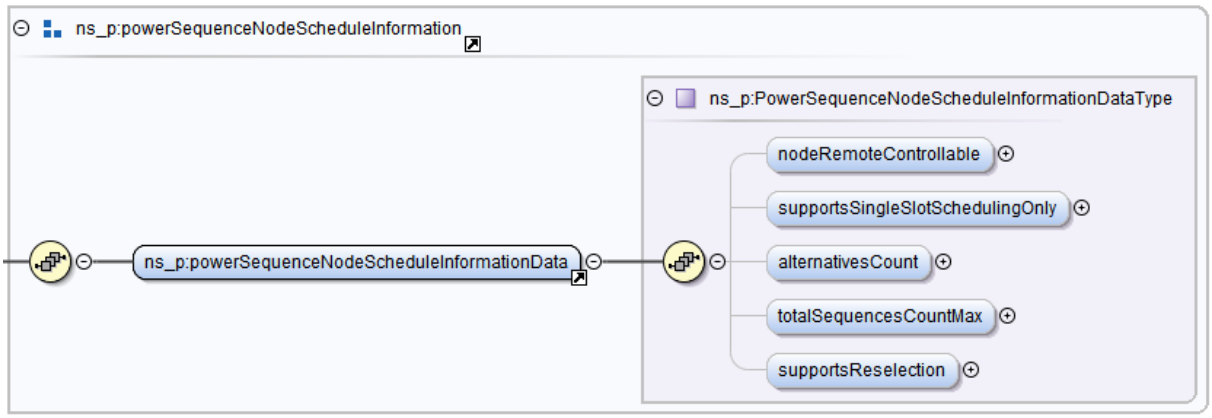
5.3.19.14 Sub-class PowerSequenceNode

This sub-class includes PowerSequences information affecting the whole node rather than a specific sequence.

5.3.19.15 powerSequenceNodeScheduleInformationData

5.3.19.15.1 General

The *powerSequenceNodeScheduleInformationData* function permits modelling of some schedule information that applies to a whole node rather than to a specific sequence. This information can change during runtime (especially the *nodeRemoteControllable* element).



9214

9215 *Figure 161: powerSequenceNodeScheduleInformationData function overview*

9216

9217 *5.3.19.15.2 Detailed description of elements*

Element	Type	Description
nodeRemoteControllable	xs:boolean (W3C standard type)	This element denotes whether a node permits being scheduled (value is “true”) by a remote device or not (value is “false”).
supportsSingleSlotSchedulingOnly	xs:boolean (W3C standard type)	If set to “true” the appliance does NOT permit the modification of more than one slot per configuration command: The sub-class PowerTimeSlot provides functions that can be used to configure slots of an appliance. Multiple slots may be configured with a single command (single function instance). However, some kind of appliances may support the scheduling of only a single slot per sequence as configuration request. This constraint can be modelled setting this element to *true*. In this case for each slot a separate configuration request (write command with the function properly set) for scheduling should be used.
alternativesCount	xs: unsignedInt (W3C standard type)	The total number of “alternatives groups” (see sections 5.3.19.1.3, 5.3.19.7). If powerSequenceAlternativesRelationListData is used, the element alternativesCount denotes the list size.
totalSequencesCountMax	xs:unsignedInt (W3C standard type)	The maximum total number of sequences the device may offer at any given moment. This information is primarily intended for energy management systems to get a notion of an appliance’s schedule complexity. If an appliance counts all of its currently known sequences (i.e. across all alternatives groups and also regardless of the sequences’ states) the result will not exceed totalSequencesCountMax.
supportsReselection	xs:boolean (W3C standard type)	As mentioned in section 5.3.19.1.3 an appliance may permit an energy management system to select one of its sequences from an alternatives group for execution. Some appliances may permit such a selection only one time per alternatives group. This

		can be expressed with “supportsReselection” setting to “false”. If an appliance grants the energy management system to alter the selection (i.e. re-select) of the alternatives group, this can be expressed with the value “true”.

Table 274: powerSequenceNodeScheduleInformationData function element description

5.3.19.15.3 Available selectors

None.

5.3.19.15.4 Examples

5.3.19.15.4.1 Notify

An Energy Management Software wants to get an overview on the current state of a washing machine, esp. in order to see whether it can be scheduled. It sends a requests and gets this response:

```
<powerSequenceNodeScheduleInformationData>
    <nodeRemoteControllable>true</nodeRemoteControllable>

<supportsSingleSlotSchedulingOnly>true</supportsSingleSlotSchedulingOnly>
    <alternativesCount>1</alternativesCount>
    <totalSequencesCountMax>3</totalSequencesCountMax>
    <supportsReselection>false</supportsReselection>
</powerSequenceNodeScheduleInformationData>
```

5.3.19.16 PowerSequence calls

Subsequently, the calls available in the PowerSequence Class are described.

5.3.19.17 powerSequenceScheduleConfigurationRequestCall

5.3.19.17.1 General

Some functions of the sub-class PowerSequence can be used to model notification of new or changed sequences. Energy management gateways may listen to such notifications and take action on demand. However, the nature of notifications does not imply any mandatory action upon receipt. With the function *powerSequenceScheduleConfigurationRequestCall* the owner of a sequence (i.e. a server with regards to the PowerSequences class) can model an explicit request stating a sequence shall be configured. I.e. an Energy Management Software (i.e. a client with regards to the PowerSequences class) receiving this function would know it needs to take proper action (provided it is capable of such action, i.e. performing write operations with sequence configuration).

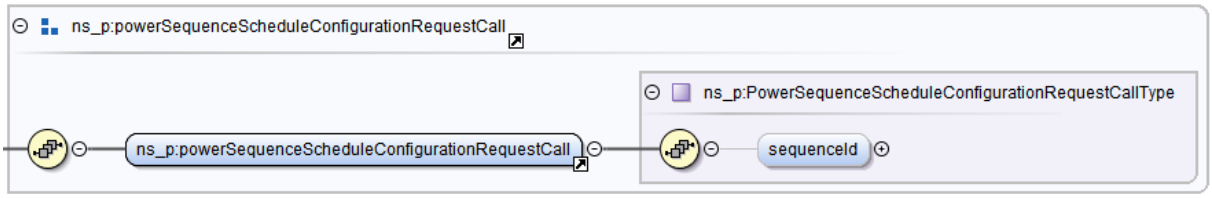


Figure 162: powerSequenceScheduleConfigurationRequestCall function overview

5.3.19.17.2 Detailed description of elements

Element	Type	Description
sequenceId	*Identifier "PowerSequenceIdType" (see* Table 339*).*	The sequence identifier the owner of the power sequence wants to be configured.

Table 275: powerSequenceScheduleConfigurationRequestCall function element description

5.3.19.17.3 Available selectors

None.

5.3.19.17.4 Examples

5.3.19.17.4.1 Call

If a node wants one of its power sequences to be configured, it could send the following XML with a *call* classifier to an appropriate Energy Management Software:

```
<powerSequenceScheduleConfigurationRequestCall>
    <sequenceId>2</sequenceId>
</powerSequenceScheduleConfigurationRequestCall>
```

5.3.19.18 powerSequencePriceCalculationRequestCall

5.3.19.18.1 General

The owner of a power sequence can use this function in order to trigger the determination of a price for one of its sequences at an external node. This shall be explained in more detail now.

Section 5.3.19.12.1 describes how the owner of a sequence can express a price for a given sequence and start time. Before the owner is able to tell a price upon request it needs to get knowledge of the price by itself. The section also briefly mentions different possibilities to achieve this. One of these possibilities is the use of the *powerSequencePriceCalculationRequestCall* function: It requires a sequence is available that expresses its energy or power demand. It also requires the owner of the sequence knows of an external node that has the capability to determine a price for the owner's sequence. Then, the owner of the sequence can submit a powerSequencePriceCalculationResponseCall function to this external node in order to trigger the determination of the price. This function can take the sequenceId and also a potential start time.

Finally, the external node may use the function *powerSequencePriceData* (see section 5.3.19.12) in order to return the calculated price to the owner of the sequence.

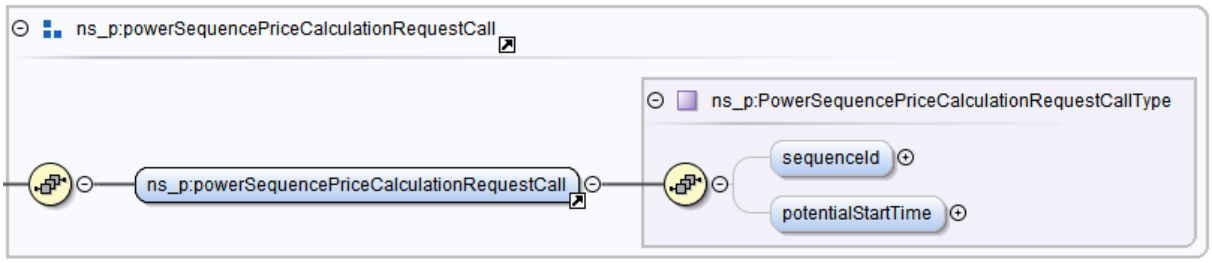


Figure 163: powerSequencePriceCalculationRequestCall function overview

5.3.19.18.2 Detailed description of elements

Element	Type	Description
sequenceId	*Identifier "PowerSequenceIdType" (see* Table 339*).*	The sequence identifier the owner of the power sequence wants to get a price for.
potentialStartTime	*Common data type "AbsoluteOrRelativeTimeType". See section 3.10.2.3.*	The price shall be determined for a potential start time given in this element.

Table 276: powerSequencePriceCalculationRequestCall function element description

5.3.19.18.3 Available selectors

None.

5.3.19.18.4 Examples

5.3.19.18.4.1 Call

The scenario assumes an external node (an energy management system, e.g.) that is capable of calculating prices based upon a given energy demand and schedule and also has knowledge of the tariff that is relevant for the required energy. A washing machine prepared a power sequence and wants to get a price for this sequence with a potential start time. It sends this XML with a *call* classifier to the energy management system:

```
<powerSequencePriceCalculationRequestCall>
    <sequenceId>2</sequenceId>
    <potentialStartTime>2013-07-14T12:35:00.0Z</potentialStartTime>
</powerSequencePriceCalculationRequestCall>
```


5.3.20 Sensing

5.3.20.1 Introduction

The *Sensing* class is used for modelling data obtained by sensors in the meaning of more state-based sensors like contact sensors, switches and buttons, etc., in contrast to value-based sensors like temperature or humidity sensors that are treated with the *Measurement* class.

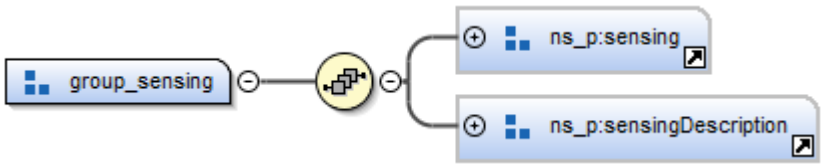


Figure 164: Sensing function-group overview

5.3.20.2 sensingListData

5.3.20.2.1 General

The list function of sensing can be used to model sensor information over time. This may be of interest for a motion detector (when someone was at the backdoor), a contact sensor (when was the window open in the last 24 hours), etc.

Some sensors may only store the latest value and therefore only support one entry in the list. To enable users and applications to query past changes, a communications technology interface may cache states/values of simple sensors that only support the single data function. That is no mandatory feature, however.

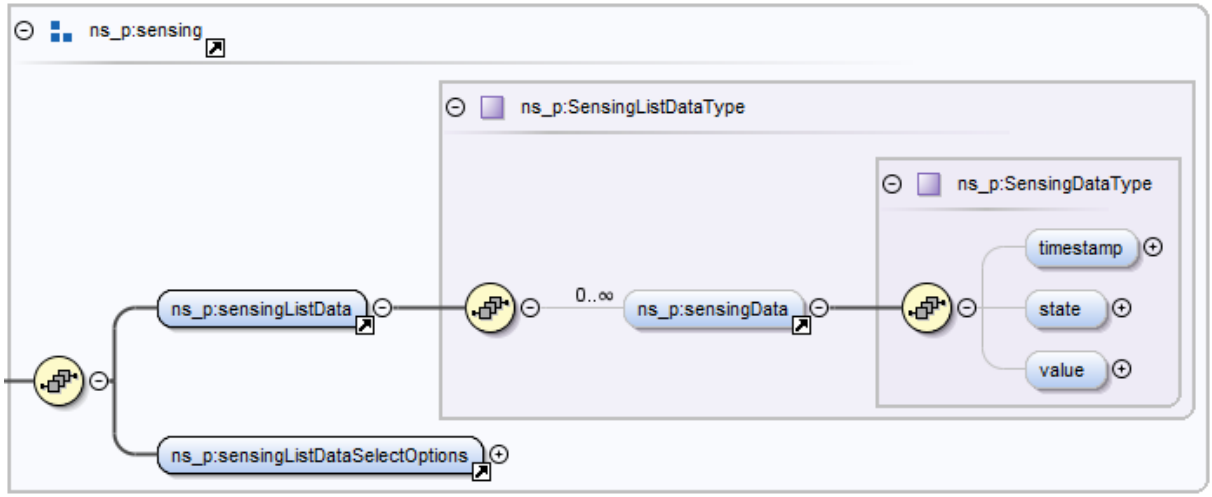


Figure 165: sensingListData function overview

5.3.20.2.2 Detailed description of elements

Element	Type	Description
timestamp	*Common data type "AbsoluteOrRelativeTimeType". See section 3.10.2.3.*	The creation time of the *sensingData* instance.

state	*Union "*SensingStateType*":* *- Enum (see* Table 278*):* SensingStateEnumType - EnumExtendType (see section 3.10.1.5)	The state of the sensor is denoted in this element.
value	*Common data type "ScaledNumberType". See section 3.10.1.8.*	Some sensors deliver a value (e.g. a *levelSwitch*).

9323 *Table 277: sensingListData function detailed description of elements*

9324 Enumeration **SensingStateEnumType**:

Value	Description
on	"on" was detected, meaning that the according functionality was switched on.
off	"off" was detected, meaning that the according functionality was switched off.
toggle	"toggle" was detected, meaning that the according functionality was toggled (e.g. from "on" to "off" or vice versa).
level	"level" was detected, meaning that the according functionality changed its level (e.g. from 2.4 to 8.2) that can be found in the Element "value".
levelUp	"levelUp" was detected, meaning that the according functionality changed its level upwards.
levelDown	"levelDown" was detected, meaning that the according functionality changed its level downwards.
levelStart	"levelStart" was detected, meaning that the according functionality started to change its level.
levelStop	"levelStop" was detected, meaning that the according functionality stopped to change its level.
levelAbsolute	"levelAbsolute" was detected, meaning that the according functionality changed its level to an absolute value stated in the Element "value" (e.g. 1190).
levelRelative	"levelRelative" was detected, meaning that the according functionality moved its level in comparison to the old one. The relative change can be found in the Element "value" (e.g. a change from 22 to 33 would result in a content of the Element "value" of 11).
levelPercentageAbsolute	"levelPercentageAbsolute" was detected, meaning that the according functionality changed its level to an absolute percentage value stated in the Element "value" (e.g. 90%).
levelPercentageRelative	"levelPercentageRelative" was detected, meaning that the according functionality changed its percentage level relative to the old one. The relative changed can be found in the Element "value" (e.g. a change from 40% to 50% would result in a content of the Element "value" of 10).
pressed	"pressed" was detected, meaning that the according functionality was pressed (e.g. a button, that was pressed).
longPressed	"longPressed" was detected, meaning that the according functionality was pressed longer (e.g. a button, that was pressed longer). The timing difference between "pressed" and "longPressed" is up to the vendor.
released	"released" was detected, meaning that the according functionality was released (e.g. a button, that was released).
changed	"changed" was detected, meaning that the according functionality changed.

started	"started" was detected, meaning that the according functionality started.
stopped	"stopped" was detected, meaning that the according functionality stopped.
paused	"paused" was detected, meaning that the according functionality paused.
middle	"middle" was detected, meaning that the according functionality (e.g. a scroll bar or a throttle) is in position "middle".
up	"up" was detected, meaning that the according functionality moves up.
down	"down" was detected, meaning that the according functionality moves down.
forward	"forward" was detected, meaning that the according functionality moves forward.
backwards	"backwards" was detected, meaning that the according functionality moves backwards.
open	"open" was detected, meaning that the according functionality is open (e.g. a door).
closed	"closed" was detected, meaning that the according functionality is closed (e.g. a door).
opening	"opening" was detected, meaning that the according functionality is opening (e.g. a door, that is changing from "closed" to "open").
closing	"closing" was detected, meaning that the according functionality is closing (e.g. a door, that is changing from "open" to "closed").
high	"high" was detected, meaning that the according functionality is at a high position or value.
low	"low" was detected, meaning that the according functionality is at a low position or value.
day	"day" was detected, meaning that the according functionality is in day-mode.
night	"night" was detected, meaning that the according functionality is in night-mode.
detected	"detected" was detected, meaning that the according functionality was detected (e.g. a motion sensor detected a motion).
notDetected	"notDetected" was detected, meaning that the according functionality was detected (e.g. a motion sensor reports that no motion is detected).
alarmed	"alarmed" was detected, meaning that the according functionality is in alarm mode (e.g. a smoke detector).
notAlarmed	"notAlarmed" was detected, meaning that the according functionality is not in alarm mode (e.g. a smoke detector).

Table 278: Enumeration SensingStateEnumType

5.3.20.2.3 Available selectors

- timestampInterval

5.3.20.2.4 Examples

5.3.20.2.4.1 Notify

If a contact sensor detects an opening (e.g. of a window), it will send a (complete) notification about that occurrence:

```
<sensingListData>
    <sensingData>
        <timestamp>2013-07-15T12:01:02.0Z</timestamp>
        <state>opened</state>
    </sensingData>
</sensingListData>
```

5.3.20.2.4.2 Read-reply

If one is interested in the activities of a switch for the time between 12:00 and 14:00 o'clock, he could send the following read command:

```
<function>sensingListData</function>
<filter>
    <cmdControl>
        <partial/>
    </cmdControl>
    <sensingListDataSelectors>
        <timestampInterval>
            <startTime>2013-07-15T12:00:00.0Z</startTime>
            <endTime>2013-07-15T14:00:00.0Z</endTime>
        </timestampInterval>
    </sensingListDataSelectors>
</filter>
<sensingListData/>
```

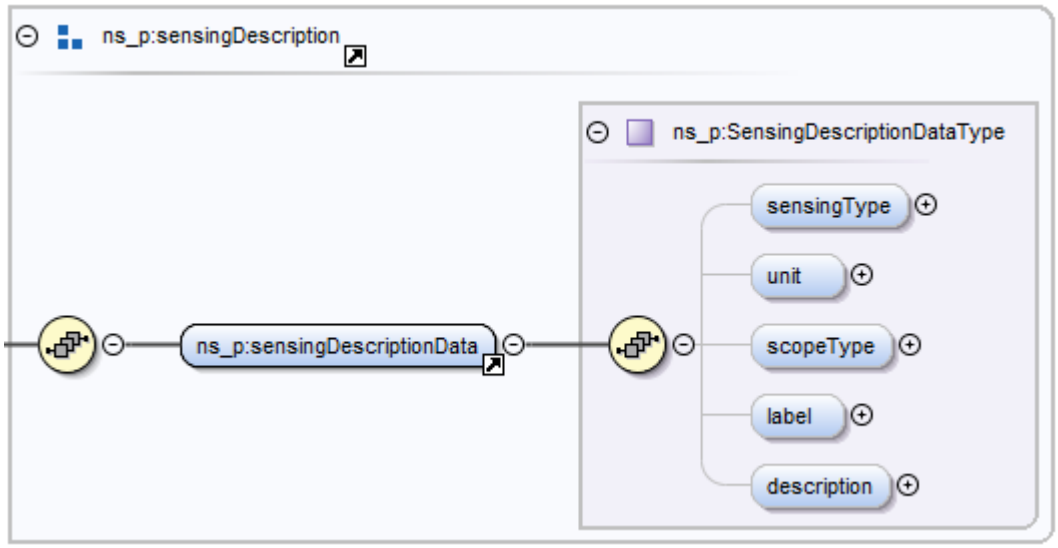
As a reply, something like the following could be returned:

```
<function>sensingListData</function>
<filter>
    <cmdControl>
        <partial/>
    </cmdControl>
</filter>
<sensingListData>
    <sensingData>
        <timestamp>2013-07-15T12:01:00.0Z</timestamp>
        <state>on</state>
    </sensingData>
    <sensingData>
        <timestamp>2013-07-15T12:50:32.0Z</timestamp>
        <state>off</state>
    </sensingData>
    <sensingData>
        <timestamp>2013-07-15T13:15:05.0Z</timestamp>
        <state>on</state>
    </sensingData>
</sensingListData>
```


9379 ***5.3.20.3 sensingDescriptionData***

9380 *5.3.20.3.1 General*

9381 The rather static (non-changing) elements of the *Sensing* class are combined in the
9382 *sensingDescriptionData* function. Typically, it is only requested once by a user or an application.



9383

9384 *Figure 166: sensingDescriptionData function overview*

9385

9386 *5.3.20.3.2 Detailed description of elements*

Element	Type	Description
sensingType	*Union "*SensingTypeType*":* *- Enum (see* Table 280*):* SensingTypeEnumType - EnumExtendType (see section 3.10.1.5)	Which type of sensor is modelled is denoted with this element.
unit	*Common data type "UnitOfMeasurementType". See section 3.10.1.17.*	The unit in which the values of the sensor are given. Not all sensor types need the value element and therefore the unit may be omitted as well.
scopeType	*Common data type "ScopeTypeType". See section 3.10.1.21.*	Specifies a more detailed meaning of the sensor (e.g. shutter).
label	*Common data type "LabelType". See section 3.10.1.2.*	A short label of the sensor.
description	*Common data type "DescriptionType". See section 3.10.1.3.*	Descriptive information on the sensor.

9387 *Table 279: sensingDescriptionData function detailed description of elements*

9388 Enumeration **SensingTypeEnumType**:

Value	Description
switch	A switch that has usually two states "on" and "off".
button	A button that can be "pressed" (and eventually supports "longPressed" and "released")
level	Some level functionality supporting one or more of the level-related sensing states.
levelSwitch	Some level functionality supporting one or more of the level-related sensing states and a switch functionality ("on" / "off").

windowHandle	A window handle supporting different positions.
contactSensor	Sensor that can detect whether something is "open" or "closed".
occupancySensor	Sensor that can detect whether someone is present in a room ("detected") or not ("notDetected").
motionDetector	Sensor that can detect motion ("detected") and eventually supports the "notDetected" state, too.
fireDetector	A sensor that detects fire ("detected") and eventually the absence of fire ("notDetected").
smokeDetector	A sensor that detects smoke ("detected") and eventually the absence of smoke ("notDetected").
heatDetector	A sensor that detects heat ("detected") and eventually the absence of heat ("notDetected").
waterDetector	A sensor that detects water ("detected") and eventually the absence of water ("notDetected").
gasDetector	A sensor that detects gas ("detected") and eventually the absence of gas ("notDetected").
alarmSensor	A sensor providing alarm information ("alarmed" / "notAlarmed").
powerAlarmSensor	A sensor reporting power alarms ("alarmed" / "notAlarmed").
dayNightIndicator	A sensor detecting whether it is "day" or "night".

Table 280: Enumeration SensingTypeEnumType

5.3.20.3.3 Available selectors

None.

5.3.20.3.4 Examples

5.3.20.3.4.1 Notify

An example instance of *sensingDescriptionData* of a smoke detector, sent as notification:

```
<sensingDescriptionData>
    <sensingType>smokeDetector</sensingType>
    <label>SmokeDetector Bedroom</label>
</sensingDescriptionData>
```

5.3.21 Setpoint

5.3.21.1 Introduction

Setting setpoint values is an often chosen way to control some functionality of a device. Especially in the home automation and HVAC domain, several use cases depend on setpoints. This class handles the setpoint itself. Relations to Measurement and TimeTable class enables the control of a measurand for specific time frames.

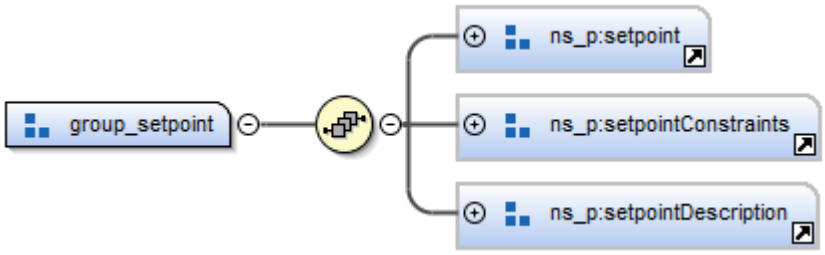


Figure 167: Setpoint function-group overview

5.3.21.2 setpointListData

5.3.21.2.1 General

The setpoint values and (if needed) the identifier of related measurands or time tables are modelled within this function. It is possible to either define an exact value, that should be reached or a range which shall not be fallen below or exceeded. Absolute or relative tolerances can be defined, too.

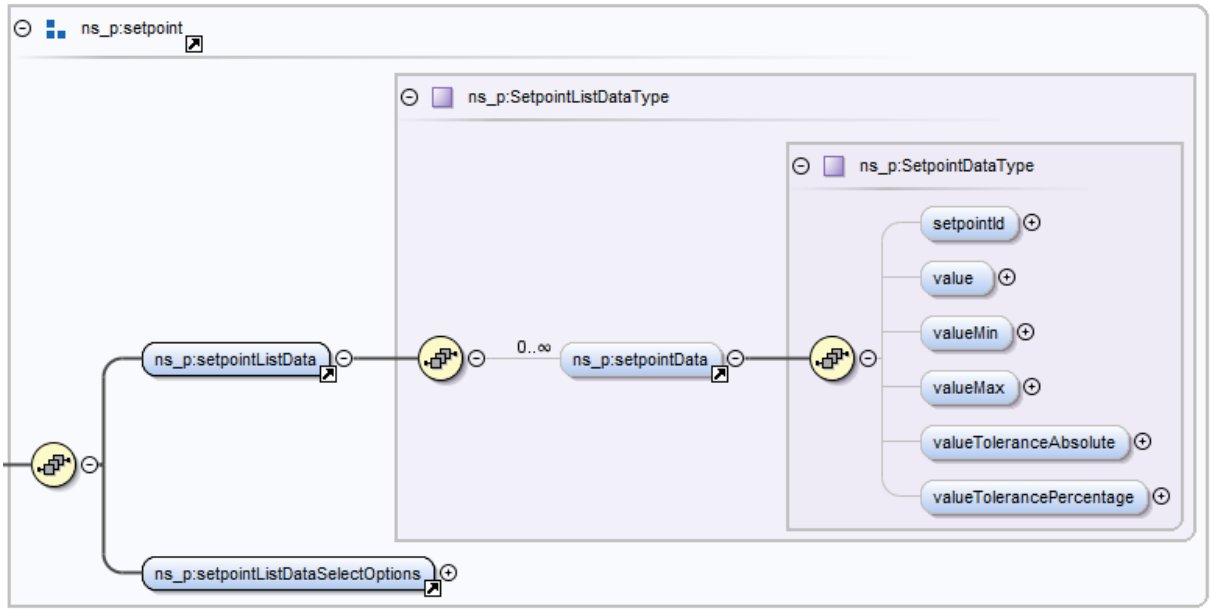


Figure 168: setpointListData function overview

5.3.21.2.2 Detailed description of elements

Element	Type	Description
setpointId	*Identifier "SetpointIdType" (see* Table 339*).*	Identifier of the setpoint.
value	*Common data type "ScaledNumberType". See section 3.10.1.8.*	The setpoint value itself.
valueMin	*Common data type "ScaledNumberType". See section 3.10.1.8.*	Instead of a fixed value (see above) a value range can be set. This would be the lower limit.

valueMax	*Common data type "ScaledNumberType". See section 3.10.1.8.*	Instead of a fixed value (see above) a value range can be set. This would be the upper limit.
valueToleranceAbsolute	*Common data type "ScaledNumberType". See section 3.10.1.8.*	If the value may vary some absolute value around the desired setpoint, the maximum variation allowed (in both directions) can be stated here.
valueTolerancePercentage	*Common data type "ScaledNumberType". See section 3.10.1.8.*	If the value may vary some percentage around the desired setpoint, the maximum variation allowed (in both directions) can be stated here.

Table 281: setpointListData function detailed description of elements

5.3.21.2.3 Available selectors

- setpointId

5.3.21.2.4 Examples

5.3.21.2.4.1 Read-reply

If one is interested in the setpoint with the *setpointId* "3", he could send the following read command:

```
<function>setpointListData</function>
<filter>
    <cmdControl>
        <partial/>
    </cmdControl>
    <setpointListDataSelectors>
        <setpointId>3</setpointId>
    </setpointListDataSelectors>
</filter>
<setpointListData/>
```

As a reply, something like the following could be returned:

```
<function>setpointListData</function>
<filter>
    <cmdControl>
        <partial/>
    </cmdControl>
</filter>
<setpointListData>
    <setpointData>
        <setpointId>3</setpointId>
        <value>
            <number>25</number>
        </value>
        <valueToleranceAbsolute>
            <number>1</number>
        </valueToleranceAbsolute>
    </setpointData>
</setpointListData>
```


5.3.21.3 setpointConstraintsListData

5.3.21.3.1 General

The constraints regarding the possible setpoints allowed are modelled within this function.

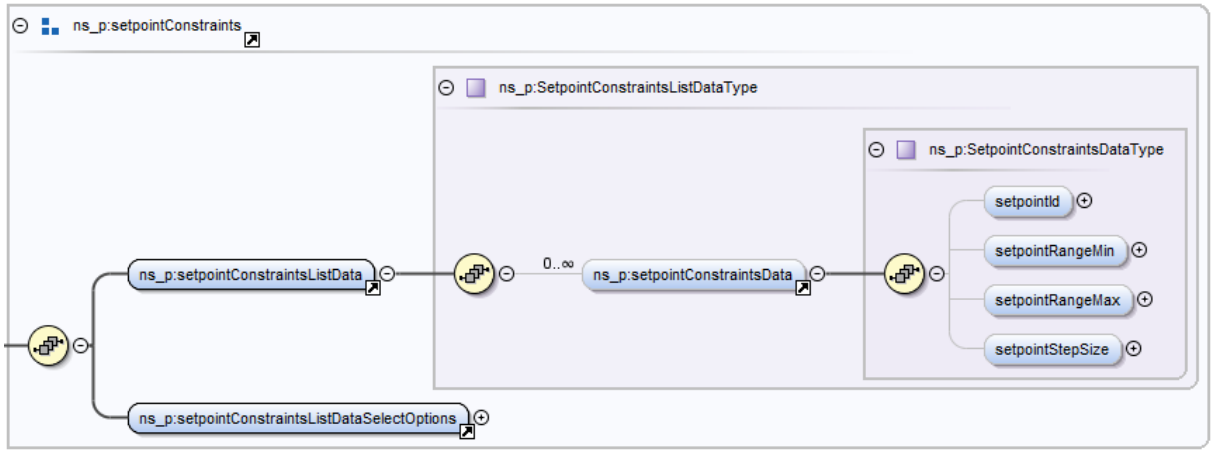


Figure 169: setpointConstraintsListData function overview

5.3.21.3.2 Detailed description of elements

Element	Type	Description
setpointId	*Identifier "SetpointIdType" (see* Table 339*).*	Identifier of the setpoint, these constraints belong to.
setpointRangeMin	*Common data type "ScaledNumberType". See section 3.10.1.8.*	The minimum value, the setpoint can be set to.
setpointRangeMax	*Common data type "ScaledNumberType". See section 3.10.1.8.*	The maximum value, the setpoint can be set to.
setpointStepSize	*Common data type "ScaledNumberType". See section 3.10.1.8.*	The minimum step size between two different values.

Table 282: setpointConstraintsListData function detailed description of elements

5.3.21.3.3 Available selectors

- setpointId

5.3.21.3.4 Examples

5.3.21.3.4.1 Read-reply

If one is interested in the setpoint constratins of all *setpointId*s, he could send the standard read command. A reply could be:

```
<setpointConstraintsListData>
    <setpointConstraintsData>
        <setpointId>1</setpointId>
```



```
            <setpointRangeMin>
                <number>0</number>
            </setpointRangeMin>
            <setpointRangeMax>
                <number>3</number>
                <scale>1</scale>
            </setpointRangeMax>
            <setpointStepSize>
                <number>5</number>
                <scale>-1</scale>
            </setpointStepSize>
        </setpointConstraintsData>
        <setpointConstraintsData>
            <setpointId>2</setpointId>
            <setpointRangeMin>
                <number>3</number>
                <scale>1</scale>
            </setpointRangeMin>
            <setpointRangeMax>
                <number>8</number>
                <scale>1</scale>
            </setpointRangeMax>
            <setpointStepSize>
                <number>1</number>
            </setpointStepSize>
        </setpointConstraintsData>
        <setpointConstraintsData>
            <setpointId>3</setpointId>
            <setpointRangeMin>
                <number>5</number>
            </setpointRangeMin>
            <setpointRangeMax>
                <number>2</number>
                <scale>2</scale>
            </setpointRangeMax>
            <setpointStepSize>
                <number>5</number>
            </setpointStepSize>
        </setpointConstraintsData>
    </setpointConstraintsListData>
```

5.3.21.4 setpointDescriptionListData

5.3.21.4.1 General

The rather static (non-changing) elements of the *Setpoint* class are combined in the *setpointDescriptionListData* function. Typically, it is only requested once by a user or an application.

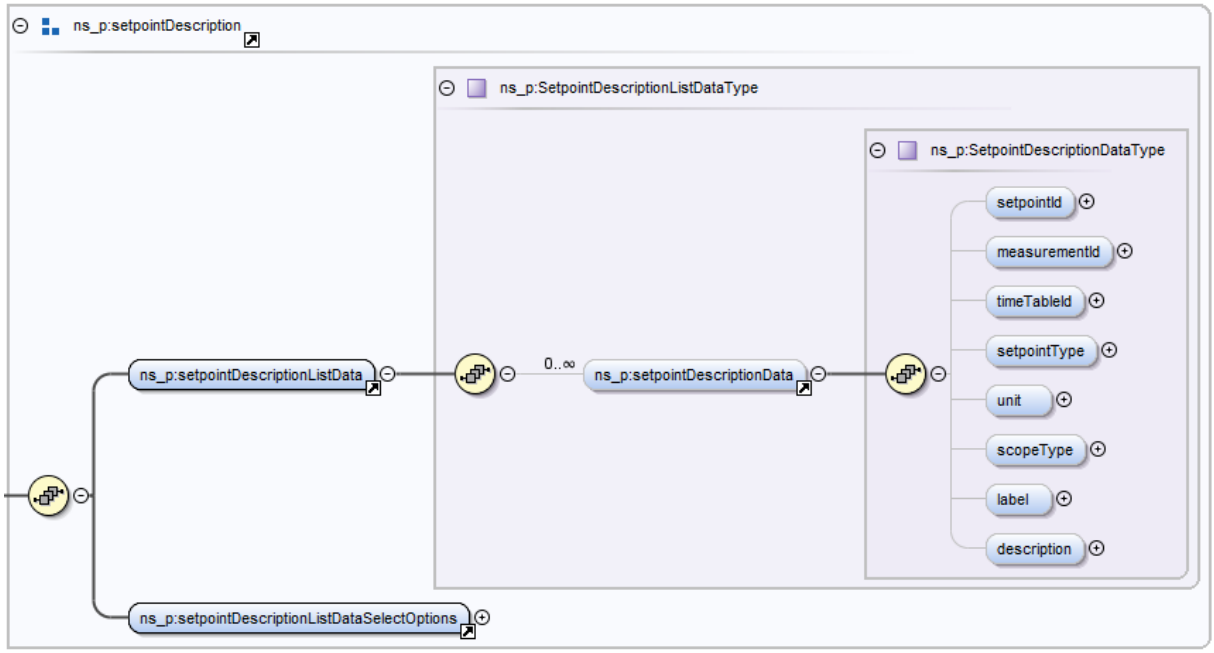


Figure 170: setpointDescriptionListData function overview

5.3.21.4.2 Detailed description of elements

Element	Type	Description
setpointId	*Identifier "SetpointIdType" (see* Table 339*).*	Identifier of the setpoint.
measurementId	*Identifier "MeasurementIdType" (see* Table 339*).*	If the setpoint relates to a measurand, the corresponding ID can be stated here.
timeTableId	*Identifier "timeTableIdType" (see* Table 339*).*	If the activation of the setpoint is time dependent, the ID of a corresponding time table can denoted in this element.
setpointType	*Union "*SetpointTypeType*":* *- Enum (see* Table 284*):* SetpointTypeEnumType - EnumExtendType (see section 3.10.1.5)	This is the type of setpoint used for this specific setpoint.
unit	*Common data type "UnitOfMeasurementType". See section 3.10.1.17.*	The unit used for this setpoint (see SPINE data model for all possible values).
scopeType	*Common data type "ScopeTypeType". See section 3.10.1.21.*	Specifies a more detailed meaning of the setpoint.
label	*Common data type "LabelType". See section 3.10.1.2.*	A short label of the setpoint.
description	*Common data type "DescriptionType". See section 3.10.1.3.*	Descriptive information on the setpoint.

Table 283: setpointDescriptionListData function detailed description of elements

Enumeration **SetpointTypeEnumType**:

Value	Description

valueAbsolute	An absolute value, e.g. 28°C. A unit is needed in most cases.
valueRelative	A relative value in percentage. The unit is "pct" in most cases.

Table 284: Enumeration SetpointTypeEnumType

5.3.21.4.3 Available selectors

- setpointId
- measurementId
- timeTableId
- setpointType
- scopeType

5.3.21.4.4 Examples

5.3.21.4.4.1 Read-reply

If one is interested in the setpoints related to *measurementId* "3", he could send the following read command:

```
<function>setpointDescriptionListData</function>
<filter>
    <cmdControl>
        <partial/>
    </cmdControl>
    <setpointDescriptionListDataSelectors>
        <measurementId>3</measurementId>
    </setpointDescriptionListDataSelectors>
</filter>
<setpointDescriptionListData/>
```

The reply could look like this:

```
<function>setpointDescriptionListData</function>
<filter>
    <cmdControl>
        <partial/>
    </cmdControl>
</filter>
<setpointDescriptionListData>
    <setpointDescriptionData>
        <setpointId>1</setpointId>
        <measurementId>3</measurementId>
        <timeTableId>2</timeTableId>
        <setpointType>valueAbsolute</setpointType>
        <unit>degC</unit>
        <label>Day heating</label>
    </setpointDescriptionData>
    <setpointDescriptionData>
        <setpointId>2</setpointId>
        <measurementId>3</measurementId>
        <timeTableId>4</timeTableId>
        <setpointType>valueAbsolute</setpointType>
        <unit>degC</unit>
        <label>Night heating</label>
    </setpointDescriptionData>
</setpointDescriptionListData>
```


5.3.22 SubscriptionManagement

5.3.22.1 Introduction

Subscriptions are used to simplify communication between devices. Based on matching features, finding suitable communication partners is a lot easier than via manual configuration only.

Each subscription exists between a subscription client and a subscription server. Only clients may initiate new subscriptions. After a successful establishment of a subscription, the subscription server updates the client about changes of values or parameters of the corresponding feature, depending on the subscribed functionality. E.g. a subscription server sends new temperature values of a subscribed measurement feature to the subscription client. The exact definition of rules and proceedings is out of scope of this section.

The *SubscriptionManagement* class provides functions for establishing, reading and deleting subscriptions.

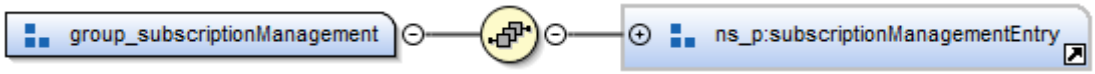


Figure 171: SubscriptionManagement function-group overview (data)

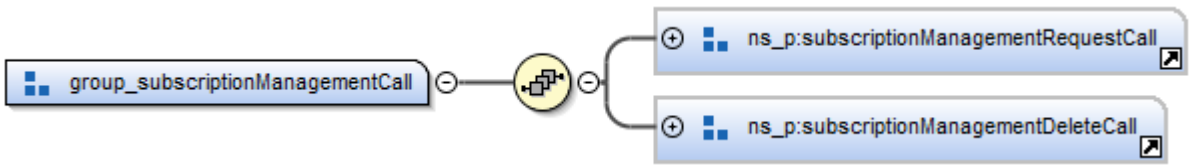


Figure 172: SubscriptionManagement function-group overview (call)

5.3.22.2 subscriptionManagementEntryListData

5.3.22.2.1 General

All existing subscriptions can be listed in the *subscriptionManagementEntryListData* function. It may be used to store the entries in the device itself. If another device asks for the list of subscriptions, a device may decide whether it only replies the subscriptions with the requesting device or all available subscriptions.

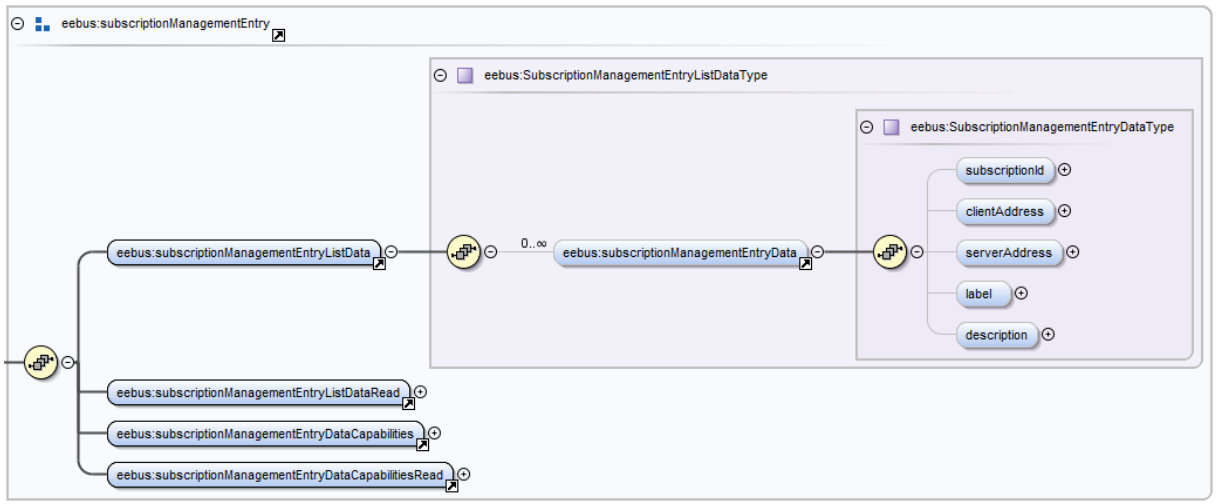


Figure 173: subscriptionManagementEntryListData function overview

5.3.22.2.2 Detailed description of elements

Element	Type	Description
subscriptionId	*Identifier "SubscriptionIdType" (see* Table 339*).*	Internal identifier for the specific subscription entry.
clientAddress	*Common data type "FeatureAddressType". See section 3.10.3.6.*	Address of the feature with the role *client* in this subscription.
serverAddress	*Common data type "FeatureAddressType". See section 3.10.3.6.*	Address of the feature with the role *server* in this subscription.
label	*Common data type "LabelType". See section 3.10.1.2.*	A short label of the subscription.
description	*Common data type "DescriptionType". See section 3.10.1.3.*	Descriptive information on the subscription.

Table 285: subscriptionManagementEntryListData function detailed description of elements

5.3.22.2.3 Available selectors

- subscriptionId
- clientAddress
- serverAddress

5.3.22.2.4 Examples

5.3.22.2.4.1 Read-reply

If one is interested in all subscriptions of a device, he could send a standard read command to the device, which could reply something like this:

```
<subscriptionManagementEntryListData>
    <subscriptionManagementEntryData>
```



```
            <subscriptionId>1</subscriptionId>
            <clientAddress>
                <device>d:_i:46925_CP\_pqtl-276381626831726378123568l3</device>
                <entity>2</entity>
                <feature>1</feature>
            </clientAddress>
            <serverAddress>
                <device>d:_i:46925_CP\_arqf-64986751356897215468975468</device>
                <entity>1</entity>
                <feature>1</feature>
            </serverAddress>
            <label>LightInfo_01</label>
        </subscriptionManagementEntryData>
        <subscriptionManagementEntryData>
            <subscriptionId>2</subscriptionId>
            <clientAddress>
                <device>d:_i:46925_CP\_pqtl-276381626831726378123568l3</device>
                <entity>2</entity>
                <feature>4</feature>
            </clientAddress>
            <serverAddress>
                <device>d:_i:46925_CP\_arqf-64986751356897215468975468</device>
                <entity>1</entity>
                <feature>2</feature>
            </serverAddress>
            <label>LightInfo_02</label>
        </subscriptionManagementEntryData>
</subscriptionManagementEntryListData>
```

5.3.22.3 subscriptionManagementRequestCall

5.3.22.3.1 General

To initiate a new subscription, an entity capable of node management on the client side, sends a *subscriptionManagementRequestCall* to an entity capable of node management on the server side, with the clientAddress element in the payload set to one of its features with the role *client* and the serverAddress element in the payload set to one of the other device's feature address with the role *server*. The receiving server device checks whether it will accept or decline the request and responds with an acknowledgement message (see [ProtocolSpecification], section "Acknowledgement message") with the *resultData* function and an according *errorNumber* set to 0 (accept subscription request) or some other value (definition of the error code is out of scope of this section; decline subscription request).

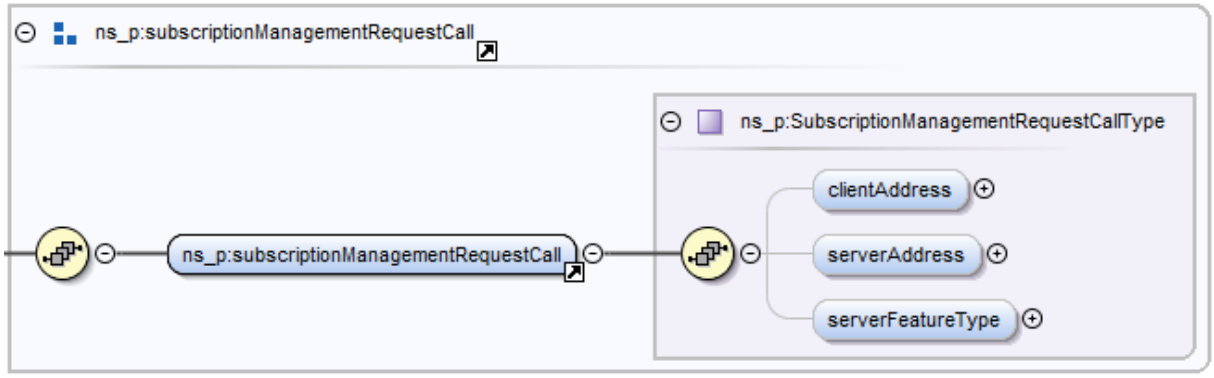


Figure 174: subscriptionManagementRequestCall function overview

9659

9660 *5.3.22.3.2 Detailed description of elements*

Element	Type	Description
clientAddress	*Common data type "FeatureAddressType". See section 3.10.3.6.*	Address of the feature with the role *client* in the subscription request.
serverAddress	*Common data type "FeatureAddressType". See section 3.10.3.6.*	Address of the feature with the role *server* in the subscription request.
serverFeatureType	*Common data type "FeatureTypeType". See section 3.10.1.29.*	The *Feature Type* of the server feature. Stated for verification.

9661 *Table 286: subscriptionManagementRequestCall function detailed description of elements*

9662

9663 *5.3.22.3.3 Available selectors*

9664 None.

9665

9666 *5.3.22.3.4 Examples*

9667 *5.3.22.3.4.1 Call*

9668 If a new subscription shall be initiated, the client device could send a command like this to the server
9669 device:

```
9670 <subscriptionManagementRequestCall>
9671     <clientAddress>
9672         <device>d:_i:46925_CP\_pqtl-27638162683172637812356813</device>
9673         <entity>2</entity>
9674         <feature>1</feature>
9675     </clientAddress>
9676     <serverAddress>
9677         <device>d:_i:46925_CP\_arqf-64986751356897215468975468</device>
9678         <entity>1</entity>
9679         <feature>1</feature>
9680     </serverAddress>
9681     <serverFeatureType>ActuatorSwitch</serverFeatureType>
9682 </subscriptionManagementRequestCall>
```

9683

9684 ***5.3.22.4 subscriptionManagementDeleteCall***

9685 *5.3.22.4.1 General*

9686 A device may delete an existing subscription with itself and another device. Therefore, this call is
9687 used.

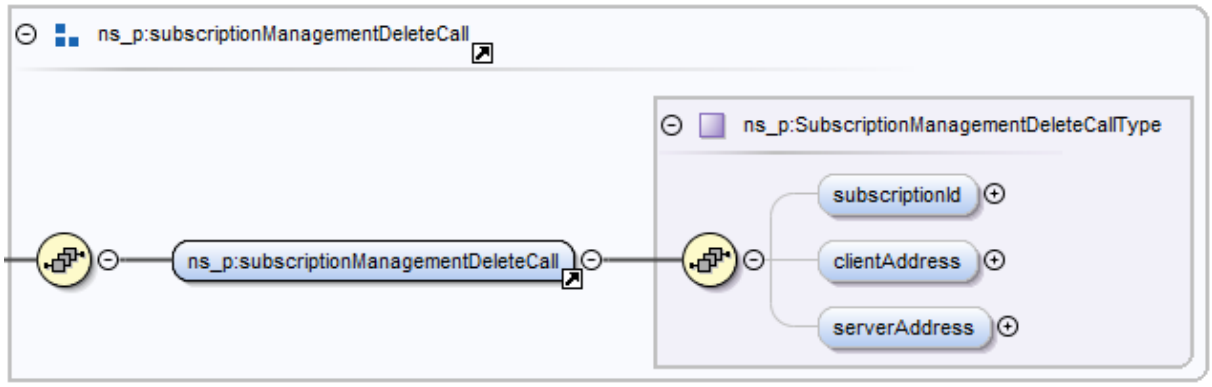


Figure 175: subscriptionManagementDeleteCall function overview

5.3.22.4.2 Detailed description of elements

Element	Type	Description
subscriptionId	*Identifier "SubscriptionIdType" (see* Table 339*).*	Internal identifier for the specific subscription entry.
clientAddress	*Common data type "FeatureAddressType". See section 3.10.3.6.*	Address of the feature with the role *client* in the subscription request.
serverAddress	*Common data type "FeatureAddressType". See section 3.10.3.6.*	Address of the feature with the role *server* in the subscription request.

Table 287: subscriptionManagementDeleteCall function detailed description of elements

5.3.22.4.3 Available selectors

None.

5.3.22.4.4 Examples

5.3.22.4.4.1 Call

If an existing subscription shall be deleted, one device could send a command like this to the other device:

```
<subscriptionManagementDeleteCall>
    <subscriptionId>1</subscriptionId>
    <clientAddress>
        <device>d:_i:46925_CP\_pqtl-276381626831726378123568l3</device>
        <entity>2</entity>
        <feature>1</feature>
    </clientAddress>
    <serverAddress>
        <device>d:_i:46925_CP\_arqf-64986751356897215468975468</device>
        <entity>1</entity>
        <feature>1</feature>
    </serverAddress>
</subscriptionManagementDeleteCall>
```


5.3.23 SupplyCondition

5.3.23.1 Introduction

Practice shows that the supply of electricity, water, etc. can be temporarily impaired. Some possible causes are announced maintenance work at the supply line, supply network overload, or depletion of a local energy resource (battery, e.g.). The Supply Condition concept permits indication of the supply condition by a household's SPINE data models.

The concept is less intended for any automatisms. It is rather intended for indications to users in order to prepare or apply measures in time.

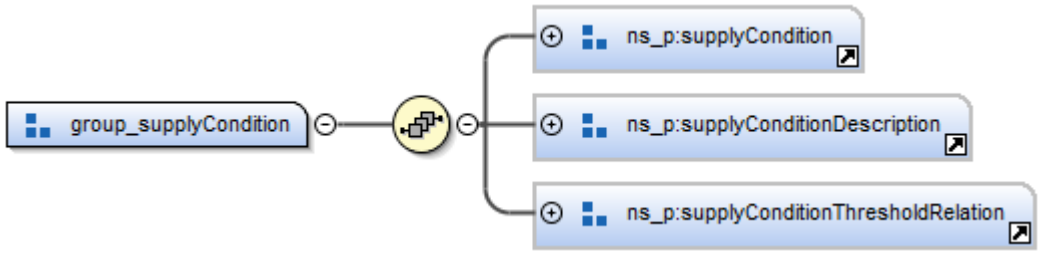


Figure 176: SupplyCondition function-group overview

5.3.23.2 supplyConditionListData

5.3.23.2.1 General

This function can be used to model the "events", i.e. the information on a specific supply condition. These events can, e.g., be sent from a DSO to a household. In this case, the household would be the data owner and the DSO would send (as a client) write commands to this resource.

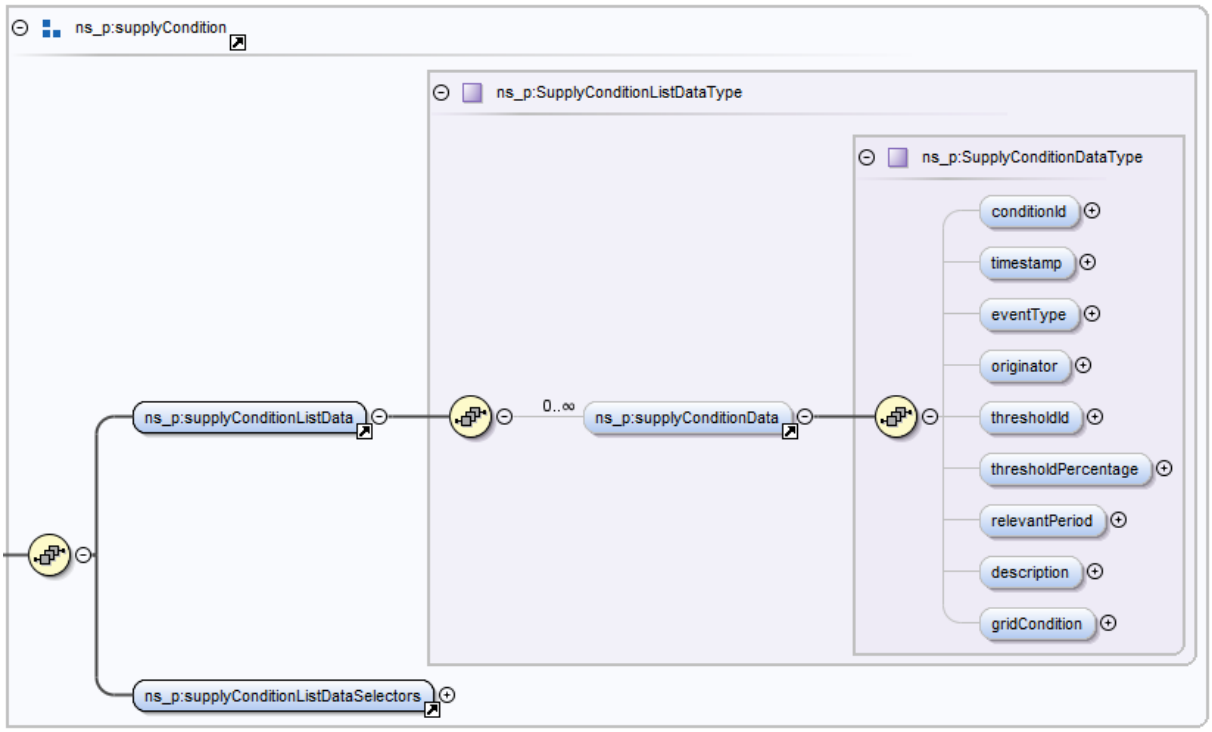


Figure 177: supplyConditionListData function overview

9733

9734 *5.3.23.2.2 Detailed description of elements*

Element	Type	Description
conditionId	*Identifier "ConditionIdType" (see* Table 339*).*	Identifier of the condition.
timestamp	*Common data type "AbsoluteOrRelativeTimeType". See section 3.10.2.3.*	The timestamp of the event.
eventType	*Union "*SupplyConditionEventTypeType*":* *- Enum (see* Table 289*):* SupplyConditionEventTypeEnumType - EnumExtendType (see section 3.10.1.5)	The kind of the event.
originator	*Union "*SupplyConditionOriginatorType*":* *- Enum (see* Table 290*):* SupplyConditionOriginatorEnumType - EnumExtendType (see section 3.10.1.5)	The originator (in terms of instance, not in terms of address) of the information.
thresholdId	*Identifier "ThresholdIdType" (see* Table 339*).*	The identifier of this event's threshold (if available).
thresholdPercentage	*Common data type "ScaledNumberType". See section 3.10.1.8.*	Relation of current situation to (configured) threshold (if available).
relevantPeriod	*Common data type "TimePeriodType". See section 3.10.2.1.*	Esp. the announcement of maintenance can make use of this element in order to tell when a maintenance is planned.
description	*Common data type "DescriptionType". See section 3.10.1.3.*	Descriptive information on the event.
gridCondition	*Union "*GridConditionType*":* *- Enum (see* Table 291*):* GridConditionEnumType - EnumExtendType (see section 3.10.1.5)	Indicates the condition of the electricity grid.

9735 *Table 288: supplyConditionListData function detailed description of elements*

9736 Enumeration **SupplyConditionEventTypeEnumType**:

Value	Description
thesholdExceeded	The related threshold was exceeded (may be good or bad, see definition of the corresponding *thresholdType* element in section 5.3.26.4.2 for details).
fallenBelowThreshold	The related threshold was fallen below (may be good or bad, see definition of the corresponding *thresholdType* element in section 5.3.26.4.2 for details).
supplyInterrupt	The supply of the corresponding commodity was interrupted.
releaseOfLimitations	"Back to normal", no limitations or threshold violations.
otherProblem	Some problem with the supply without further details about the problem.

9737 *Table 289: Enumeration SupplyConditionEventTypeEnumType*

Enumeration **SupplyConditionOriginatorEnumType**:

Value	Description
externDSO	The originator of the current condition "event" is the DSO (distribution system operator).
externSupplier	The originator of the current condition "event" is the energy supplier.
internalLimit	Some internal limit at the premises generated the current condition "event".
internalService	Due to an internal service at the premises, the current condition "event" was generated.
internalUser	An user interaction is the source of the current condition "event".

Table 290: Enumeration SupplyConditionOriginatorEnumType

Enumeration **GridConditionEnumType**:

Value	Description
consumptionRed	The electricity grid condition for consumption from the grid is "red".
consumptionYellow	The electricity grid condition for consumption from the grid is "yellow".
good	The electricity grid condition is "good".
productionYellow	The electricity grid condition for feeding into the grid is "yellow".
productionRed	The electricity grid condition for feeding into the grid is "red".

Table 291: Enumeration GridConditionEnumType

5.3.23.2.3 Available selectors

- conditionId
- timestampInterval

5.3.23.2.4 Examples

5.3.23.2.4.1 Notify

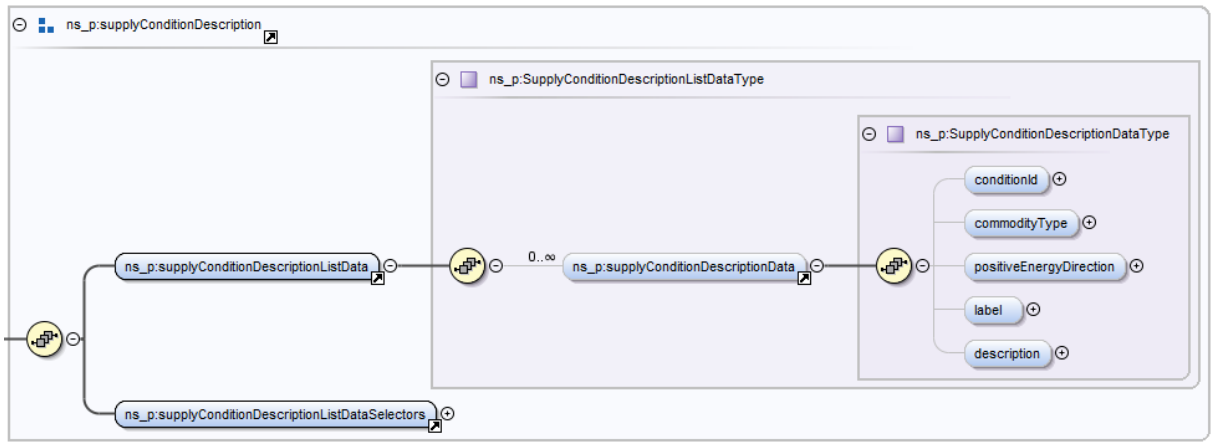
A user ordered a plumber and broadcasts this information to all displays (the “filter/partial” part to announce the “restricted function exchange is simplified by “...” just to improve the readability”):

```
...
<supplyConditionListData>
    <supplyConditionData>
        <conditionId>3</conditionId>
        <timestamp>2006-05-04T10:13:51.0Z</timestamp>
        <eventType>supplyInterrupt</eventType>
        <originator>internalService</originator>
        <relevantPeriod>
            <startTime>2006-05-04T14:00:00.0Z</startTime>
            <endTime>2006-05-04T15:00:00.0Z</endTime>
        </relevantPeriod>
        <description>Attention: No water supply from 2PM to
3PM</description>
        <gridCondition>good</gridCondition>
    </supplyConditionData>
</supplyConditionListData>
```


9768 ***5.3.23.3 supplyConditionDescriptionListData***

9769 *5.3.23.3.1 General*

9770 The more static information about a supply condition is modelled within this function.



9771

9772 *Figure 178: supplyConditionDescriptionListData function overview*

9773

9774 *5.3.23.3.2 Detailed description of elements*

Element	Type	Description
conditionId	*Identifier "ConditionIdType" (see* Table 339*).*	Identifier of the condition.
commodityType	*Common data type "CommodityTypeType". See section 3.10.1.9.*	The type of the commodity handled by this condition is stated here.
positiveEnergyDirection	*Common data type "EnergyDirectionType". See section 3.10.1.13.*	The *positiveEnergyDirection* states whether energy consumption or production will be counted as positive value.
label	*Common data type "LabelType". See section 3.10.1.2.*	A short label of the condition.
description	*Common data type "DescriptionType". See section 3.10.1.3.*	Descriptive information on the condition.

9775 *Table 292: supplyConditionDescriptionListData function detailed description of elements*

9776

9777 *5.3.23.3.3 Available selectors*

9778 - conditionId

9779

9780 *5.3.23.3.4 Examples*

9781 *5.3.23.3.4.1 Read-Reply*

9782 If one is interested in all supply conditions, he could send a standard read command to the server.

9783 The reply could look like this:


```
<supplyConditionDescriptionListData>
    <supplyConditionDescriptionData>
        <conditionId>1</conditionId>
        <commodityType>electricity</commodityType>
        <positiveEnergyDirection>consume</positiveEnergyDirection>
        <label>Electricity supply</label>
        <description>Supply for electricity (3-phase
connection).</description>
    </supplyConditionDescriptionData>
    <supplyConditionDescriptionData>
        <conditionId>2</conditionId>
        <commodityType>water</commodityType>
        <positiveEnergyDirection>consume</positiveEnergyDirection>
        <label>Water supply</label>
        <description>Supply for domestic water.</description>
    </supplyConditionDescriptionData>
</supplyConditionDescriptionListData>
```

5.3.23.4 supplyConditionThresholdRelationListData

5.3.23.4.1 General

Multiple thresholds can be associated to a condition. This leads to the relation

- 1 conditionId -> 0..m thresholdId

The supplyConditionThresholdRelationListData function can be used to model concrete relations.

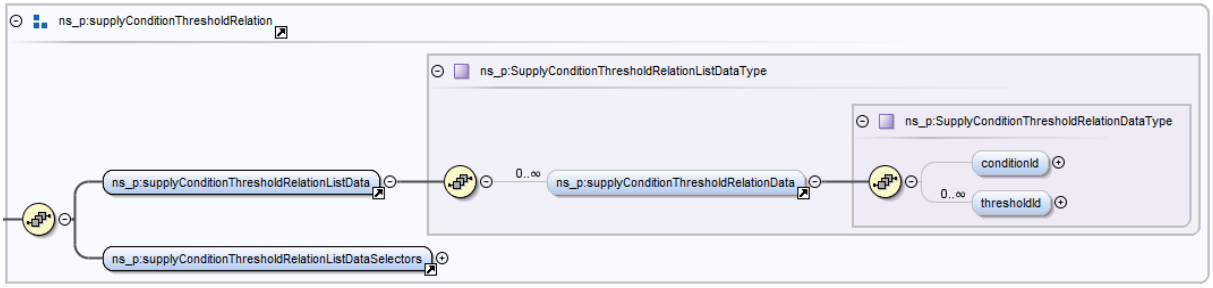


Figure 179: supplyConditionThresholdRelationListData function overview

5.3.23.4.2 Detailed description of elements

Element	Type	Description
conditionId	*Identifier "ConditionIdType" (see* Table 339*).*	Identifier of the condition.
thresholdId (list)	*Identifier "ThresholdIdType" (see* Table 339*).*	Related threshold identifier(s).

Table 293: supplyConditionThresholdRelationListData function detailed description of elements

5.3.23.4.3 Available selectors

- conditionId
- thresholdId

5.3.23.4.4 Examples

5.3.23.4.4.1 Read-reply

An energy management gateway is able to keep relation information between condition and (supply) threshold and provide the list upon request.

A user selects the energy management gateway on an external display and queries for information on conditionId 5. The display sends a read classifier together with the read command:

```
<function>supplyConditionThresholdRelationListData</function>
<filter>
    <cmdControl>
        <partial/>
    </cmdControl>
    <supplyConditionThresholdRelationListDataSelectors>
        <conditionId>5</conditionId>
    </supplyConditionThresholdRelationListDataSelectors>
</filter>
<messagingListData/>
```

The energy management gateway replies with a proper supplyConditionThresholdRelationListData:

```
<function>supplyConditionThresholdRelationListData</function>
<filter>
    <cmdControl>
        <partial/>
    </cmdControl>
</filter>
<supplyConditionThresholdRelationListData>
    <supplyConditionThresholdRelationData>
        <conditionId>5</conditionId>
        <thresholdId>3</thresholdId>
        <thresholdId>4</thresholdId>
    </supplyConditionThresholdRelationData>
</supplyConditionThresholdRelationListData>
```

5.3.24 TariffInformation

5.3.24.1 Introduction

The TariffInformation Class provides functions for modelling simple or complex data related to tariffs. The following figures visualize the dependencies of the different categories available in this class:

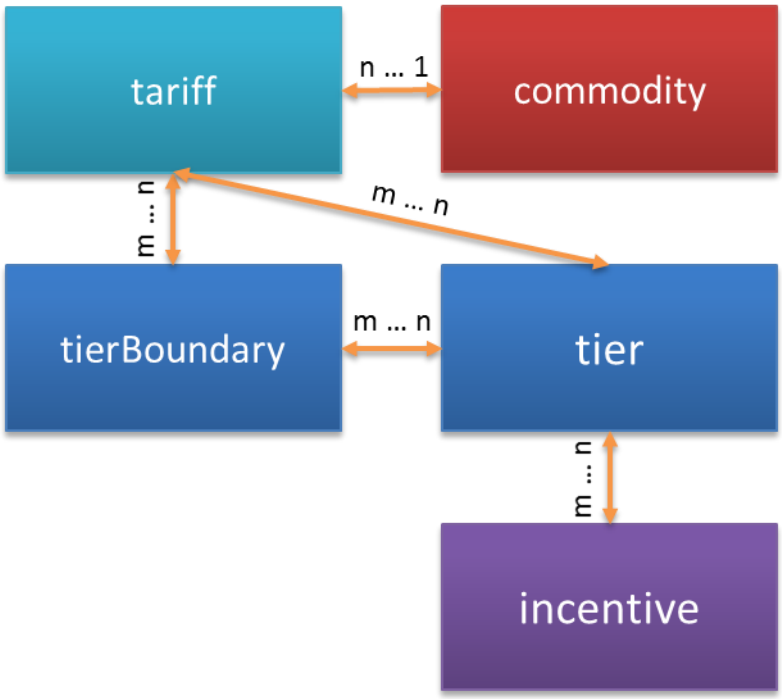


Figure 180: TariffInformation relations, sub-class level

Basis is the *tariff* that has one *commodity* and some *tiers* with *tierBoundaries*. Each *tier* itself has *incentives*. Identifiers are used for referencing the different aspects of the tariff model. They are shown in the figure below:

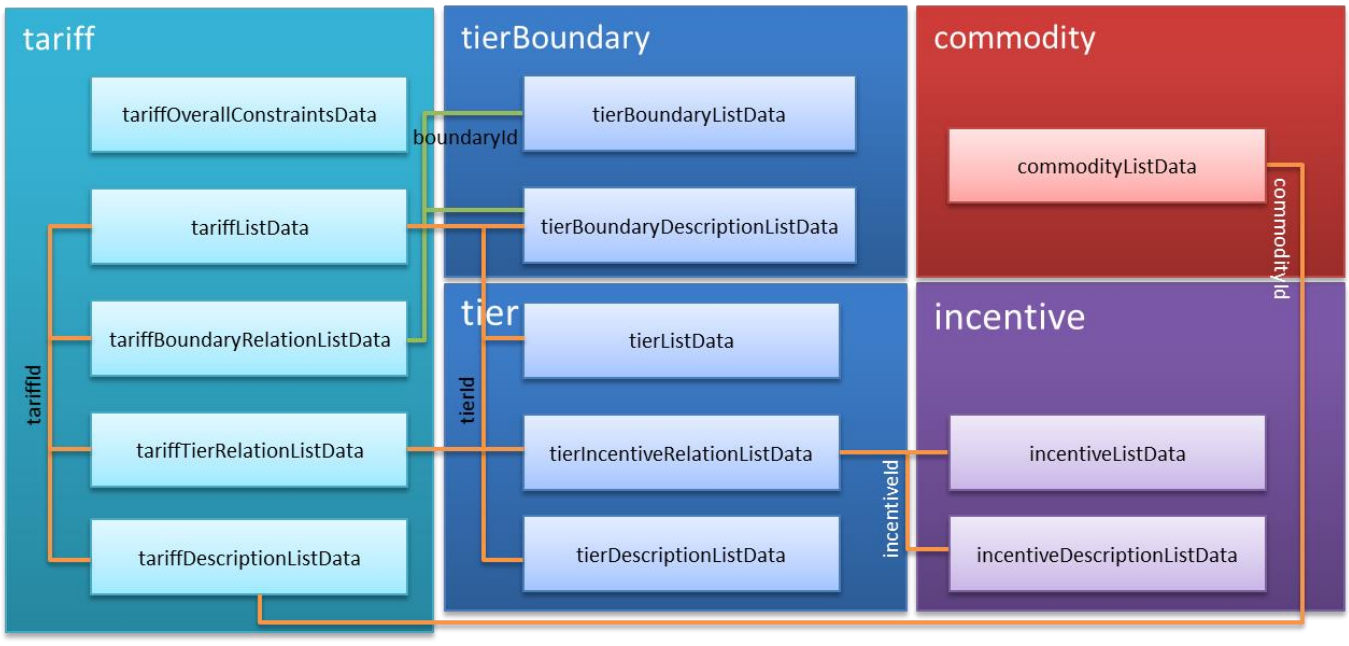


Figure 181: TariffInformation relations, function level

As it can be seen, the ***tariff*** consists of some overall constraints (*tariffOverallConstraintsData*), a dynamic data part (*tariffListData*) and the rather static part (*tariffDescriptionListData*). Additionally, relations are given to the related boundaries (*tariffBoundaryRelationListDat*) and tiers (*tariffTierRelationListData*).

The ***tierBoundary*** has a dynamic (*tierBoundaryListData*) and a rather static part (*tierBoundaryDescriptionListData*).

The ***tier*** consists of the dynamic (*tierListData*), the rather static part (*tierDescriptionListData*) and a relation to related incentives (*tierIncentiveRelationListData*).

The ***incentive*** has two functions, the dynamic part (*incentiveListData*) and the rather static part (*incentiveDescriptionListDat*).

The ***commodity*** only has one information (*commodityListData*).

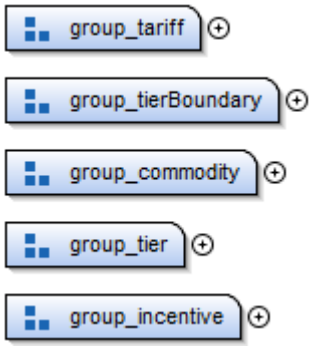


Figure 182: TariffInformation Sub-class overview

5.3.24.2 Sub-class Tariff

The information related to the tariff itself are modelled within the functions of this sub-class.

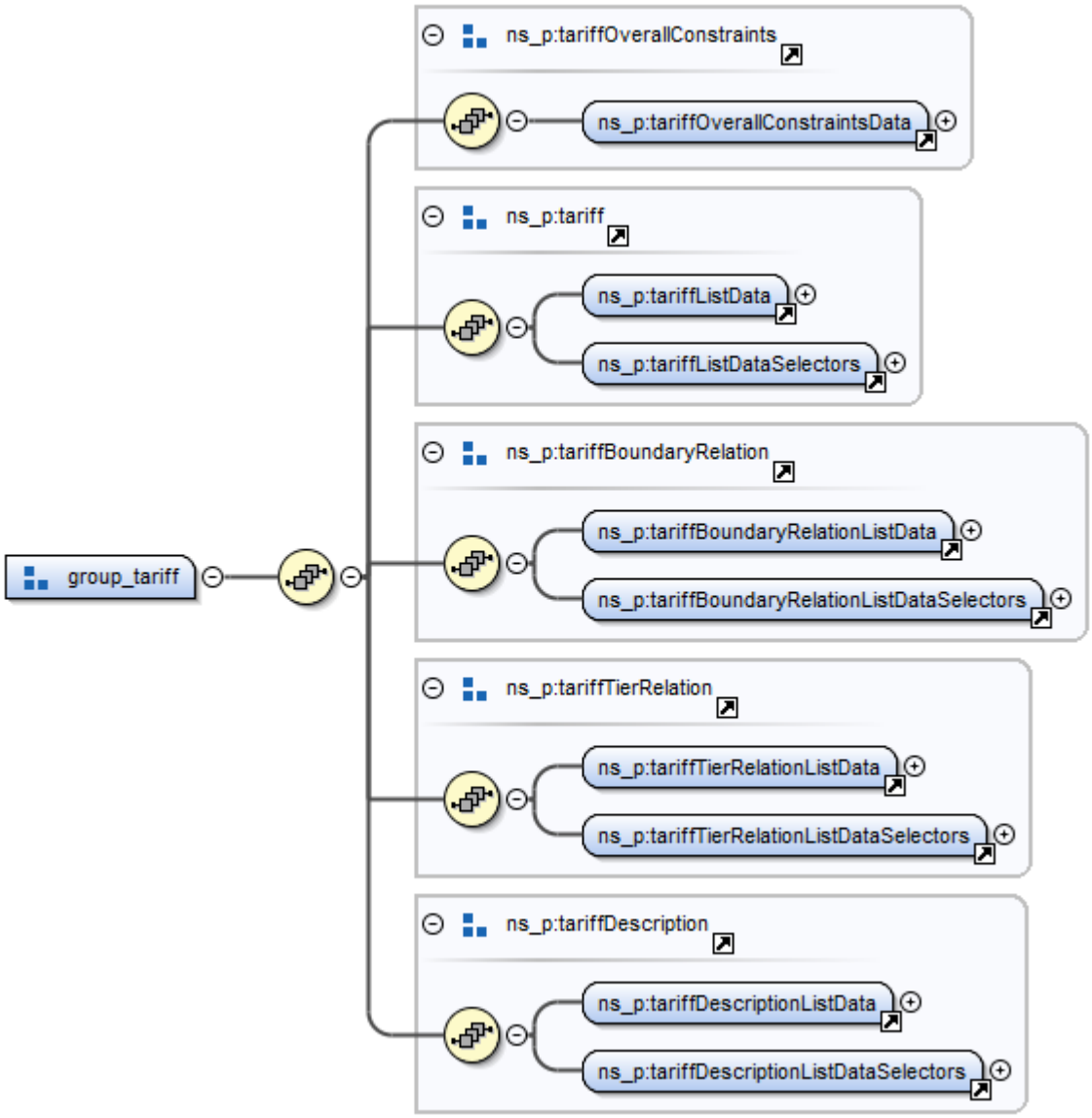
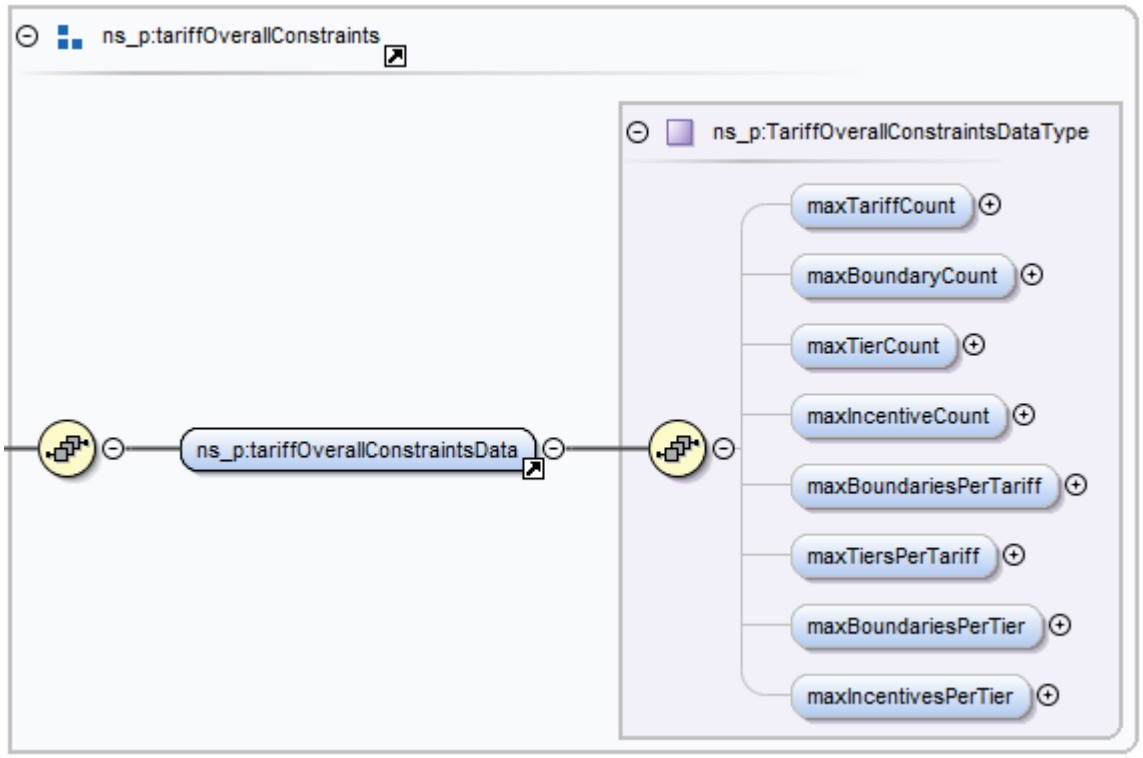


Figure 183: TariffInformation, sub-class Tariff, function-group overview

5.3.24.3 tariffOverallConstraintsData

5.3.24.3.1 General

Constraints that are valid for all tariffs on this particular feature are included in this function.



9881

9882 *Figure 184: tariffOverallConstraintsData function overview*

9883

9884 *5.3.24.3.2 Detailed description of elements*

Element	Type	Description
maxTariffCount	*Simple type "TariffCountType" (restriction of "TariffIdType" (see* Table 339*)).*	Maximum amount of tariffs supported by the server.
maxBoundaryCount	*Simple type "TierBoundaryCountType" (restriction of "TierBoundaryIdType" (see* Table 339*)).*	Maximum amount of tier boundaries supported by the server.
maxTierCount	*Simple type "TierCountType" (restriction of "TierIdType" (see* Table 339*)).*	Maximum amount of tiers supported by the server.
maxIncentiveCount	*Simple type "IncentiveCountType" (restriction of "IncentiveIdType" (see* Table 339*)).*	Maximum amount of incentives supported by the server.
maxBoundariesPerTariff	*Simple type "TierBoundaryCountType" (restriction of "TierBoundaryIdType" (see* Table 339*)).*	Maximum amount of (tier) boundaries per tariff supported by the server.
maxTiersPerTariff	*Simple type "TierCountType" (restriction of "TierIdType" (see* Table 339*)).*	Maximum amount of tiers per tariff supported by the server.
maxBoundariesPerTier	*Simple type "TierBoundaryCountType" (restriction of "TierBoundaryIdType" (see* Table 339*)).*	Maximum amount of boundaries per tier supported by the server.

maxIncentivesPerTier	*Simple type "IncentiveCountType" (restriction of "IncentiveIdType" (see* Table 339*)).*	Maximum amount of incentives per tier supported by the server.

Table 294: tariffOverallConstraintsData function detailed description of elements

5.3.24.3.3 Available selectors

None.

5.3.24.3.4 Examples

5.3.24.3.4.1 Read-reply

If one is interested in the tariff overall constraints, he could send a standard read command to the server. The reply could look like this:

```
<tariffOverallConstraintsData>
    <maxTariffCount>5</maxTariffCount>
    <maxBoundaryCount>40</maxBoundaryCount>
    <maxTierCount>20</maxTierCount>
    <maxIncentiveCount>60</maxIncentiveCount>
    <maxBoundariesPerTariff>8</maxBoundariesPerTariff>
    <maxTiersPerTariff>4</maxTiersPerTariff>
    <maxBoundariesPerTier>2</maxBoundariesPerTier>
    <maxIncentivesPerTier>3</maxIncentivesPerTier>
</tariffOverallConstraintsData>
```

5.3.24.4 tariffListData

5.3.24.4.1 General

The dynamic part of the tariff sub-class delivers the information which tier(s) is/are currently active for this tariff.

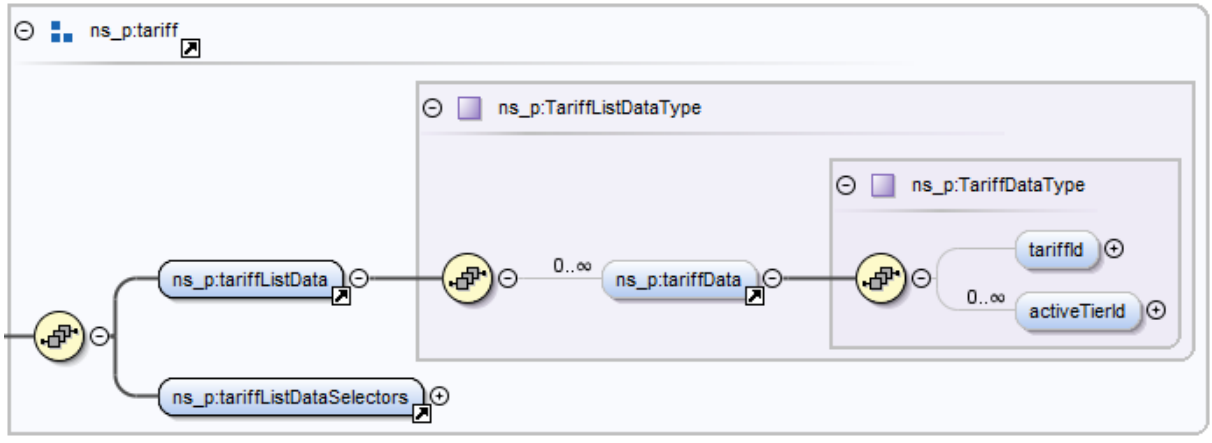


Figure 185: tariffListData function overview

9912 *5.3.24.4.2 Detailed description of elements*

Element	Type	Description
tariffId	*Identifier "TariffIdType" (see* Table 339*).*	Primary identifier for the tariff.
activeTierId (list)	*Identifier "TierIdType" (see* Table 339*).*	Foreign key(s) of the active tier(s) for this tariff.

9913 *Table 295: tariffListData function detailed description of elements*

9914

9915 *5.3.24.4.3 Available selectors*

9916 - tariffId
9917 - activeTierId

9918

9919 *5.3.24.4.4 Examples*

9920 *5.3.24.4.4.1 Read-reply*

9921 If one is interested in all dynamic tariff information, he could send a standard read command to the
9922 server. The reply could look like this:

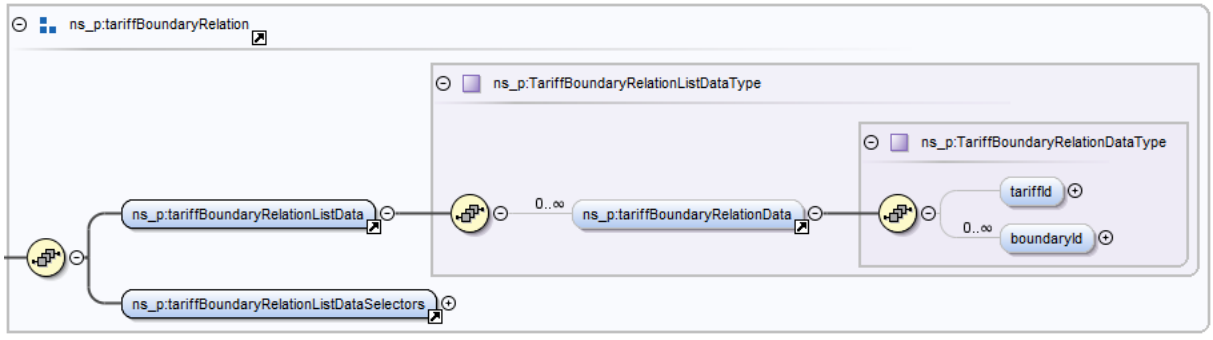
```
9923 <tariffListData>
9924     <tariffData>
9925         <tariffId>1</tariffId>
9926         <activeTierId>1</activeTierId>
9927         <activeTierId>2</activeTierId>
9928     </tariffData>
9929     <tariffData>
9930         <tariffId>2</tariffId>
9931         <activeTierId>3</activeTierId>
9932     </tariffData>
9933 </tariffListData>
```

9934

9935 ***5.3.24.5 tariffBoundaryRelationListData***

9936 *5.3.24.5.1 General*

9937 The boundaries related to a tariff are modelled within this function.



9938

9939 *Figure 186: tariffBoundaryRelationListData function overview*

9940

5.3.24.5.2 Detailed description of elements

Element	Type	Description
tariffId	*Identifier "TariffIdType" (see* Table 339*).*	Primary identifier for the tariff.
boundaryId (list)	*Identifier "TierBoundaryIdType" (see* Table 339*).*	Foreign key(s) of the related boundary(/ies) for this tariff.

Table 296: tariffBoundaryRelationListData function detailed description of elements

5.3.24.5.3 Available selectors

- tariffId
- boundaryId

5.3.24.5.4 Examples

5.3.24.5.4.1 Read-reply

If one is interested in the possible boundaries of all tariffs, he could send a standard read command to the server. The reply could look like this:

```
<tariffBoundaryRelationListData>
    <tariffBoundaryRelationData>
        <tariffId>1</tariffId>
        <boundaryId>1</boundaryId>
        <boundaryId>2</boundaryId>
        <boundaryId>3</boundaryId>
    </tariffBoundaryRelationData>
    <tariffBoundaryRelationData>
        <tariffId>2</tariffId>
        <boundaryId>1</boundaryId>
        <boundaryId>3</boundaryId>
    </tariffBoundaryRelationData>
</tariffBoundaryRelationListData>
```

5.3.24.6 tariffTierRelationListData

5.3.24.6.1 General

The tiers related to a tariff are modelled within this function.

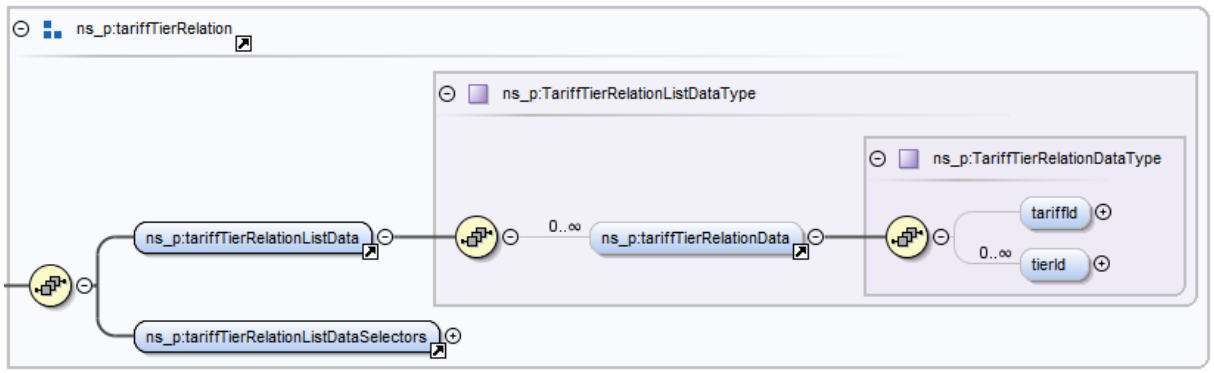


Figure 187: tariffTierRelationListData function overview

9972 *5.3.24.6.2 Detailed description of elements*

Element	Type	Description
tariffId	*Identifier "TariffIdType" (see* Table 339*).*	Primary identifier for the tariff.
tierId (list)	*Identifier "TierIdType" (see* Table 339*).*	Foreign key(s) of the related tier(s) for this tariff.

9973 *Table 297: tariffTierRelationListData function detailed description of elements*

9974

9975 *5.3.24.6.3 Available selectors*

9976 - tariffId
9977 - tierId

9978

9979 *5.3.24.6.4 Examples*

9980 *5.3.24.6.4.1 Read-reply*

9981 If one is interested in the possible tiers of all tariffs, he could send a standard read command to the
9982 server. The reply could look like this:

```
9983 <tariffTierRelationListData>
9984     <tariffTierRelationData>
9985         <tariffId>1</tariffId>
9986         <tierId>1</tierId>
9987         <tierId>2</tierId>
9988         <tierId>3</tierId>
9989         <tierId>4</tierId>
9990     </tariffTierRelationData>
9991     <tariffTierRelationData>
9992         <tariffId>2</tariffId>
9993         <tierId>1</tierId>
9994         <tierId>2</tierId>
9995     </tariffTierRelationData>
9996 </tariffTierRelationListData>
```

9997

9998 ***5.3.24.7 tariffDescriptionListData***

9999 *5.3.24.7.1 General*

10000 The rather static information about a tariff (e.g. the references to a commodity or a measurand, the
10001 scopeType, etc.) are modelled within this function.

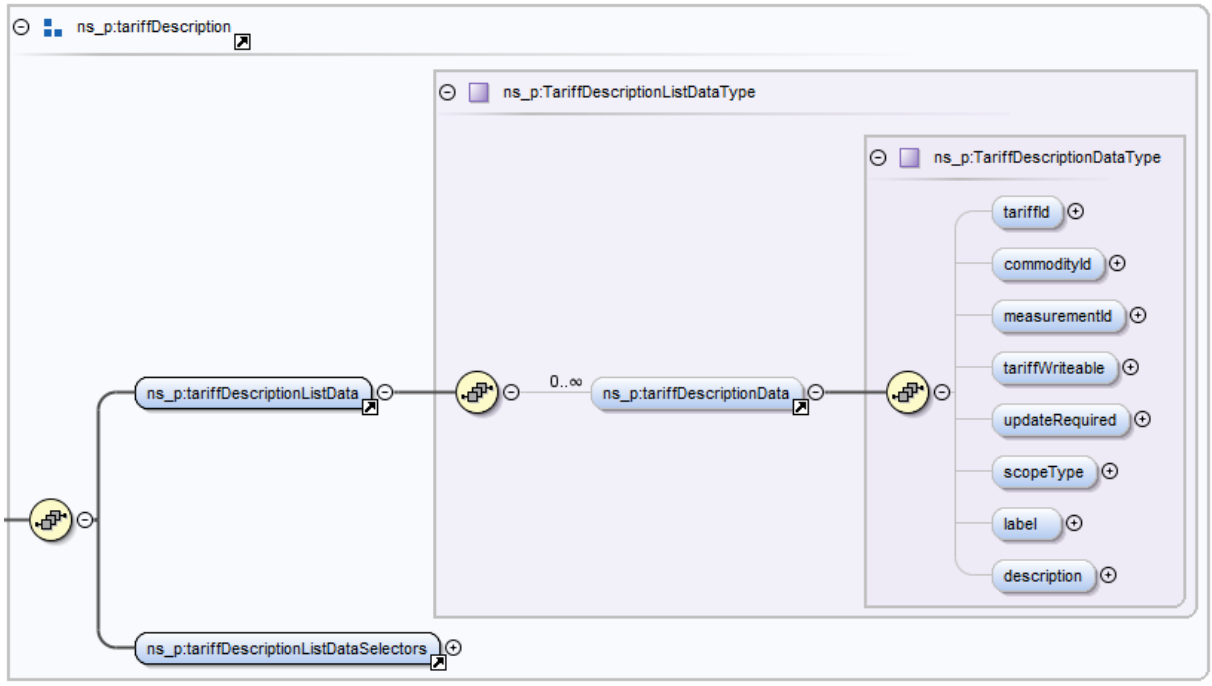


Figure 188: tariffDescriptionListData function overview

5.3.24.7.2 Detailed description of elements

Element	Type	Description
tariffId	*Identifier "TariffIdType" (see* Table 339*).*	Primary identifier for the tariff.
commodityId	*Identifier "CommodityIdType" (see* Table 339*).*	If the tariff is linked to a commodity, the corresponding identifier can be stated here.
measurementId	*Identifier "MeasurementIdType" (see* Table 339*).*	If the tariff is linked to a measurand, the corresponding identifier can be stated here.
tariffWriteable	xs:boolean (W3C standard type)	Whether the tariff is writeable or not can be denoted in this element.
updateRequired	xs:boolean (W3C standard type)	If an update for this tariff (by some bound client) is required, this element will be set to "true".
scopeType	*Common data type "ScopeTypeType". See section 3.10.1.21.*	The scope type for this tariff.
label	*Common data type "LabelType". See section 3.10.1.2.*	A short label of the tariff.
description	*Common data type "DescriptionType". See section 3.10.1.3.*	Descriptive information on the tariff.

Table 298: tariffDescriptionListData function detailed description of elements

5.3.24.7.3 Available selectors

- tariffId
- commodityId
- measurementId
- scopeType

10013

10014 *5.3.24.7.4 Examples*

10015 *5.3.24.7.4.1 Read-reply*

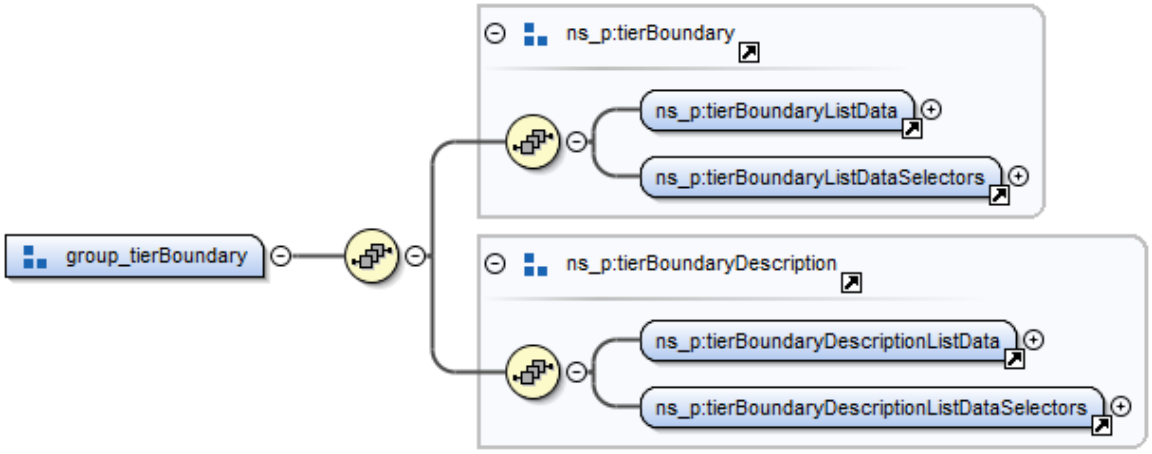
10016 If one is interested in all tariff descriptions, he could send a standard read command to the server.
10017 The reply could look like this:

```
10018 <tariffDescriptionListData>
10019     <tariffDescriptionData>
10020         <tariffId>1</tariffId>
10021         <commodityId>1</commodityId>
10022         <measurementId>1</measurementId>
10023         <tariffWriteable>false</tariffWriteable>
10024         <updateRequired>true</updateRequired>
10025         <label>Tariff 1 - Basic</label>
10026     </tariffDescriptionData>
10027     <tariffDescriptionData>
10028         <tariffId>2</tariffId>
10029         <commodityId>1</commodityId>
10030         <measurementId>4</measurementId>
10031         <tariffWriteable>true</tariffWriteable>
10032         <updateRequired>false</updateRequired>
10033         <label>Tariff 2 - Eco Green</label>
10034     </tariffDescriptionData>
10035 </tariffDescriptionListData>
```

10036

10037 ***5.3.24.8 Sub-class TierBoundary***

10038 All functions related to the boundaries of tiers are clustered in this *TierBoundary* sub-class.



10039

10040 *Figure 189: TariffInformation, sub-class TierBoundary, function-group overview*

10041

10042 ***5.3.24.9 tierBoundaryListData***

10043 *5.3.24.9.1 General*

10044 Boundaries (for tiers) are needed to identify in which value range the tier is valid. The validity can
10045 also depend on a time interval / time table.

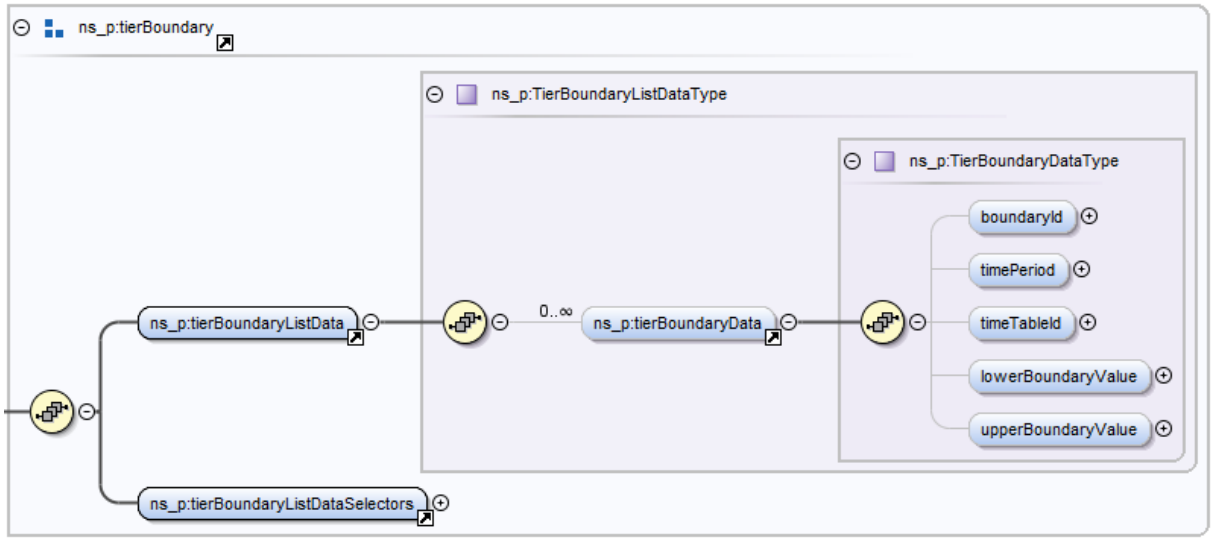


Figure 190: tierBoundaryListData function overview

5.3.24.9.2 Detailed description of elements

Element	Type	Description
boundaryId	*Identifier "TierBoundaryIdType" (see* Table 339*).*	Primary identifier for the boundary.
timePeriod	*Common data type "TimePeriodType". See section 3.10.2.1.*	If the boundary depends on a time period, it can be denoted here.
timeTableId	*Identifier "TimeTableIdType" (see* Table 339*).*	The identifier of a time table can be set here if the boundary depends on the given start and end times there.
lowerBoundaryValue	*Common data type "ScaledNumberType". See section 3.10.1.8.*	This represents to lower end value range boundary.
upperBoundaryValue	*Common data type "ScaledNumberType". See section 3.10.1.8.*	This represents to upper end value range boundary.

Table 299: tierBoundaryListData function detailed description of elements

5.3.24.9.3 Available selectors

- boundaryId

5.3.24.9.4 Examples

5.3.24.9.4.1 Read-reply

If one is interested in the dynamic parts of all tier boundaries, he could send a standard read command to the server. The reply could look like this:

```
<tierBoundaryListData>
    <tierBoundaryData>
```

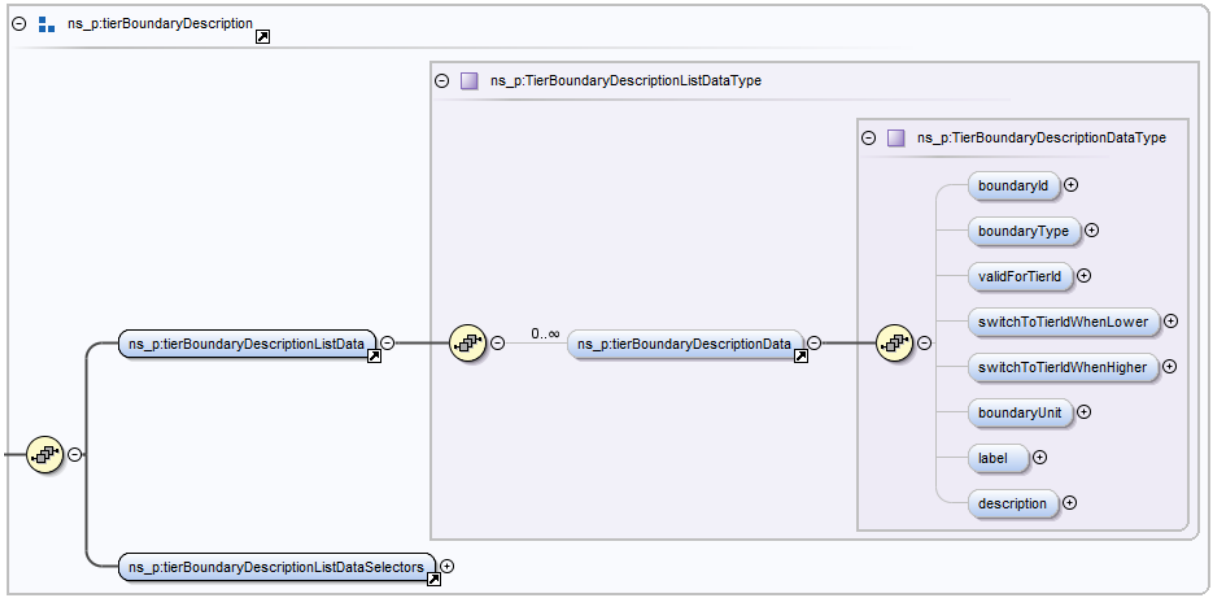


```
            <boundaryId>1</boundaryId>
            <timePeriod>
                <startTime>2018-05-06T10:00:00.0Z</startTime>
                <endTime>2018-05-06T20:00:00.0Z</endTime>
            </timePeriod>
            <lowerBoundaryValue>
                <number>0</number>
            </lowerBoundaryValue>
            <upperBoundaryValue>
                <number>20</number>
                <scale>3</scale>
            </upperBoundaryValue>
        </tierBoundaryData>
        <tierBoundaryData>
            <boundaryId>2</boundaryId>
            <timePeriod>
                <startTime>2018-05-06T20:00:00.0Z</startTime>
                <endTime>2018-05-07T10:00:00.0Z</endTime>
            </timePeriod>
            <lowerBoundaryValue>
                <number>0</number>
            </lowerBoundaryValue>
            <upperBoundaryValue>
                <number>20</number>
                <scale>3</scale>
            </upperBoundaryValue>
        </tierBoundaryData>
        <tierBoundaryData>
            <boundaryId>3</boundaryId>
            <timePeriod>
                <startTime>2018-05-06T10:00:00.0Z</startTime>
                <endTime>2018-05-06T20:00:00.0Z</endTime>
            </timePeriod>
            <lowerBoundaryValue>
                <number>20</number>
                <scale>3</scale>
            </lowerBoundaryValue>
        </tierBoundaryData>
        <tierBoundaryData>
            <boundaryId>4</boundaryId>
            <timePeriod>
                <startTime>2018-05-06T20:00:00.0Z</startTime>
                <endTime>2018-05-07T10:00:00.0Z</endTime>
            </timePeriod>
            <lowerBoundaryValue>
                <number>20</number>
                <scale>3</scale>
            </lowerBoundaryValue>
        </tierBoundaryData>
    </tierBoundaryListData>
```

5.3.24.10 tierBoundaryDescriptionListData

5.3.24.10.1 General

The rather static part of the (tier) boundary is modelled in this function.



10115

10116 *Figure 191: tierBoundaryDescriptionListData function overview*

10117

10118 *5.3.24.10.2 Detailed description of elements*

Element	Type	Description
boundaryId	*Identifier "TierBoundaryIdType" (see* Table 339*).*	Primary identifier for the boundary.
boundaryType	*Union "*TierBoundaryTypeType*":* *- Enum (see* Table 301*):* TierBoundaryTypeEnumType - EnumExtendType (see section 3.10.1.5)	The type of the boundary.
validForTierId	*Identifier "TierIdType" (see* Table 339*).*	Identifier of the tier this boundary is valid for.
switchToTierIdWhenLower	*Identifier "TierIdType" (see* Table 339*).*	Identifier of the tier that will become valid if the according value falls below the element lowerBoundaryValue in function tierBoundaryListData.
switchToTierIdWhenHigher	*Identifier "TierIdType" (see* Table 339*).*	Identifier of the tier that will become valid if the according value rises above the element upperBoundaryValue in function tierBoundaryListData.
boundaryUnit	*Common data type "UnitOfMeasurementType". See section 3.10.1.17.*	A unit for the boundary.
label	*Common data type "LabelType". See section 3.10.1.2.*	A short label of the tier boundary.

description	*Common data type "DescriptionType". See section 3.10.1.3.*	Descriptive information on the tier boundary.

Table 300: tierBoundaryDescriptionListData function detailed description of elements

Enumeration **TierBoundaryTypeEnumType**:

Value	Description
powerBoundary	Provides boundaries for power.
energyBoundary	Provides boundaries for energy.
countBoundary	Provides boundaries for counts.

Table 301: Enumeration TierBoundaryTypeEnumType

5.3.24.10.3 Available selectors

- boundaryId
- boundaryType

5.3.24.10.4 Examples

5.3.24.10.4.1 Read-reply

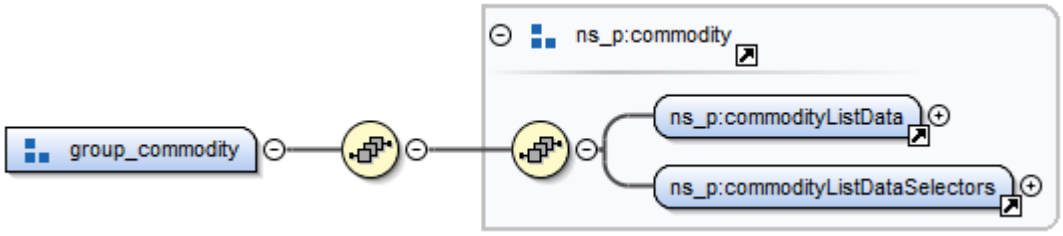
If one is interested in the rather static parts of all tier boundaries, he could send a standard read command to the server. The reply could look like this:

```
<tierBoundaryDescriptionListData>
    <tierBoundaryDescriptionData>
        <boundaryId>1</boundaryId>
        <boundaryType>powerBoundary</boundaryType>
        <validForTierId>1</validForTierId>
        <switchToTierIdWhenHigher>3</switchToTierIdWhenHigher>
        <boundaryUnit>W</boundaryUnit>
    </tierBoundaryDescriptionData>
    <tierBoundaryDescriptionData>
        <boundaryId>2</boundaryId>
        <boundaryType>powerBoundary</boundaryType>
        <validForTierId>2</validForTierId>
        <switchToTierIdWhenHigher>4</switchToTierIdWhenHigher>
        <boundaryUnit>W</boundaryUnit>
    </tierBoundaryDescriptionData>
    <tierBoundaryDescriptionData>
        <boundaryId>3</boundaryId>
        <boundaryType>powerBoundary</boundaryType>
        <validForTierId>3</validForTierId>
        <switchToTierIdWhenLower>1</switchToTierIdWhenLower>
        <boundaryUnit>W</boundaryUnit>
    </tierBoundaryDescriptionData>
    <tierBoundaryDescriptionData>
        <boundaryId>4</boundaryId>
        <boundaryType>powerBoundary</boundaryType>
        <validForTierId>4</validForTierId>
        <switchToTierIdWhenLower>2</switchToTierIdWhenLower>
        <boundaryUnit>W</boundaryUnit>
    </tierBoundaryDescriptionData>
</tierBoundaryDescriptionListData>
```


10161

10162 ***5.3.24.11 Sub-class Commodity***

10163 Commodities (like "electricity", "water", etc.) are described within this sub-class.



10164

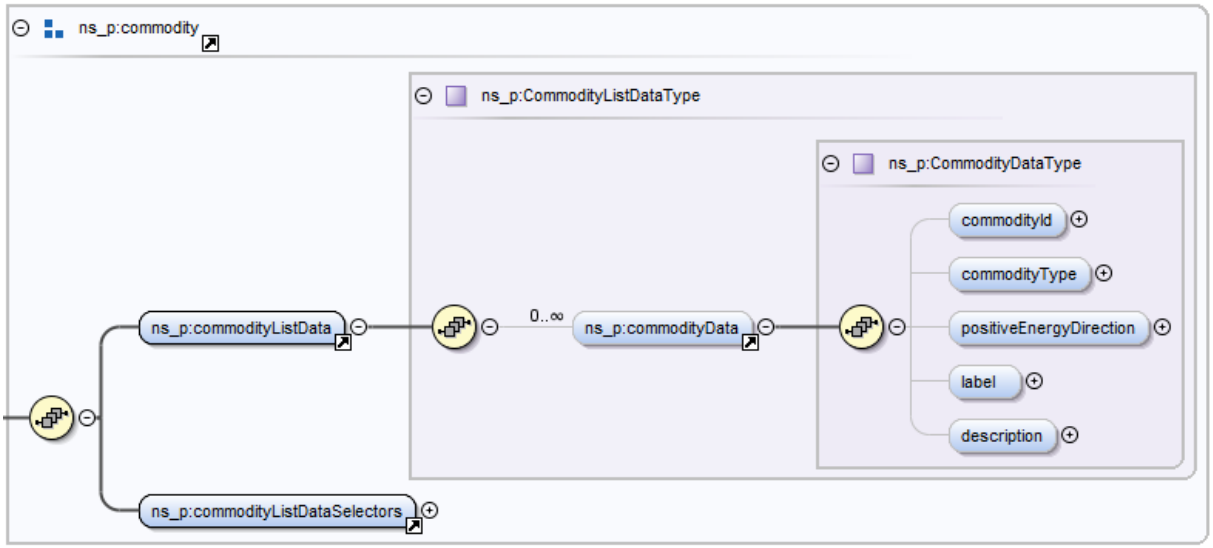
10165 *Figure 192: TariffInformation, sub-class Commodity, function-group overview*

10166

10167 ***5.3.24.12 commodityListData***

10168 *5.3.24.12.1 General*

10169 A tariff may be linked to a commodity. The commodity information is modelled with this function.



10170

10171 *Figure 193: commodityListData function overview*

10172

10173 *5.3.24.12.2 Detailed description of elements*

Element	Type	Description
commodityId	*Identifier "CommodityIdType" (see* Table 339*).*	Primary identifier for the commodity.
commodityType	*Common data type "CommodityTypeType". See section 3.10.1.9.*	The type of the commodity
positiveEnergyDirection	*Common data type "EnergyDirectionType". See section 3.10.1.13.*	States which direction counts positives values for this commodity.
label	*Common data type "LabelType". See section 3.10.1.2.*	A short label of the commodity.

description	*Common data type "DescriptionType". See section 3.10.1.3.*	Descriptive information on the commodity.

Table 302: commodityListData function detailed description of elements

5.3.24.12.3 Available selectors

- commodityId
- commodityType

5.3.24.12.4 Examples

5.3.24.12.4.1 Read-reply

If one is interested in the information about all commodities, he could send a standard read command to the server. The reply could look like this:

```
<commodityListData>
    <commodityData>
        <commodityId>1</commodityId>
        <commodityType>electricity</commodityType>
        <positiveEnergyDirection>consume</positiveEnergyDirection>
    </commodityData>
</commodityListData>
```

5.3.24.13 Sub-class Tier

Tiers are relevant for tariffs as they allow to different levels within the tariff (e.g. time based ("day" and "night" prices), boundary based ("the more you consume, the more you pay")).

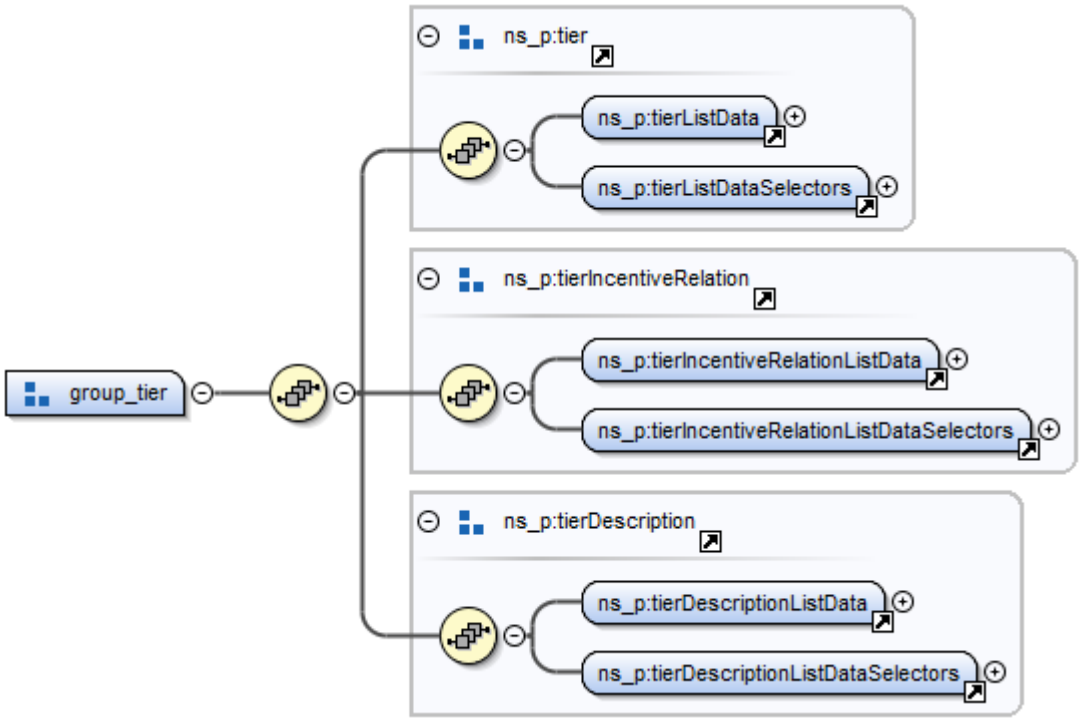


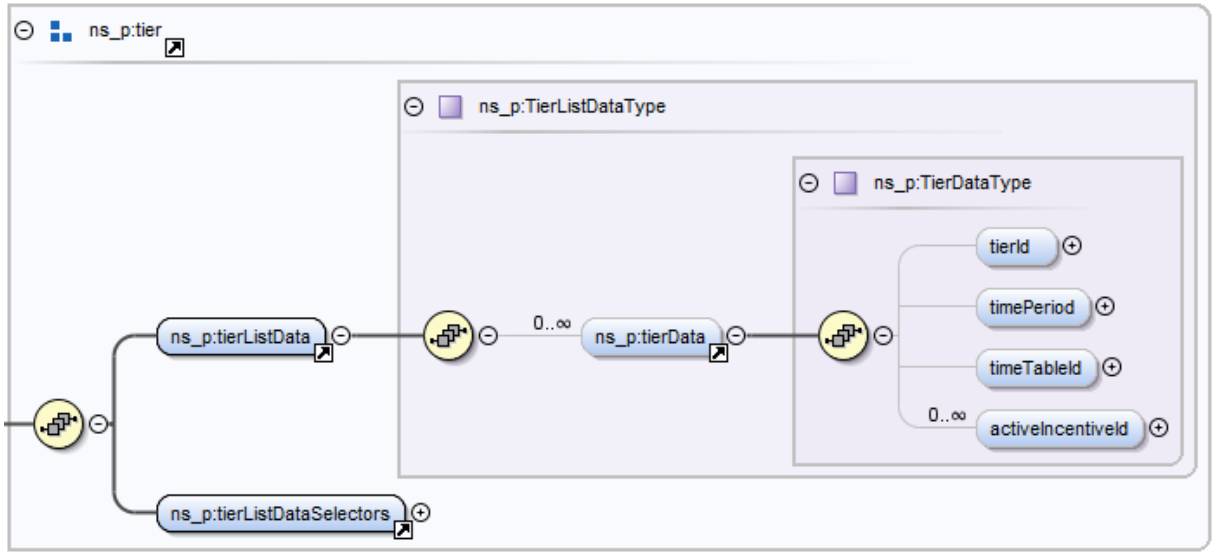
Figure 194: TariffInformation, sub-class Tier, function-group overview

10197

10198 ***5.3.24.14 tierListData***

10199 *5.3.24.14.1 General*

10200 The dynamic parts (time dependency and active incentives) of the tiers is described within this
10201 function.



10202

10203 *Figure 195: tierListData function overview*

10204

10205 *5.3.24.14.2 Detailed description of elements*

Element	Type	Description
tierId	*Identifier "TierIdType " (see* Table 339*).*	Primary identifier for the tier.
timePeriod	*Common data type "TimePeriodType". See section 3.10.2.1.*	If the tier is only valid for some time period, this can be denoted here.
timeTableId	*Identifier "TimeTableIdType" (see* Table 339*).*	The foreign key of a time table can be set here if the tier is only valid during the given start and end times there.
activeIncentiveId (list)	*Identifier "IncentiveIdType" (see* Table 339*).*	Foreign key(s) of the currently active incentive(s) for this tier.

10206 *Table 303: tierListData function detailed description of elements*

10207

10208 *5.3.24.14.3 Available selectors*

10209 - tierId
10210 - activeIncentiveId

10211

10212 *5.3.24.14.4 Examples*

10213 *5.3.24.14.4.1 Read-reply*

10214 If one is interested in the dynamic parts of all tiers, he could send a standard read command to the
10215 server. The reply could look like this:

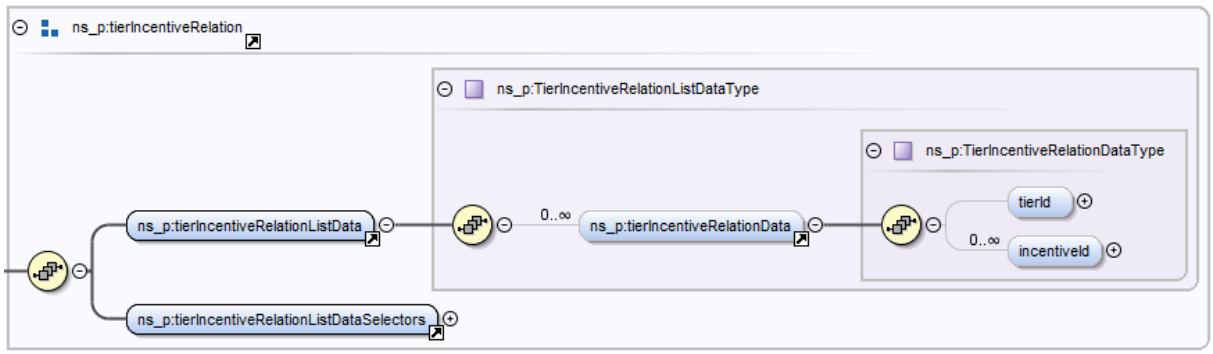
```
10216 <tierListData>
10217     <tierData>
10218         <tierId>1</tierId>
10219         <timeTableId>1</timeTableId>
10220         <activeIncentiveId>1</activeIncentiveId>
10221     </tierData>
10222     <tierData>
10223         <tierId>2</tierId>
10224         <timeTableId>2</timeTableId>
10225         <activeIncentiveId>1</activeIncentiveId>
10226     </tierData>
10227     <tierData>
10228         <tierId>3</tierId>
10229         <timeTableId>1</timeTableId>
10230         <activeIncentiveId>1</activeIncentiveId>
10231     </tierData>
10232     <tierData>
10233         <tierId>4</tierId>
10234         <timeTableId>2</timeTableId>
10235         <activeIncentiveId>1</activeIncentiveId>
10236     </tierData>
10237 </tierListData>
```

10238

10239 ***5.3.24.15 tierIncentiveRelationListData***

10240 *5.3.24.15.1 General*

10241 Which incentives can be used for specific tiers is modelled in the tierIncentiveRelationListData
10242 function.



10243

10244 *Figure 196: tierIncentiveRelationListData function overview*

10245

10246 *5.3.24.15.2 Detailed description of elements*

Element	Type	Description
tierId	*Identifier "TierIdType " (see* Table 339*).*	Primary identifier for the tier.
incentiveId (list)	*Identifier "IncentiveIdType" (see* Table 339*).*	Foreign key(s) of the related incentive(s) for this tier.

Table 304: tierIncentiveRelationListData function detailed description of elements

5.3.24.15.3 Available selectors

- tierId
- incentiveId

5.3.24.15.4 Examples

5.3.24.15.4.1 Read-reply

If one is interested in the possible incentives of all tiers, he could send a standard read command to the server. The reply could look like this:

```
<tierIncentiveRelationListData>
    <tierIncentiveRelationData>
        <tierId>1</tierId>
        <incentiveId>1</incentiveId>
    </tierIncentiveRelationData>
    <tierIncentiveRelationData>
        <tierId>2</tierId>
        <incentiveId>1</incentiveId>
    </tierIncentiveRelationData>
    <tierIncentiveRelationData>
        <tierId>3</tierId>
        <incentiveId>1</incentiveId>
    </tierIncentiveRelationData>
    <tierIncentiveRelationData>
        <tierId>4</tierId>
        <incentiveId>1</incentiveId>
    </tierIncentiveRelationData>
</tierIncentiveRelationListData>
```

5.3.24.16 tierDescriptionListData

5.3.24.16.1 General

The rather static part of the tiers is modelled with this function.

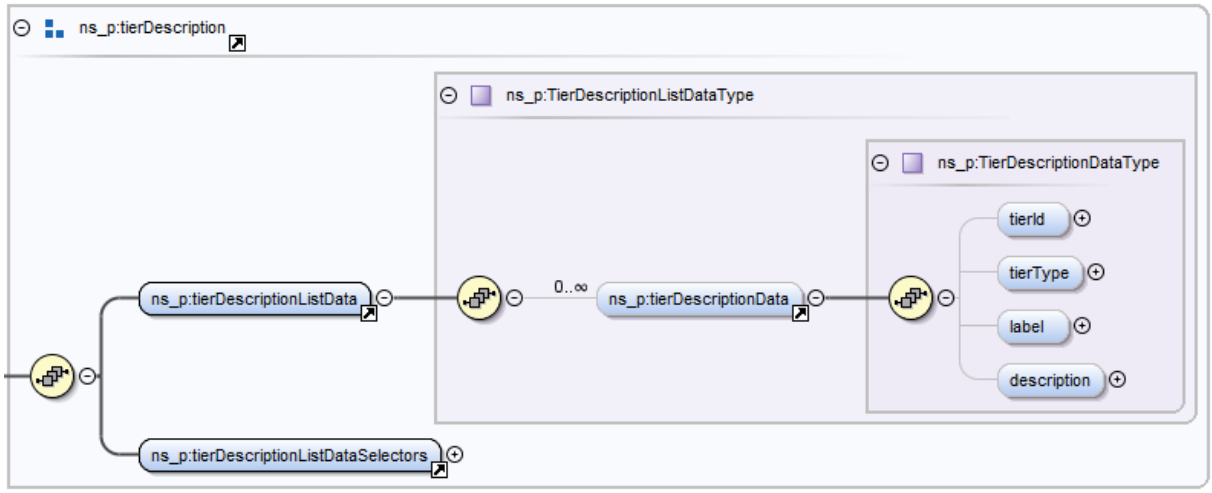


Figure 197: tierDescriptionListData function overview

5.3.24.16.2 Detailed description of elements

Element	Type	Description
tierId	*Identifier "TierIdType " (see* Table 339*).*	Primary identifier for the tier.
tierType	*Union "*TierTypeType*":* *- Enum (see* Table 306*):* TierTypeEnumType - EnumExtendType (see section 3.10.1.5)	The type of the tier.
label	*Common data type "LabelType". See section 3.10.1.2.*	A short label of the tier.
description	*Common data type "DescriptionType". See section 3.10.1.3.*	Descriptive information on the tier.

Table 305: tierDescriptionListData function detailed description of elements

Enumeration **TierTypeEnumType**:

Value	Description
fixedCost	The fixed cost have to be payed, independent from the consumed energy.
dynamicCost	The dynamic cost has a (linear) dependency to the consumed energy.

Table 306: Enumeration TierTypeEnumType

5.3.24.16.3 Available selectors

- tierId
- tierType

5.3.24.16.4 Examples

5.3.24.16.4.1 Read-reply

If one is interested in the rather static parts of all tiers, he could send a standard read command to the server. The reply could look like this:

```
<tierDescriptionListData>
```



```
    <tierDescriptionData>
        <tierId>1</tierId>
        <tierType>dynamicCost</tierType>
    </tierDescriptionData>
    <tierDescriptionData>
        <tierId>2</tierId>
        <tierType>dynamicCost</tierType>
    </tierDescriptionData>
    <tierDescriptionData>
        <tierId>3</tierId>
        <tierType>dynamicCost</tierType>
    </tierDescriptionData>
    <tierDescriptionData>
        <tierId>4</tierId>
        <tierType>dynamicCost</tierType>
    </tierDescriptionData>
</tierDescriptionListData>
```

5.3.24.17 Sub-class Incentive

The actual costs are modelled as so-called incentives (e.g. real monetary costs, "co2Emission", etc.).

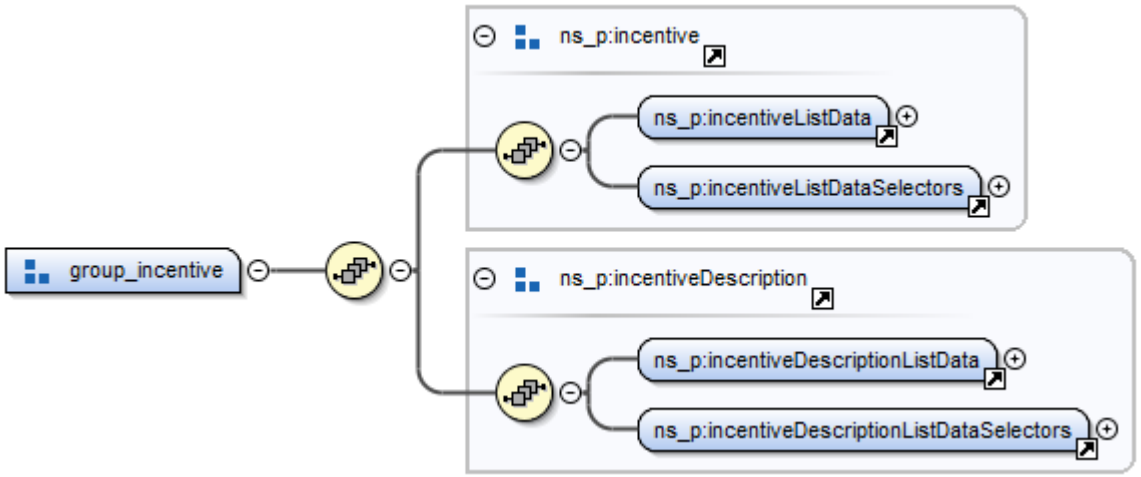


Figure 198: TariffInformation, sub-class Incentive, function-group overview

5.3.24.18 incentiveListData

5.3.24.18.1 General

The value of the costs, together with a time dependency (if needed), is part of this dynamic incentive function.

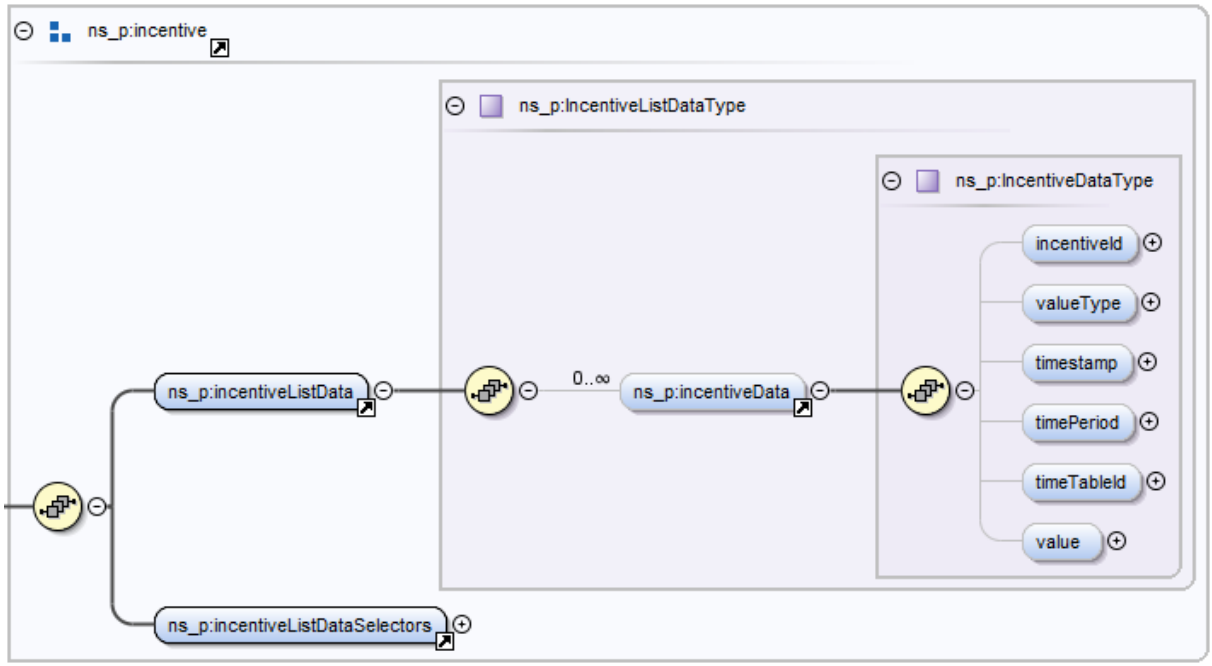


Figure 199: incentiveListData function overview

5.3.24.18.2 Detailed description of elements

Element	Type	Description
incentiveId	*Identifier "IncentiveIdType" (see* Table 339*).*	Primary identifier for the incentive.
valueType	*Union "*IncentiveValueTypeType*":* *- Enum (see* Table 308*):* IncentiveValueTypeEnumType - EnumExtendType (see section 3.10.1.5)	A type for the value (e.g. "value", "minValue", etc.).
timestamp	*Common data type "AbsoluteOrRelativeTimeType". See section 3.10.2.3.*	A timestamp for the value information.
timePeriod	*Common data type "TimePeriodType". See section 3.10.2.1.*	If there exists a time dependency for which the incentive value is valid, this can be denoted here.
timeTableId	*Identifier "TimeTableIdType" (see* Table 339*).*	The foreign key of a time table can be set here if the incentive value depends on the given start and end times there.
value	*Common data type "ScaledNumberType". See section 3.10.1.8.*	The actual value for the incentive costs.

Table 307: incentiveListData function detailed description of elements

Enumeration **IncentiveValueTypeEnumType**:

Value	Description
value	Used for normal cost values for an incentive.
averageValue	Used if only an average cost value can be stated for an incentive.
minValue	Used for the minimum cost value of an incentive.
maxValue	Used for the maximum cost value of an incentive.

Table 308: Enumeration IncentiveValueTypeEnumType

5.3.24.18.3 Available selectors

- incentiveId
- timestamp
- timestampInterval

5.3.24.18.4 Examples

5.3.24.18.4.1 Read-reply

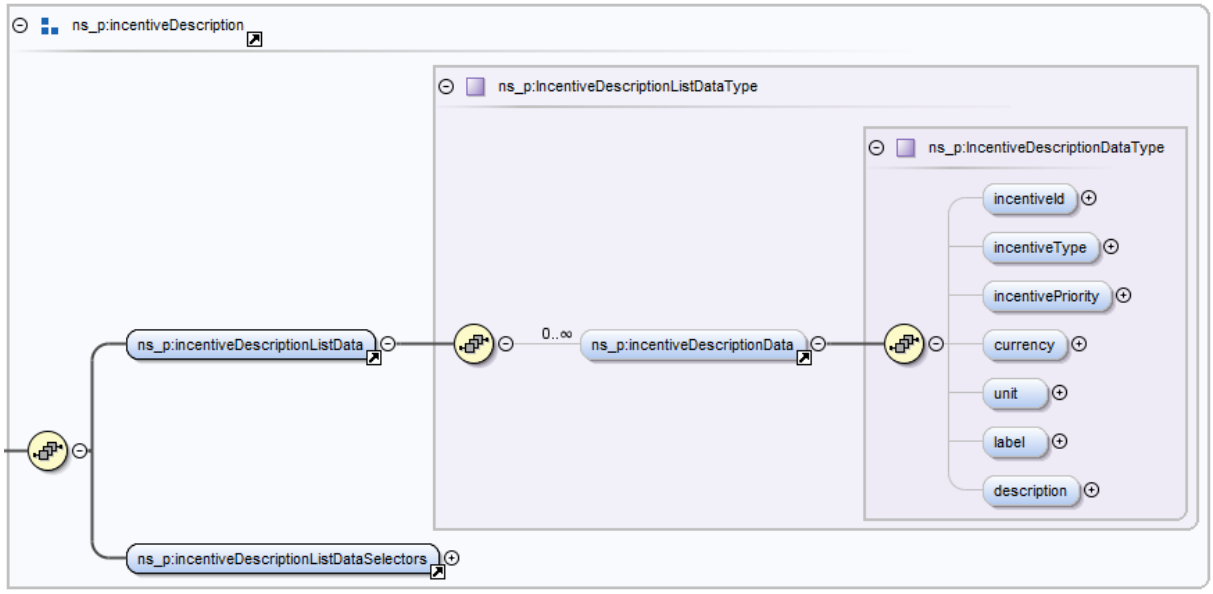
If one is interested in the dynamic parts of all incentives, he could send a standard read command to the server. The reply could look like this:

```
<incentiveListData>
    <incentiveData>
        <incentiveId>1</incentiveId>
        <valueType>value</valueType>
        <timestamp>2018-05-06T09:00:00.0Z</timestamp>
        <timePeriod>
            <startTime>2018-05-06T10:00:00.0Z</startTime>
            <endTime>2018-05-07T10:00:00.0Z</endTime>
        </timePeriod>
        <value>
            <number>28</number>
            <scale>-2</scale>
        </value>
    </incentiveData>
</incentiveListData>
```

5.3.24.19 incentiveDescriptionListData

5.3.24.19.1 General

The rather static information about the incentives is modelled with this function.



10359

10360 *Figure 200: incentiveDescriptionListData function overview*

10361

10362 *5.3.24.19.2 Detailed description of elements*

Element	Type	Description
incentiveId	*Identifier "IncentiveIdType" (see* Table 339*).*	Primary identifier for the incentive.
incentiveType	*Union* "IncentiveTypeType*":* *- Enum (see* Table 310*):* IncentiveTypeEnumType - EnumExtendType (see section 3.10.1.5)	The type of the incentive.
incentivePriority	*Simple type "IncentivePriorityType" (restriction of xs:unsignedInt)*	A priority can be stated for an incentive.
currency	*Common data type "CurrencyType". See section 3.10.1.19.*	The currency for the incentive costs.
unit	*Common data type "UnitOfMeasurementType". See section 3.10.1.17.*	A unit for the measured commodity this incentives models costs for.
label	*Common data type "LabelType". See section 3.10.1.2.*	A short label of the incentive.
description	*Common data type "DescriptionType". See section 3.10.1.3.*	Descriptive information on the incentive.

10363 *Table 309: incentiveDescriptionListData function detailed description of elements*

10364 Enumeration **IncentiveTypeEnumType**:

Value	Description
absoluteCost	Absolute costs with a currency and a unit. The total costs can be calculated.
relativeCost	Relative costs that cannot be interpreted as some monetary costs.
renewableEnergyPercentage	Percentage of renewable energy of the total obtained energy. This is an ecological incentive that allows optimizing consumption of renewable energy instead of costs.
co2Emission	CO2 emission related to the obtained energy. This is an ecological incentive that allows optimizing the CO2 emission instead of costs.

Table 310: Enumeration IncentiveTypeEnumType

5.3.24.19.3 Available selectors

- incentiveId
- incentiveType

5.3.24.19.4 Examples

5.3.24.19.4.1 Read-reply

If one is interested in the rather static parts of all incentives, he could send a standard read command to the server. The reply could look like this:

```
<incentiveDescriptionListData>
    <incentiveDescriptionData>
        <incentiveId>1</incentiveId>
        <incentiveType>absoluteCost</incentiveType>
        <incentivePriority>1</incentivePriority>
        <currency>EUR</currency>
        <unit>Wh</unit>
    </incentiveDescriptionData>
</incentiveDescriptionListData>
```

5.3.25 TaskManagement

5.3.25.1 Introduction

The TaskManagement Class lists all "tasks" that are currently relevant on an entity. This enables one to have a quick look on the tasks of related classes, but not for their modification. Further information has to be gathered with the corresponding classes supporting the specific functionality.

In this version of the specification, tasks can be "jobs" like in DirectControl, PowerSequences / SmartEnergyManagementPs, etc. (see the related function groups below).

Remark: Future versions of this specification may add other kinds of tasks (i.e. tasks that are not covered by “jobs”).

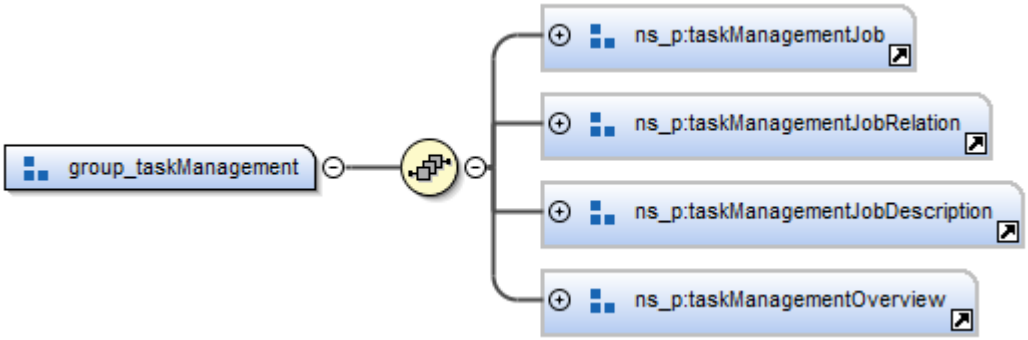
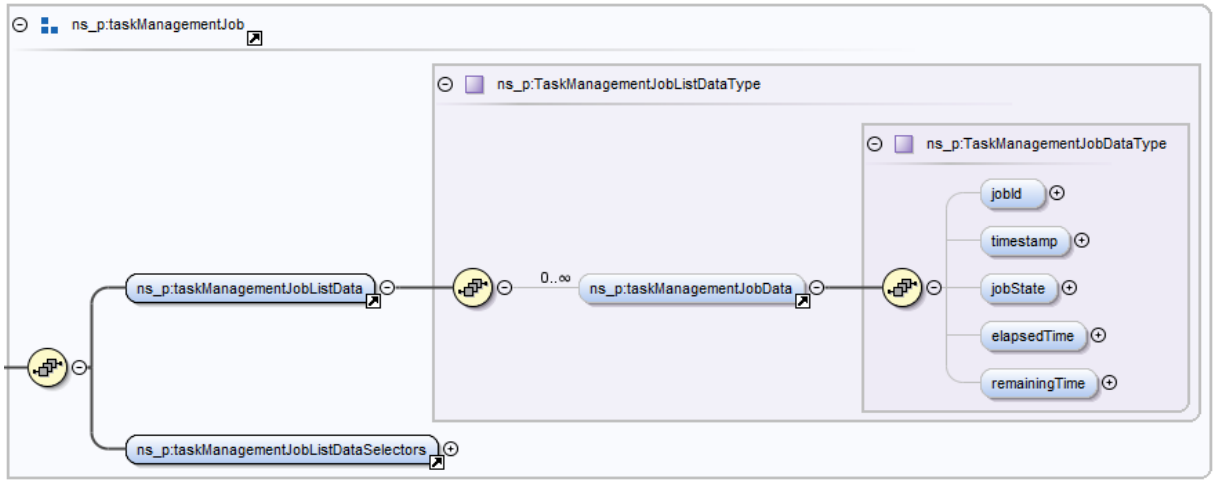


Figure 201: TaskManagement function-group overview

10397 ### *5.3.25.2 taskManagementJobListData*

10398 #### *5.3.25.2.1 General*

10399 All jobs that are available on the entity are listed here, together with their state and (if available) the
10400 elapsed and remaining time.



10401

10402 *Figure 202: taskManagementJobListData function overview*

10403

10404 #### *5.3.25.2.2 Detailed description of elements*

Element	Type	Description
jobId	*Identifier "TaskManagementJobIdType" (see* Table 339*).*	The *jobId* is used for indentifying different job information collected by this class.
timestamp	*Common data type "AbsoluteOrRelativeTimeType". See section 3.10.2.3.*	Point in time, this information was generated.
jobState	*Union "*TaskManagementJobStateType*":* *- Enum (see* Table 196*):* DirectControlActivityStateEnumType *- Enum (see* Table 215*):* HvacOverrunStatusEnumType *- Enum (see* Table 224*):* LoadControlEventStateEnumType *- Enum (see* Table 269*):* PowerSequenceStateEnumType - EnumExtendType (see section 3.10.1.5)	Current state of the job.
elapsedTime	xs:duration (W3C standard type)	The duration, this job was already active (normally, pause times are not counted).
remainingTime	xs:duration (W3C standard type)	The duration, this job will most likely be active before it ends.

10405 *Table 311: taskManagementJobListData function detailed description of elements*

10406

10407 #### *5.3.25.2.3 Available selectors*

10408 - jobId

- jobState

5.3.25.2.4 Examples

5.3.25.2.4.1 Read-reply

If one wants to request information about a specific job (jobId = "2"), he could sent the following read command:

```
<function>taskManagementJobListData</function>
<filter>
    <cmdControl>
        <partial/>
    </cmdControl>
    <taskManagementJobListDataSelectors>
        <jobId>2</jobId>
    </taskManagementJobListDataSelectors>
</filter>
<taskManagementJobListData/>
```

A response could be:

```
<function>taskManagementJobListData</function>
<filter>
    <cmdControl>
        <partial/>
    </cmdControl>
</filter>
<taskManagementJobListData>
    <taskManagementJobData>
        <jobId>2</jobId>
        <timestamp>2015-11-18T14:59:23.3Z</timestamp>
        <jobState>running</jobState>
        <elapsedTime>PT2H8M32S</elapsedTime>
        <remainingTime>PT21M28S</remainingTime>
    </taskManagementJobData>
</taskManagementJobListData>
```

5.3.25.3 taskManagementJobRelationListData

5.3.25.3.1 General

To identify where a job originates from, the *taskManagementJobRelationListData* function is used. If the related class allows more than one job the corresponding identifier(s) are denoted.

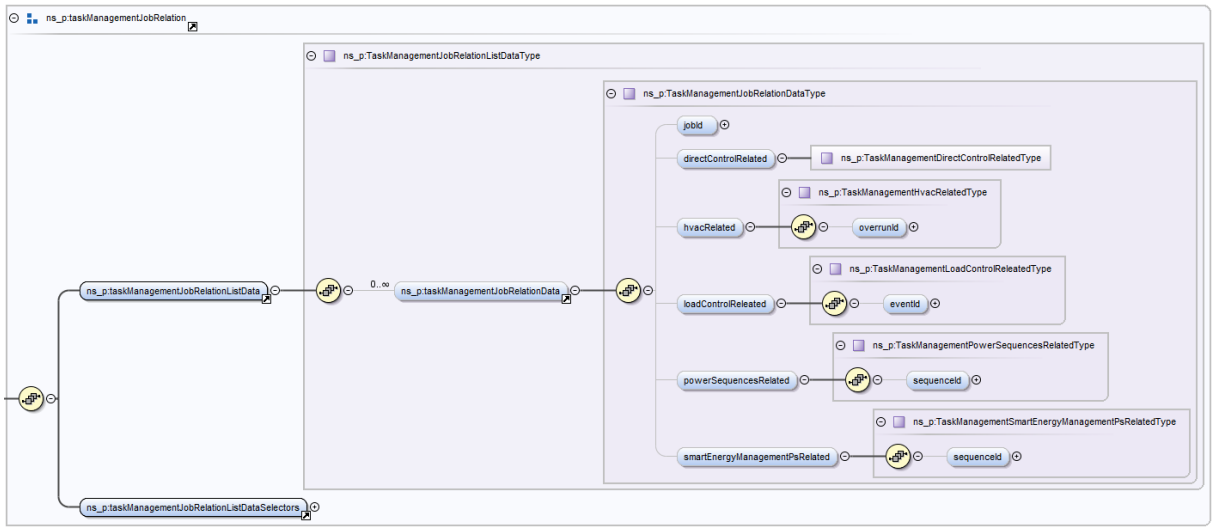


Figure 203: taskManagementJobRelationListData function overview

5.3.25.3.2 Detailed description of elements

Element	Type	Description
jobId	*Identifier "TaskManagementJobIdType" (see* Table 339*).*	The *jobId* is used for indentifying different job information collected by this class.
directControlRelated	*ComplexType "TaskManagementDirectControlRelatedType"*	Flag that indicates that the job is related to DirectControl.
hvacRelated	*ComplexType "TaskManagementHvacRelatedType"*	Flag that indicates that the job is related to HVAC.
hvacRelated. overrunId	*Identifier "HvacOverrunIdType" (see* Table 339*).*	Identifier for the related overrun.
loadControlReleated	*ComplexType "TaskManagementLoadControlReleatedType"*	Flag that indicates that the job is related to LoadControl.
loadControlReleated. eventId	*Identifier "LoadControlEventIdType" (see* Table 339*).*	Identifier for the related load control event.
powerSequencesRelated	*ComplexType "TaskManagementPowerSequencesRelatedType"*	Flag that indicates that the job is related to PowerSequences.
powerSequencesRelated. sequenceId	*Identifier "PowerSequenceIdType" (see* Table 339*).*	Identifier for the related power sequence.
smartEnergyManagementPs Related	*ComplexType "TaskManagementSmartEnergyManagementPsRelatedType"*	Flag that indicates that the job is related to SmartEnergyManagementPs.
smartEnergyManagementPs Related. sequenceId	*Identifier "PowerSequenceIdType" (see* Table 339*).*	Identifier for the related power sequence.

Table 312: taskManagementJobRelationListData function detailed description of elements

5.3.25.3.3 Available selectors

- jobId

5.3.25.3.4 Examples

5.3.25.3.4.1 Read-reply

If one requests all job relations, the reply could look like the following:

```
<taskManagementJobRelationListData
    <taskManagementJobRelationData>
        <jobId>1</jobId>
        <hvacRelated>
            <overrunId>3</overrunId>
        </hvacRelated>
    </taskManagementJobRelationData>
    <taskManagementJobRelationData>
        <jobId>2</jobId>
        <directControlRelated/>
    </taskManagementJobRelationData>
</taskManagementJobRelationListData>
```

5.3.25.4 taskManagementJobDescriptionListData

5.3.25.4.1 General

The more static information about the jobs are communicated with this function.

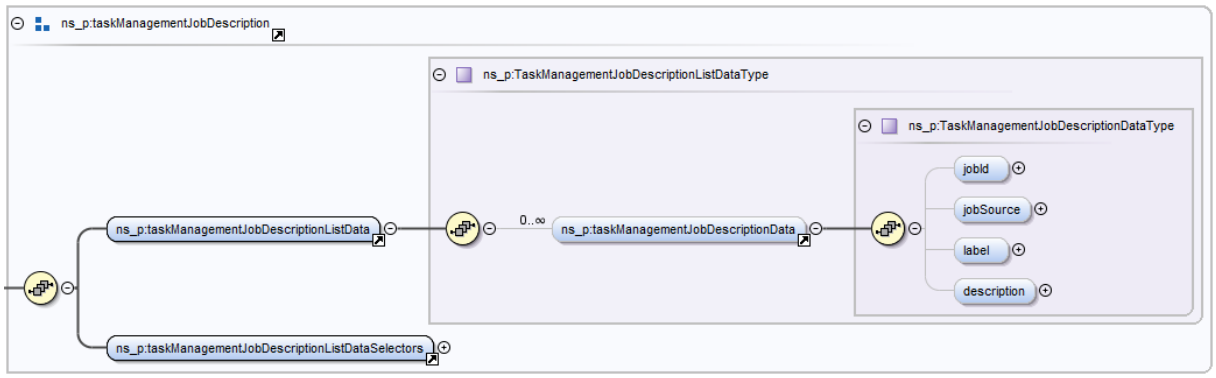


Figure 204: taskManagementJobDescriptionListData function overview

5.3.25.4.2 Detailed description of elements

Element	Type	Description
jobId	*Identifier "TaskManagementJobIdType" (see* Table 339*).*	The *jobId* is used for indentifying different job information collected by this class.
jobSource	*Union "*TaskManagementJobSourceType*":* *- Enum (see* Table 314*):* TaskManagementJobSourceEnumType - EnumExtendType (see section 3.10.1.5)	Specifies what kind of reason caused this job.

label	*Common data type "LabelType". See section 3.10.1.2.*	A short label of the job.
description	*Common data type "DescriptionType". See section 3.10.1.3.*	Descriptive information on the job.

Table 313: taskManagementJobDescriptionListData function detailed description of elements

Enumeration **TaskManagementJobSourceEnumType**:

Value	Description
internalMechanism	The job was started due to some device internal mechanism (e.g. a regular cleaning cycle).
userInteraction	The user activated some functionality directly via the UI of the device.
externalConfiguration	The job started, because it was triggered from some external device (e.g. an energy manager).

Table 314: Enumeration TaskManagementJobSourceEnumType

5.3.25.4.3 Available selectors

- jobId
- jobSource

5.3.25.4.4 Examples

5.3.25.4.4.1 Read-reply

If one requests all job descriptions, the reply could look like the following:

```
<taskManagementJobDescriptionListData>
    <taskManagementJobDescriptionData>
        <jobId>1</jobId>
        <jobSource>userInteraction</jobSource>
    </taskManagementJobDescriptionData>
    <taskManagementJobDescriptionData>
        <jobId>2</jobId>
        <jobSource>externalConfiguration</jobSource>
    </taskManagementJobDescriptionData>
</taskManagementJobDescriptionListData>
```

5.3.25.5 taskManagementOverviewData

5.3.25.5.1 General

A general overview is given with this function. It specifies whether a remote controlling is permitted at that moment and if currently jobs are active.

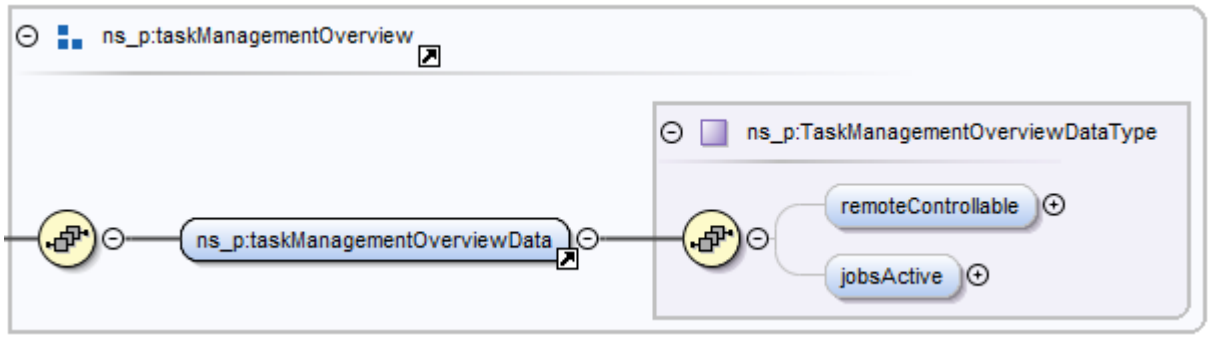


Figure 205: taskManagementOverviewData function overview

5.3.25.5.2 Detailed description of elements

Element	Type	Description
remoteControllable	xs:boolean (W3C standard type)	States if remote controlling of the device or entity is permitted.
jobsActive	xs: boolean (W3C standard type)	Denotes whether jobs are active on the device or entity or not.

Table 315: taskManagementOverviewData function detailed description of elements

5.3.25.5.3 Available selectors

None.

5.3.25.5.4 Examples

5.3.25.5.4.1 Read-reply

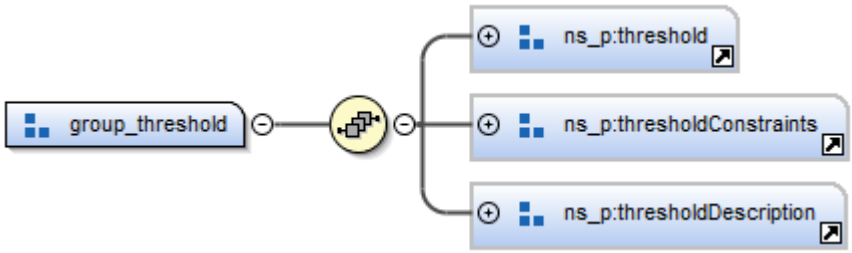
If one requests the task management overview, a response could be:

```
<taskManagementOverviewData>
    <remoteControllable>true</remoteControllable>
    <jobsActive>true</jobsActive>
</taskManagementOverviewData>
```

5.3.26 Threshold

5.3.26.1 Introduction

Thresholds are helpful in several kind of use cases. E.g. a device or application can be configured to do some specific behaviour if a measured value is above or below a configured threshold. Some measurands need to stay in a certain value range, to ensure proper and safe working of devices. Automatic alarms after reaching a threshold help reacting on the occurrence immediately.



10527

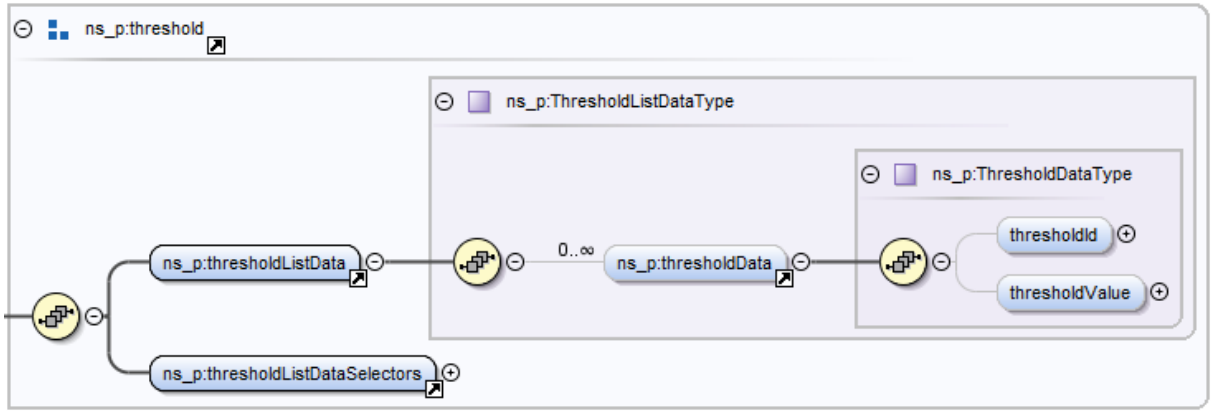
10528 *Figure 206: Threshold function-group overview*

10529

10530 ### *5.3.26.2 thresholdListData*

10531 #### *5.3.26.2.1 General*

10532 Each threshold (identified by its *thresholdId*) has a configured value. This function is used to read or
10533 write the actual value.



10534

10535 *Figure 207: thresholdListData function overview*

10536

10537 #### *5.3.26.2.2 Detailed description of elements*

Element	Type	Description
thresholdId	*Identifier "ThresholdIdType" (see* Table 339*).*	Identifier for the threshold.
thresholdValue	*Common data type "ScaledNumberType". See section 3.10.1.8.*	Value that should not be fallen below or exceeded (depends on the *thresholdType*, see section 5.3.26.4.2).

10538 *Table 316: thresholdListData function detailed description of elements*

10539

10540 #### *5.3.26.2.3 Available selectors*

10541 - thresholdId

10542

5.3.26.2.4 Examples

5.3.26.2.4.1 Read-reply

If one is interested in all thresholds, he could send the a standard read command to the device. The device could respond with something like this:

```
<thresholdListData>
    <thresholdData>
        <thresholdId>1</thresholdId>
        <thresholdValue>
            <number>155</number>
            <scale>-1</scale>
        </thresholdValue>
    </thresholdData>
    <thresholdData>
        <thresholdId>2</thresholdId>
        <thresholdValue>
            <number>23</number>
        </thresholdValue>
    </thresholdData>
</thresholdListData>
```

5.3.26.3 thresholdConstraintsListData

5.3.26.3.1 General

The value range from which a threshold value configuration may be choosen from may be restricted by a minimum and a maximum value. Between these, not every value, but only ones with a specific gap, may be used, specified with the step size.

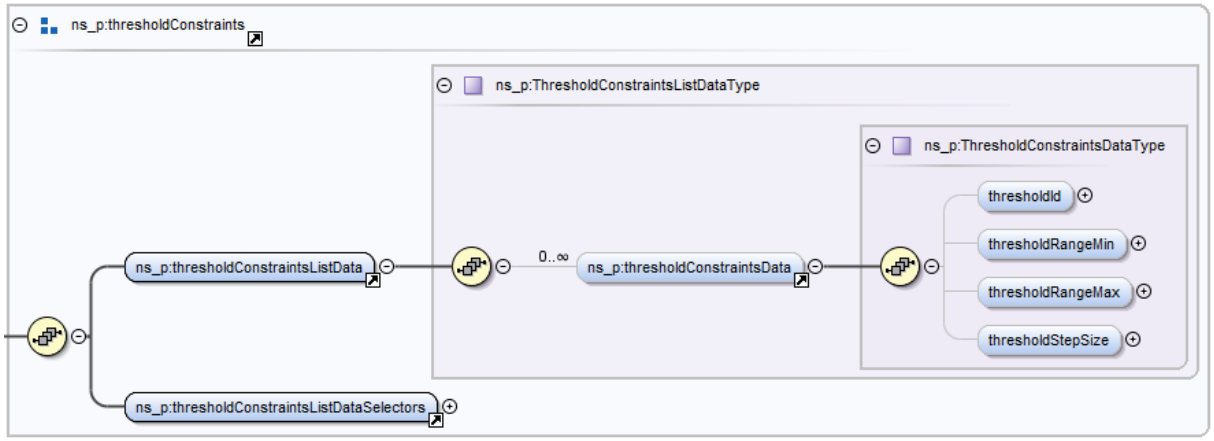


Figure 208: thresholdConstraintsListData function overview

5.3.26.3.2 Detailed description of elements

Element	Type	Description
thresholdId	*Identifier "ThresholdIdType" (see* Table 339*).*	Identifier for the threshold.
thresholdRangeMin	*Common data type "ScaledNumberType". See section 3.10.1.8.*	Minumum value that may be used to configure the element *thresholdValue* (see section 5.3.26.2.2).

thresholdRangeMax	*Common data type "ScaledNumberType". See section 3.10.1.8.*	Maximum value that may be used to configure the element *thresholdValue* (see section 5.3.26.2.2).
thresholdStepSize	*Common data type "ScaledNumberType". See section 3.10.1.8.*	Step size for the configuration of the element *thresholdValue* (see section 5.3.26.2.2).

Table 317: thresholdConstraintsListData function detailed description of elements

5.3.26.3.3 Available selectors

- thresholdId

5.3.26.3.4 Examples

5.3.26.3.4.1 Read-reply

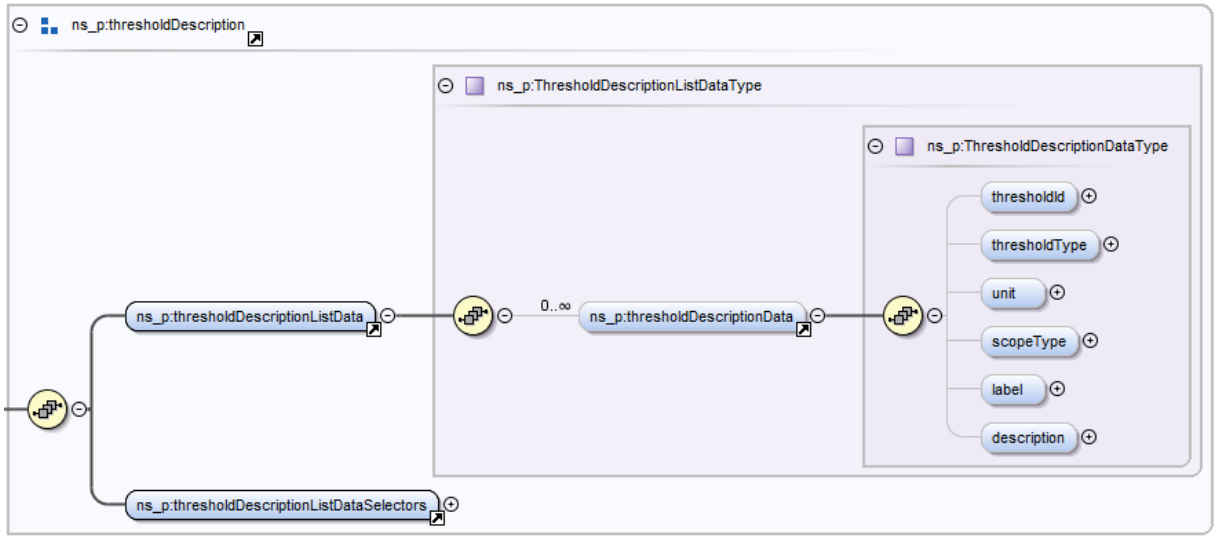
If one is interested in the constraints of all thresholds, he could send the a standard read command to the device. The device could respond with something like this:

```
<thresholdConstraintsListData>
    <thresholdConstraintsData>
        <thresholdId>1</thresholdId>
        <thresholdRangeMin>
            <number>5</number>
        </thresholdRangeMin>
        <thresholdRangeMax>
            <number>35</number>
        </thresholdRangeMax>
        <thresholdStepSize>
            <number>5</number>
            <scale>-1</scale>
        </thresholdStepSize>
    </thresholdConstraintsData>
    <thresholdConstraintsData>
        <thresholdId>2</thresholdId>
        <thresholdRangeMin>
            <number>5</number>
        </thresholdRangeMin>
        <thresholdRangeMax>
            <number>35</number>
        </thresholdRangeMax>
        <thresholdStepSize>
            <number>1</number>
        </thresholdStepSize>
    </thresholdConstraintsData>
</thresholdConstraintsListData>
```

5.3.26.4 thresholdDescriptionListData

5.3.26.4.1 General

The rather static information about thresholds is defined within this function.



10612

10613 *Figure 209: thresholdDescriptionListData function overview*

10614

10615 ### *5.3.26.4.2 Detailed description of elements*

Element	Type	Description
thresholdId	*Identifier "ThresholdIdType" (see* Table 339*).*	Identifier for the threshold.
thresholdType	*Union "*ThresholdTypeType*":* *- Enum (see* Table 319*):* ThresholdTypeEnumType - EnumExtendType (see section 3.10.1.5)	The type of the treshold.
unit	*Common data type "UnitOfMeasurementType". See section 3.10.1.17.*	The unit in which the value of the threshold is denoted.
scopeType	*Common data type "ScopeTypeType". See section 3.10.1.21.*	Specifies a more detailed meaning of the threshold.
label	*Common data type "LabelType". See section 3.10.1.2.*	A short label of the threshold.
description	*Common data type "DescriptionType". See section 3.10.1.3.*	Descriptive information on the threshold.

10616 *Table 318: thresholdDescriptionListData function detailed description of elements*

10617 Enumeration **ThresholdTypeEnumType**:

Value	Description
goodAbove	A positive reaction will follow after the value exceeds the threshold.
badAbove	A negative reaction will follow after the value exceeds the threshold.
goodBelow	A positive reaction will follow after the value falls below the threshold.
badBelow	A negative reaction will follow after the value falls below the threshold.

minValueThreshold	If this minimum threshold is underrun, some warning will be activated.
maxValueThreshold	If this maximum threshold is exceeded, some warning will be activated.
minValueThresholdExtreme	If this extreme minimum threshold is underrun, some important warning will be activated.
maxValueThresholdExtreme	If this extreme maximum threshold is exceeded, some important warning will be activated.
sagThreshold	If this sag threshold is underrun, some warning will be activated. Commonly used for electricity voltage sags.
swellThreshold	If this swell threshold is exceeded, some warning will be activated. Commonly used for electricity voltage swells.

Table 319: Enumeration ThresholdTypeEnumType

5.3.26.4.3 Available selectors

- thresholdId
- scopeType

5.3.26.4.4 Examples

5.3.26.4.4.1 Read-reply

If one is interested in the description of all thresholds, he could send the a standard read command to the device. The device could respond with something like this:

```
<thresholdDescriptionListData>
    <thresholdDescriptionData>
        <thresholdId>1</thresholdId>
        <thresholdType>badBelow</thresholdType>
        <unit>degC</unit>
        <label>Minimum temperature</label>
    </thresholdDescriptionData>
    <thresholdDescriptionData>
        <thresholdId>2</thresholdId>
        <thresholdType>badAbove</thresholdType>
        <unit>degC</unit>
        <label>Maximum temperature</label>
    </thresholdDescriptionData>
</thresholdDescriptionListData>
```

5.3.27 TimeInformation

5.3.27.1 Introduction

The exchange of time information is important for most smart grid and home automation systems. Please note the underlying revision of the SPINE data model does not specify a sophisticated synchronisation mechanism that includes a given precision. This is due to the complex of problems appearing with communication delays in the different technologies and hardware platforms. However, an application could measure all relevant delays and take care of a real synchronisation in the network using the *TimeInformation* class.

Independent of any synchronisation aspect the "owner concept" applies here as well: The "owner" of any given data (a measured value of a sensor, e.g.) applies its own SPINE time to its own data. This is relevant for the case of a substantial difference between the SPINE time of two nodes.

In order to avoid problems from such differences a typical sequence could be that a new device sends a *timeDistributorEnquiryCall* (with the *isTimeDistributor* element set to true) to all devices in the SPINE network. All time distributors would send a *timeDistributorData* instance as notification to the device. The new device would pick the time distributor with the highest priority and send a read command to the *timeInformationData* function of this distributor. The time distributor would reply with a *timeInformationData* containing its current time.

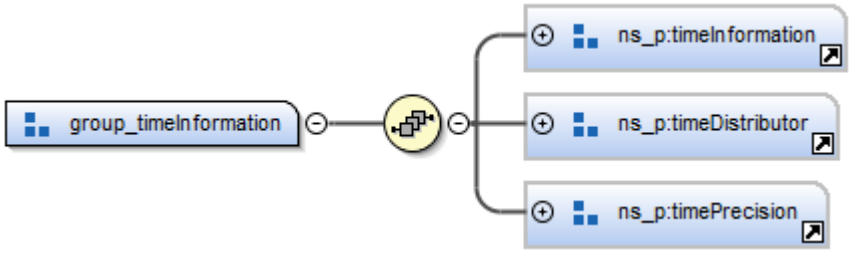


Figure 210: TimeInformation function-group overview (data)

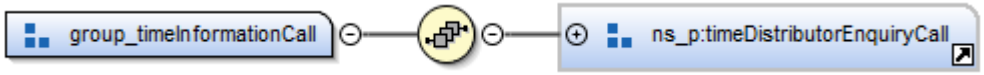


Figure 211: TimeInformation function-group overview (call)

5.3.27.2 timeInformationData

5.3.27.2.1 General

The *timeInformationData* command is used for exchanging the SPINE time.

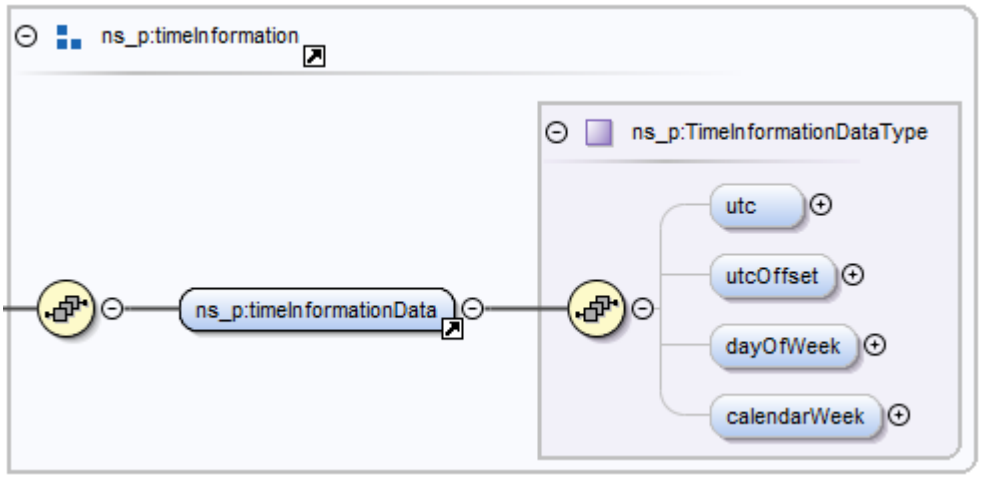


Figure 212: timeInformationData function overview

5.3.27.2.2 Detailed description of elements

Element	Type	Description

utc	xs:dateTime (W3C standard type)	The current time in UTC.
utcOffset	xs:duration (W3C standard type)	The configured offset to UTC (e.g. PT2H [= +2 hours] for Central European Summer Time (CEST)).
dayOfWeek	*Common data type "DayOfWeekType". See section 3.10.2.9.*	To model the actual day of week (Monday to Sunday) this element is used. It refers to the local time. If there is no utcOffset given, it is assumed, that the local time is equivalent to UTC-time.
calendarWeek	*Common data type "CalendarWeekType". See section 3.10.2.8.*	A value between 1 and 53 stating the current calendar week number. This element refers to the local time. If there is no utcOffset given, it is assumed, that the local time is equivalent to UTC-time.

Table 320: timeInformationData function detailed description of elements

5.3.27.2.3 Available selectors

None.

5.3.27.2.4 Examples

5.3.27.2.4.1 Read-reply

A *timeInformationData* command sent as reply to a read command on timeInformationData:

```
<timeInformationData>
    <utc>2013-07-15T12:00:00.0Z</utc>
    <utcOffset>PT2H</utcOffset>
    <dayOfWeek>tuesday</dayOfWeek>
    <calendarWeek>17</calendarWeek>
</timeInformationData>
```

5.3.27.3 timeDistributorData

5.3.27.3.1 General

One or some devices in a SPINE network should act as time distributor. The information thereof and the dedicated priority can be modelled with the *timeDistributorData* command.

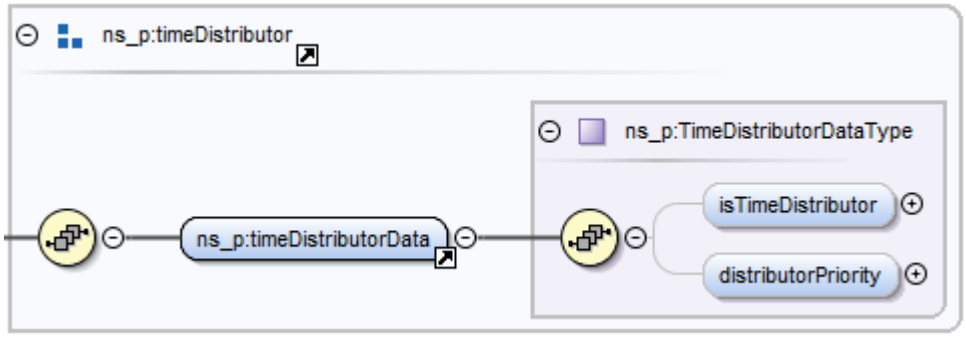


Figure 213: timeDistributorData function overview

10693

10694 *5.3.27.3.2 Detailed description of elements*

Element	Type	Description
isTimeDistributor	xs:boolean (W3C standard type)	Denotes whether a device can act as a time distributor in the SPINE network or not.
distributorPriority	xs:unsignedInt (W3C standard type)	A (configured) priority of the time distributor.

10695 *Table 321: timeDistributorData function detailed description of elements*

10696

10697 *5.3.27.3.3 Available selectors*

10698 None.

10699

10700 *5.3.27.3.4 Examples*

10701 *5.3.27.3.4.1 Notify*

10702 Example of a notification from a device, stating that it can act as a time distributor with a priority of
10703 "2":

```
10704 <timeDistributorData>
10705     <isTimeDistributor>true</isTimeDistributor>
10706     <distributorPriority>2</distributorPriority>
10707 </timeDistributorData>
```

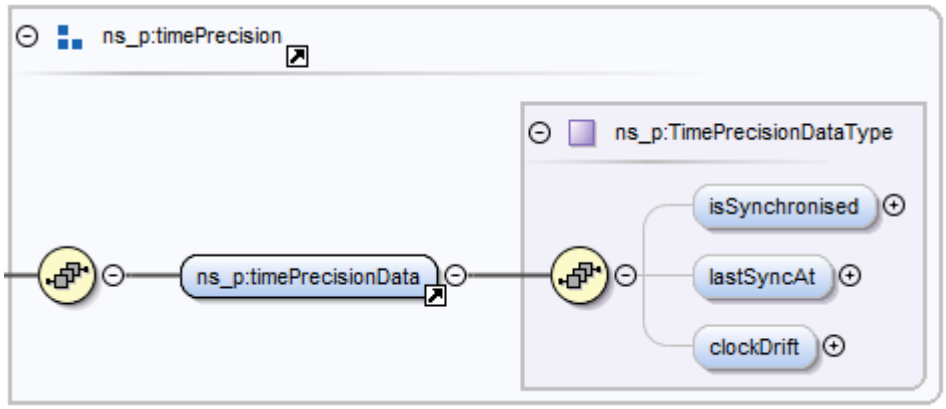
10708 Such notification could typically be send from a device that previously received a
10709 *timeDistributorEnquiryCall*.

10710

10711 ***5.3.27.4 timePrecisionData***

10712 *5.3.27.4.1 General*

10713 Information regarding the precision of the time of a device can be modelled using *timePrecisionData*.



10714

10715 *Figure 214: timePrecisionData function overview*

10716

10717 *5.3.27.4.2 Detailed description of elements*

Element	Type	Description
isSynchronised	xs:boolean (W3C standard type)	This is a Boolean value, stating whether the device's SPINE time information is synchronized with the SPINE network or not, meaning that it could receive the SPINE time from a time distributor and has still a “sufficiently precise time”.
lastSyncAt	xs:dateTime (W3C standard type)	This is the timestamp of the last synchronisation with a time distributor, or, in case of a time distributor itself, the timestamp of the last official date and time information (e.g. from an NTP server).
clockDrift	xs:integer (W3C standard type)	The drift of the internal clock (in seconds per day) is indicated by this element.

10718 *Table 322: timePrecisionData function detailed description of elements*

10719

10720 *5.3.27.4.3 Available selectors*

10721 None.

10722

10723 *5.3.27.4.4 Examples*

10724 *5.3.27.4.4.1 Read-reply*

10725 We assume a device had a synchronization with a time distributor more than two days ago. We also
10726 assume the device has a clock drift of 5 seconds per day and only 10 seconds deviation are
10727 acceptable to fulfil the network synchronization criteria. Then, if the device is requested for its
10728 current *timePrecisionData* information it could send the following reply:

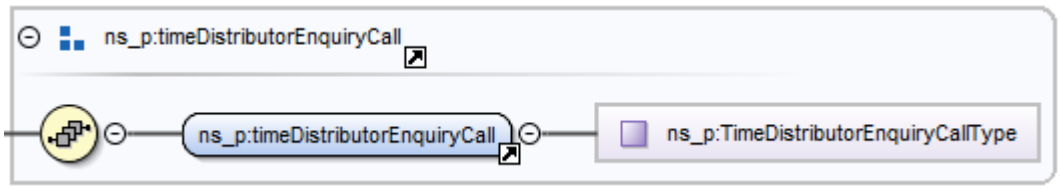
```
10729 <timePrecisionData>
10730     <isSynchronised>false</isSynchronised>
10731     <lastSyncAt>2013-07-14T15:00:00.0Z</lastSyncAt>
10732     <clockDrift>5</clockDrift>
10733 </timePrecisionData>
```

10734

10735 ***5.3.27.5 timeDistributorEnquiryCall***

10736 *5.3.27.5.1 General*

10737 This function can be used for finding all time distributors in the network. Therefore, it can be sent to
10738 all devices in the network that have their *isTimeDistributor* element in the function *timeDistributor*
10739 (see section 5.3.27.3) set to true.



10740

10741 *Figure 215: timeDistributorEnquiryCall function overview*

10742

10743 *5.3.27.5.2 Detailed description of elements*
10744 In this version of the specification the function timeDistributorEnquiryCall does not contain child
10745 elements.

10746

10747 *5.3.27.5.3 Available selectors*
10748 None.

10749

10750 *5.3.27.5.4 Examples*

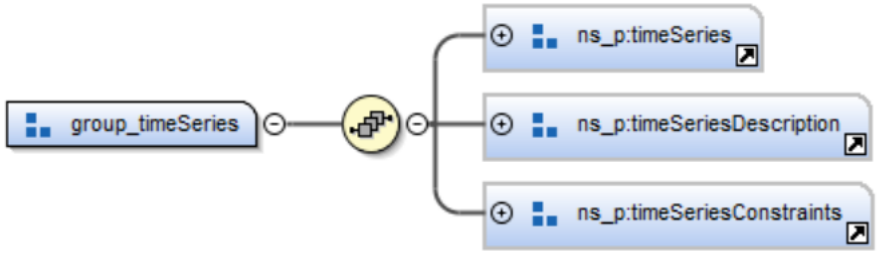
10751 *5.3.27.5.4.1 Call*
10752 A *timeDistributorEnquiryCall* instance, sent as call (the only classifier allowed for this function):

10753
```
<timeDistributorEnquiryCall/>
```

10754

10755 **5.3.28 TimeSeries**

10756 ***5.3.28.1 Introduction***
10757 TimeSeries is a versatile class that allows to express any kind of time dependent values as a time
10758 series of values. It can be used to express electricity related values like power constraints or power
10759 consumption over time, but also other values that are not electricity related like temperature or
10760 humidity over time.



10761

10762 *Figure 216: TimeSeries function-group overview*

10763

10764 ***5.3.28.2 timeSeriesListData***

10765 *5.3.28.2.1 General*
10766 The timeSeriesListData function is used to describe the value of a timeSeries over time.

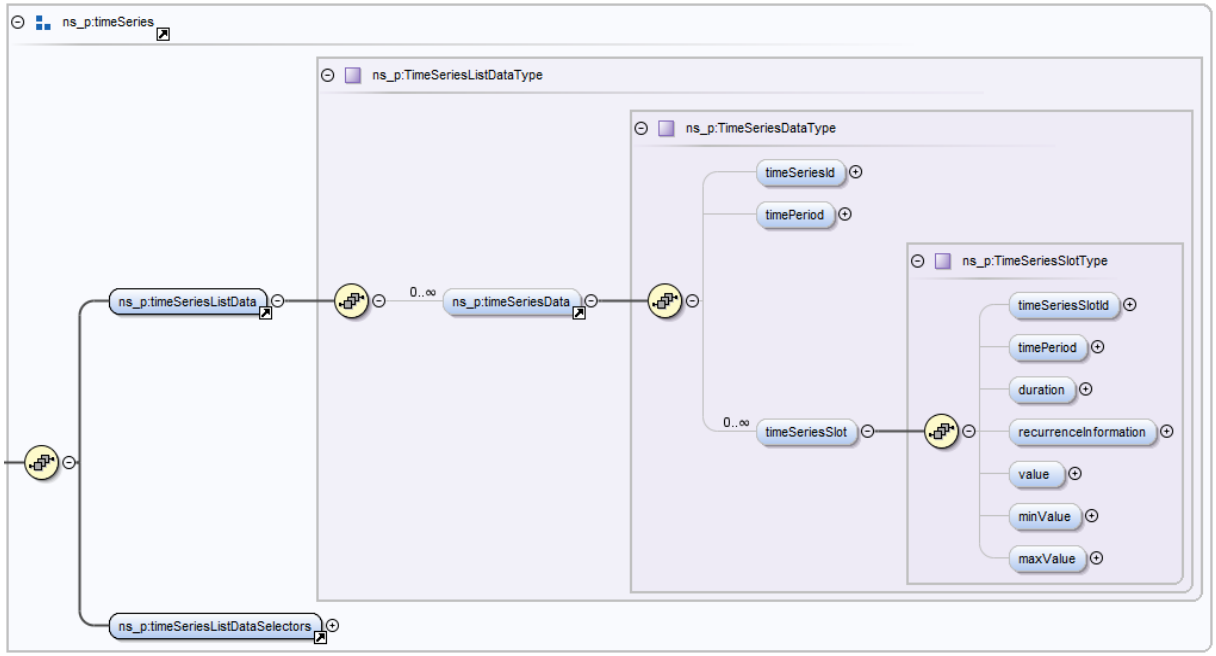


Figure 217: timeSeriesListData function overview

5.3.28.2.2 Detailed description of elements

Element	Type	Description
timeSeriesId	*Identifier "TimeSeriesIdType" (see* Table 339*).*	Allows the identification of a timeSeries. Allows linking of the different functions to the same timeSeries. Allows linking of the timeSeries within other Features that are placed in the same entity.
timePeriod	*Common data type "TimePeriodType". See section 3.10.2.1.*	Time period of the complete timeSeries.
timeSeriesSlot	*Complex type "TimeSeriesSlotType". See* Table 324*.*	Allows to define slots of a timeSeries.

Table 323: timeSeriesListData function detailed description of elements

Complex type **TimeSeriesSlotType**:

Element	Type	Description
timeSeriesSlotId	*Identifier "TimeSeriesSlotIdType" (see* Table 339*).*	Allows identification of a timeSeriesSlot within a timeSeries.
timePeriod	*Common data type "TimePeriodType". See section 3.10.2.1.*	Allows to define a timePeriod of a timeSeriesSlot and to model time gaps inbetween slots.
duration	xs:duration (W3C standard type)	Allows to define a duration of a timeSeriesSlot.

recurrenceInformation	*Common data type "RecurrenceInformationType". See section 3.10.2.14.*	If the time series slot is a recurring one, the corresponding information can be found within this element and its sub-elements.
value	*Common data type "ScaledNumberType". See section 3.10.1.8.*	Defines the expected value during a timeSeriesSlot.
minValue	*Common data type "ScaledNumberType". See section 3.10.1.8.*	Defines a lower value limit
maxValue	*Common data type "ScaledNumberType". See section 3.10.1.8.*	Defines a upper value limit

Table 324: Complex type TimeSeriesSlotType

5.3.28.2.3 Available selectors

- timeSeriesId
- timeSeriesSlotId

5.3.28.2.4 Examples

5.3.28.2.4.1 Read-reply

To read the time series entry with the *timeSeriesId* "3", the following read command can be send:

```
<function>timeSeriesListData</function>
<filter>
    <cmdControl>
        <partial/>
    </cmdControl>
    <timeSeriesListDataSelectors>
        <timeSeriesId>3</timeSeriesId>
    </timeSeriesListDataSelectors>
</filter>
<timeSeriesListData/>
```

A reply could look like this:

```
<timeSeriesListData>
    <timeSeriesData>
        <timeSeriesId>3</timeSeriesId>
        <timePeriod>
            <startTime>2018-05-30T10:00:00.0Z</startTime>
            <endTime>2018-05-30T12:00:00.0Z</endTime>
        </timePeriod>
        <timeSeriesSlot>
            <timeSeriesSlotId>1</timeSeriesSlotId>
            <timePeriod>
                <startTime>2018-05-30T10:00:00.0Z</startTime>
            </timePeriod>
            <duration>PT30M</duration>
            <recurrenceInformation>
                <daysOfWeek>
                    <saturday/>
```



```
                    <sunday/>
                </daysOfWeek>
            </recurrenceInformation>
            <value>
                <number>21</number>
                <scale>2</scale>
            </value>
        </timeSeriesSlot>
        <timeSeriesSlot>
            <timeSeriesSlotId>2</timeSeriesSlotId>
            <timePeriod>
                <startTime>2018-05-30T11:00:00.0Z</startTime>
            </timePeriod>
            <duration>PT1H</duration>
            <recurrenceInformation>
                <daysOfWeek>
                    <saturday/>
                    <sunday/>
                </daysOfWeek>
            </recurrenceInformation>
            <value>
                <number>35</number>
                <scale>2</scale>
            </value>
        </timeSeriesSlot>
    </timeSeriesData>
</timeSeriesListData>
```

5.3.28.3 timeSeriesDescriptionListData

5.3.28.3.1 General

The timeSeriesDescriptionListData function is used to describe a timeSeries.

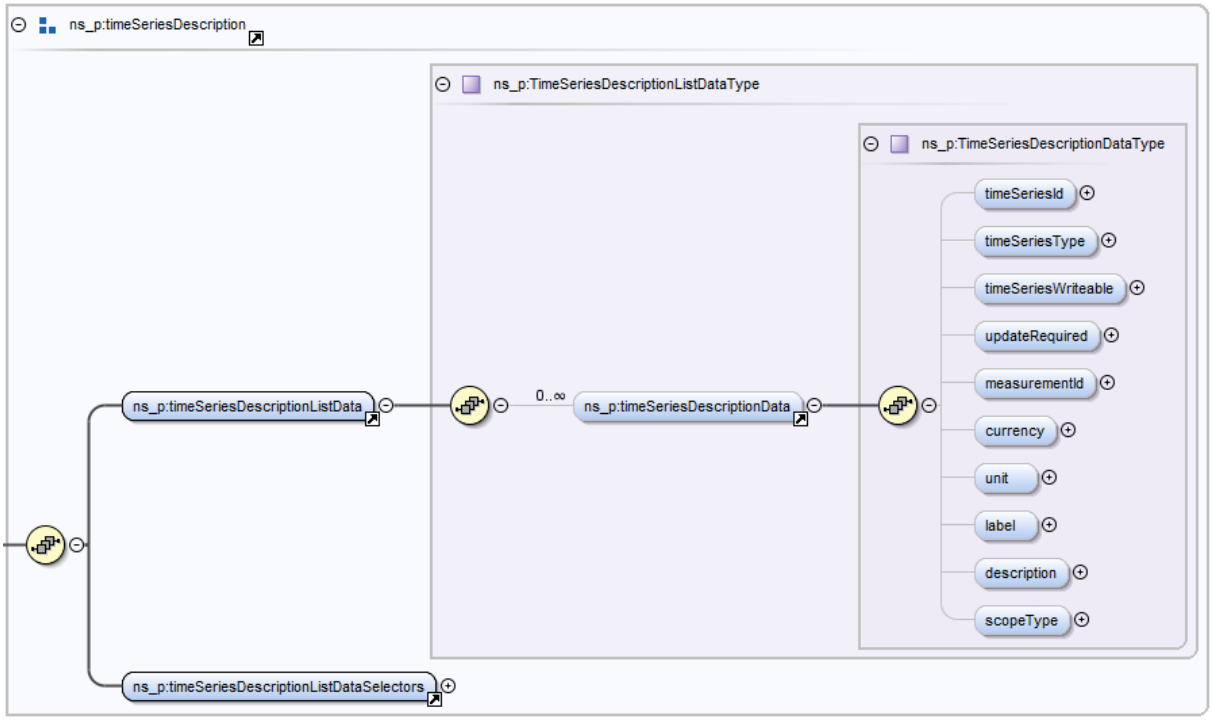


Figure 218: timeSeriesDescriptionListData function overview

10842

10843 *5.3.28.3.2 Detailed description of elements*

Element	Type	Description
timeSeriesId	*Identifier "TimeSeriesIdType" (see* Table 339*).*	Allows the identification of a timeSeries. Allows linking of the different functions to the same timeSeries. Allows linking of the timeSeries within other Features that are placed in the same entity.
timeSeriesType	*Union "*TimeSeriesTypeType*":* *- Enum (see* Table 326*):* TimeSeriesTypeEnumType - EnumExtendType (see section 3.10.1.5)	Allows to define which timeSeries type is supported.
timeSeriesWriteable	xs:boolean (W3C standard type)	Allows to define if the timeSeries is writeable.
updateRequired	xs:boolean (W3C standard type)	Request a write to timeSeriesData from the bound client.
measurementId	*Identifier "MeasurementIdType" (see* Table 339*).*	Allows to link data of other Features to a timeSeries, e.g. Measurement or ElectricalConnection data.
currency	*Common data type "CurrencyType". See section 3.10.1.19.*	Allows to describe the currency of the value, if the value is a price.
unit	*Common data type "UnitOfMeasurementType". See section 3.10.1.17.*	Allows to describe the unit of the value, if the value has a unit.
label	*Common data type "LabelType". See section 3.10.1.2.*	Allows to define user friendly name for the timeSeries.
description	*Common data type "DescriptionType". See section 3.10.1.3.*	Allows to define a user friendly description for the timeSeries.
scopeType	*Common data type "ScopeTypeType". See section 3.10.1.21.*	Specifies a more detailed meaning of the time series.

10844 *Table 325: timeSeriesDescriptionListData function detailed description of elements*

10845 Enumeration **TimeSeriesTypeEnumType**:

Value	Description
plan	"plan" is used to model how a certain value changes over time..
singleDemand	"singleDemand" is used to model a single demand with a simple timeSeries (e.g. this can be used to model a simple energy demand).
constraints	"constaints" is used to model certain value constraints over time with a min- and/or maxValue curve.

10846 *Table 326: Enumeration TimeSeriesTypeEnumType*

10847

10848 *5.3.28.3.3 Available selectors*

10849 - timeSeriesId

- timeSeriesType
- measurementId
- scopeType

5.3.28.3.4 Examples

5.3.28.3.4.1 Read-reply

To read the time series description entry with the *timeSeriesId* "3", the following read command can be send:

```
<function>timeSeriesDescriptionListData</function>
<filter>
    <cmdControl>
        <partial/>
    </cmdControl>
    <timeSeriesDescriptionListDataSelectors>
        <timeSeriesId>3</timeSeriesId>
    </timeSeriesDescriptionListDataSelectors>
</filter>
```

A reply could look like this:

```
<timeSeriesDescriptionListData>
    <timeSeriesDescriptionData>
        <timeSeriesId>3</timeSeriesId>
        <timeSeriesType>plan</timeSeriesType>
        <timeSeriesWriteable>true</timeSeriesWriteable>
        <updateRequired>false</updateRequired>
        <measurementId>1</measurementId>
        <currency>EUR</currency>
        <unit>Wh</unit>
    </timeSeriesDescriptionData>
</timeSeriesDescriptionListData>
```

5.3.28.4 timeSeriesConstraintsListData

5.3.28.4.1 General

Allows to define constraints for the timeSeries, e.g. what is the maximum amount of slots allowed or similar. This function can be especially valuable if timeSeriesData is writeable but the server has certain constraints. The client can than match its own constraints with the server constraints, to form a conforming write.

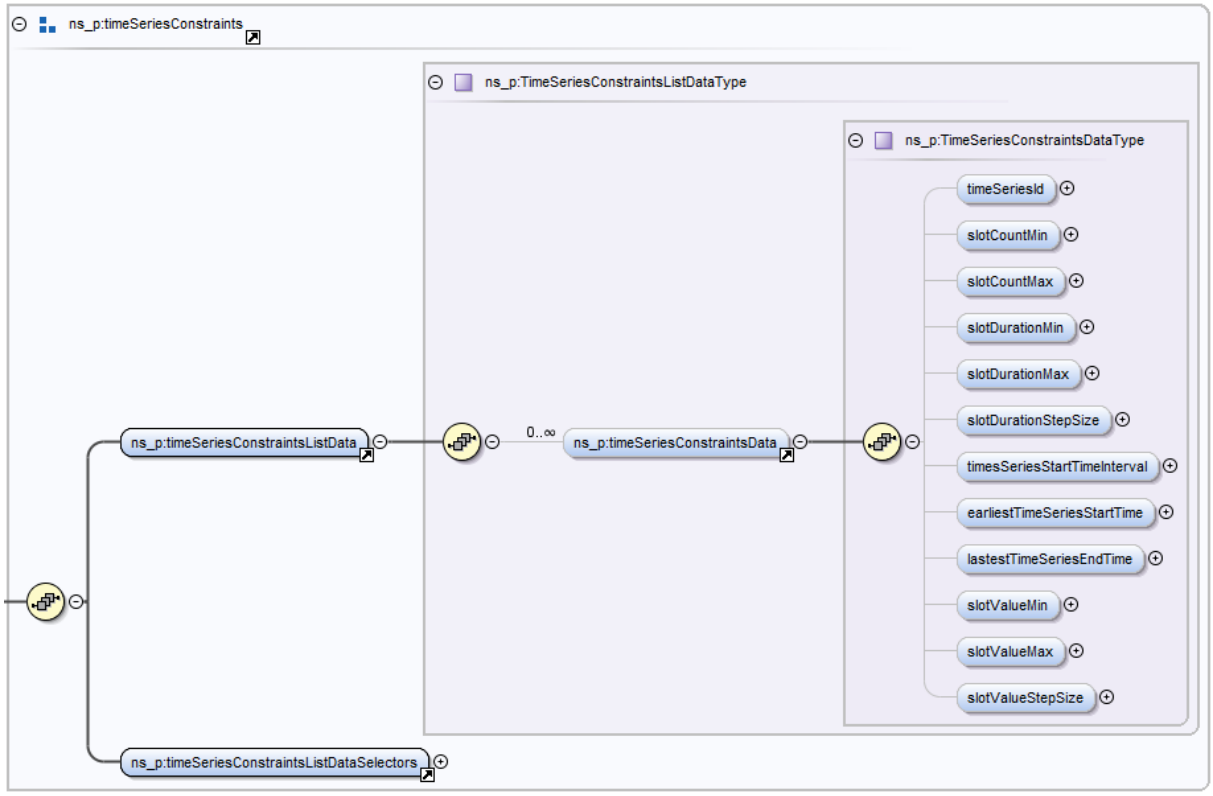


Figure 219: timeSeriesConstraintsListData function overview

5.3.28.4.2 Detailed description of elements

Element	Type	Description
timeSeriesId	*Identifier "TimeSeriesIdType" (see* Table 339*).*	Allows the identification of a timeSeries. Allows linking of the different functions to the same timeSeries. Allows linking of the timeSeries within other Features that are placed in the same entity.
slotCountMin	*Simple type "TimeSeriesSlotCountType" (restriction of "TimeSeriesSlotIdType", see* Table 324*).*	The minimum amount of slots for a specific time series is stated in this element. Recurring slots only count as one.
slotCountMax	*Simple type "TimeSeriesSlotCountType" (restriction of "TimeSeriesSlotIdType", see* Table 324*).*	The maximum amount of slots for a specific time series is stated in this element. Recurring slots only count as one.
slotDurationMin	xs:duration (W3C standard type)	If the slots of this time series have a minimum duration, it is denoted here.
slotDurationMax	xs:duration (W3C standard type)	If the slots of this time series have a maximum duration, it is denoted here.

slotDurationStepSize	xs:duration (W3C standard type)	Slots can have a specific duration step size. This element holds the corresponding value.
earliestTimeSeriesStartTime	*Common data type "AbsoluteOrRelativeTimeType". See section 3.10.2.3.*	This is the earliest start time the corresponding time series is allowed to start.
latestTimeSeriesEndTime	*Common data type "AbsoluteOrRelativeTimeType". See section 3.10.2.3.*	This is the latest end time the corresponding times series is allowed to end.
slotValueMin	*Common data type "ScaledNumberType". See section 3.10.1.8.*	The values within the slots of the corresponding time series are not allowed to be lower than this value.
slotValueMax	*Common data type "ScaledNumberType". See section 3.10.1.8.*	The values within the slots of the corresponding time series are not allowed to be higher than this value.
slotValueStepSize	*Common data type "ScaledNumberType". See section 3.10.1.8.*	The minimum step size between two different values within the slots of the corresponding time series.

Table 327: timeSeriesConstraintsListData function detailed description of elements

5.3.28.4.3 Available selectors

- timeSeriesId

5.3.28.4.4 Examples

5.3.28.4.4.1 Read-reply

To read timeSeries*Id* "3", the following read command can be send:

```
<function>timeSeriesConstraintsListData</function>
<filter>
    <cmdControl>
        <partial/>
    </cmdControl>
    <timeSeriesConstraintsListDataSelectors>
        <timeSeriesId>3</timeSeriesId>
    </timeSeriesConstraintsListDataSelectors>
</filter>
```

A reply could look like this:

```
<timeSeriesConstraintsListData>
    <timeSeriesConstraintsData>
        <timeSeriesId>3</timeSeriesId>
        <slotCountMin>1</slotCountMin>
        <slotCountMax>50</slotCountMax>
        <slotDurationMin>PT1M</slotDurationMin>
        <slotDurationMax>PT24H</slotDurationMax>
        <slotDurationStepSize>PT1S</slotDurationStepSize>
        <earliestTimeSeriesStartTime>2018-10-
15T10:00:00.0Z</earliestTimeSeriesStartTime>
```



```
        <latestTimeSeriesEndTime>2018-10-
15T18:00:00.0Z</latestTimeSeriesEndTime>
        <slotValueMin>
            <number>0</number>
        </slotValueMin>
        <slotValueMax>
            <number>99999</number>
        </slotValueMax>
        <slotValueStepSize>
            <number>1</number>
            <scale>-1</scale>
        </slotValueStepSize>
    </timeSeriesConstraintsData>
</timeSeriesConstraintsListData>
```

5.3.29 TimeTable

5.3.29.1 Introduction

All time dependent functionality like setpoint-configuration, etc. makes use of this elementary class. It provides a list of time tables, each containing some number of time slots. Each slot has a start and end time. The times can either be absolute (*dateTime*) or recurring (*daysOfWeek* plus *time* and, if required, the recurring interval step elements). The count of slots and the start and end times can be configurable or static (device specific). Other classes that make use of time tables only reference to the *timeTableId*, not to specific time slots.

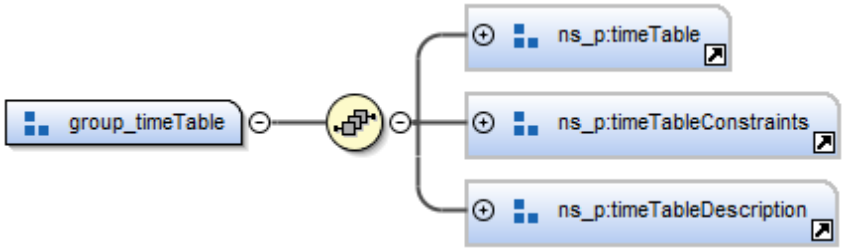


Figure 220: TimeTable function-group overview

5.3.29.2 timeTableListData

5.3.29.2.1 General

The list of time tables and time slots together with the start and end times are modelled in this function.

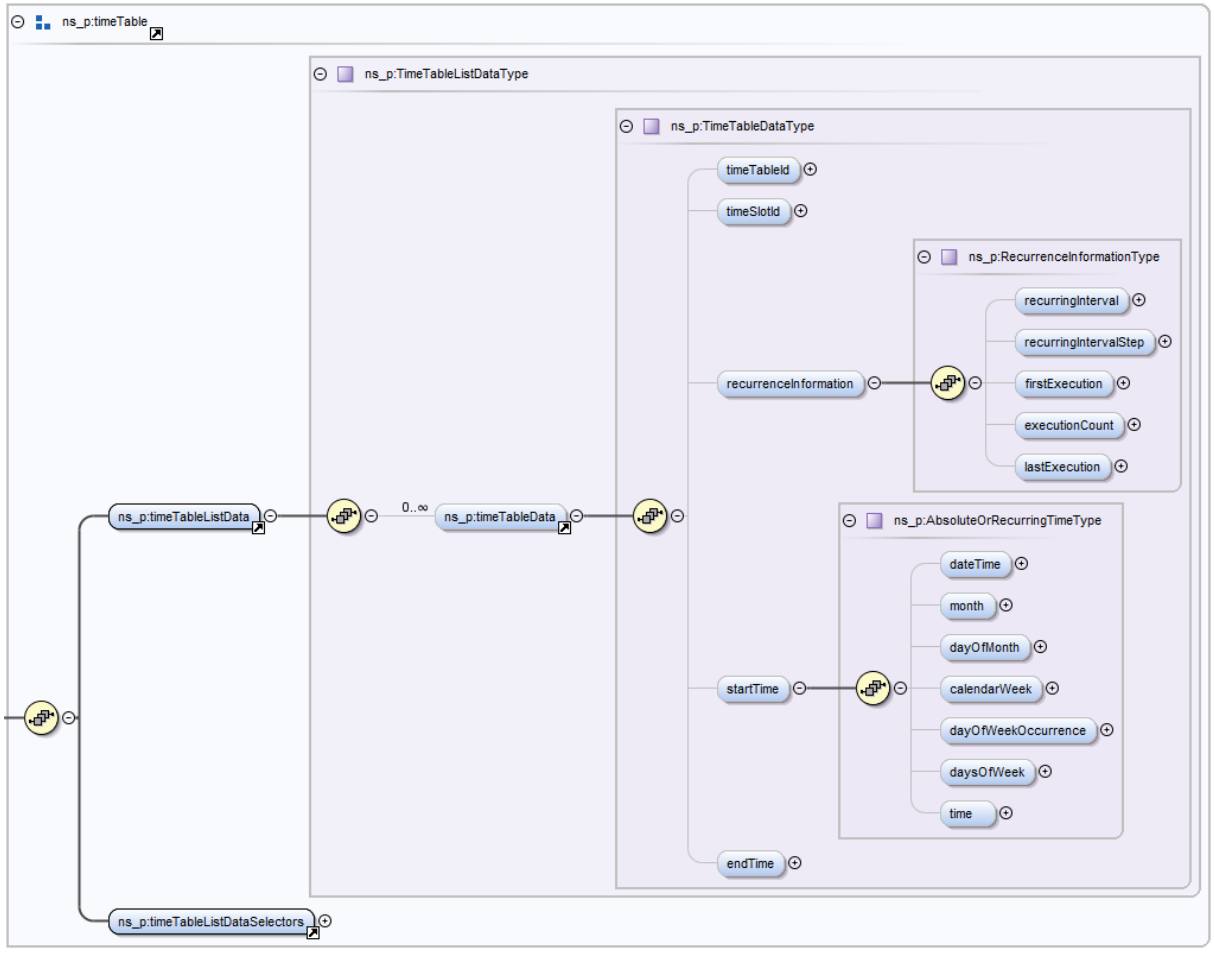


Figure 221: timeTableListData function overview

5.3.29.2.2 Detailed description of elements

Element	Type	Description
timeTableId	*Identifier "TimeTableIdType" (see* Table 339*).*	The *timeTableId* is used for indentifying specific table information in the different functions and for relations in other classes like *Setpoint*.
timeSlotId	*Identifier "TimeSlotIdType" (see* Table 339*).*	Each slot of a time table has its own slot ID.
recurrenceInformation	*Common data type "RecurrenceInformationType". See section 3.10.2.14.*	Information about the recurrence of a slot.
startTime	*Common data type "AbsoluteOrRecurringTimeType". See section 3.10.2.13.*	Start time of one slot.
endTime	*Common data type "AbsoluteOrRecurringTimeType". See section 3.10.2.13.*	End time of one slot.

Table 328: timeTableListData function detailed description of elements

5.3.29.2.3 Available selectors

- timeTableId
- timeSlotId

5.3.29.2.4 Examples

5.3.29.2.4.1 Read-reply

If one wants to request a specific time table ("2"), he could sent the following read command:

```
<function>timeTableListData</function>
<filter>
    <cmdControl>
        <partial/>
    </cmdControl>
    <timeTableListDataSelectors>
        <timeTableId>2</timeTableId>
    </timeTableListDataSelectors>
</filter>
<timeTableListData/>
```

A response could be:

```
<function>timeTableListData</function>
<filter>
    <cmdControl>
        <partial/>
    </cmdControl>
</filter>
<timeTableListData>
    <timeTableData>
        <timeTableId>2</timeTableId>
        <timeSlotId>0</timeSlotId>
        <recurrenceInformation>
            <recurringInterval>weekly</recurringInterval>
        </recurrenceInformation>
        <startTime>
            <daysOfWeek>
                <monday/>
            </daysOfWeek>
            <time>12:00:00Z</time>
        </startTime>
        <endTime>
            <daysOfWeek>
                <monday/>
            </daysOfWeek>
            <time>14:00:00Z</time>
        </endTime>
    </timeTableData>
    <timeTableData>
        <timeTableId>2</timeTableId>
        <timeSlotId>1</timeSlotId>
        <recurrenceInformation>
            <recurringInterval>weekly</recurringInterval>
        </recurrenceInformation>
        <startTime>
            <daysOfWeek>
                <tuesday/>
            </daysOfWeek>
            <time>12:00:00Z</time>
```



```
            </startTime>
            <endTime>
                <daysOfWeek>
                    <tuesday/>
                </daysOfWeek>
                <time>14:00:00Z</time>
            </endTime>
        </timeTableData>
        <timeTableData>
            <timeTableId>2</timeTableId>
            <timeSlotId>2</timeSlotId>
            <recurrenceInformation>
                <recurringInterval>weekly</recurringInterval>
            </recurrenceInformation>
            <startTime>
                <daysOfWeek>
                    <friday/>
                </daysOfWeek>
                <time>11:00:00Z</time>
            </startTime>
            <endTime>
                <daysOfWeek>
                    <friday/>
                </daysOfWeek>
                <time>17:00:00Z</time>
            </endTime>
        </timeTableData>
    </timeTableListData>
```

5.3.29.2.4.2 Write (partial)

To write new times (if allowed) into slots, one could send a message with classifier *write* like this:

```
    <function>timeTableListData</function>
    <filter>
        <cmdControl>
            <partial/>
        </cmdControl>
    </filter>
    <timeTableListData>
        <timeTableData>
            <timeTableId>1</timeTableId>
            <timeSlotId>0</timeSlotId>
            <startTime>
                <daysOfWeek>
                    <monday/>
                    <tuesday/>
                    <wednesday/>
                    <thursday/>
                    <friday/>
                    <saturday/>
                    <sunday/>
                </daysOfWeek>
                <time>10:00:00Z</time>
            </startTime>
            <endTime>
                <daysOfWeek>
                    <monday/>
                    <tuesday/>
                    <wednesday/>
                    <thursday/>
```



```
                    <friday/>
                    <saturday/>
                    <sunday/>
                </daysOfWeek>
                <time>16:00:00Z</time>
            </endTime>
        </timeTableData>
</timeTableListData>
```

5.3.29.3 timeTableConstraintsListData

5.3.29.3.1 General

If a time table has some constraints, they are denoted with the help of this function (e.g. the minimum and maximum count of slots (may differ between the time tables)).

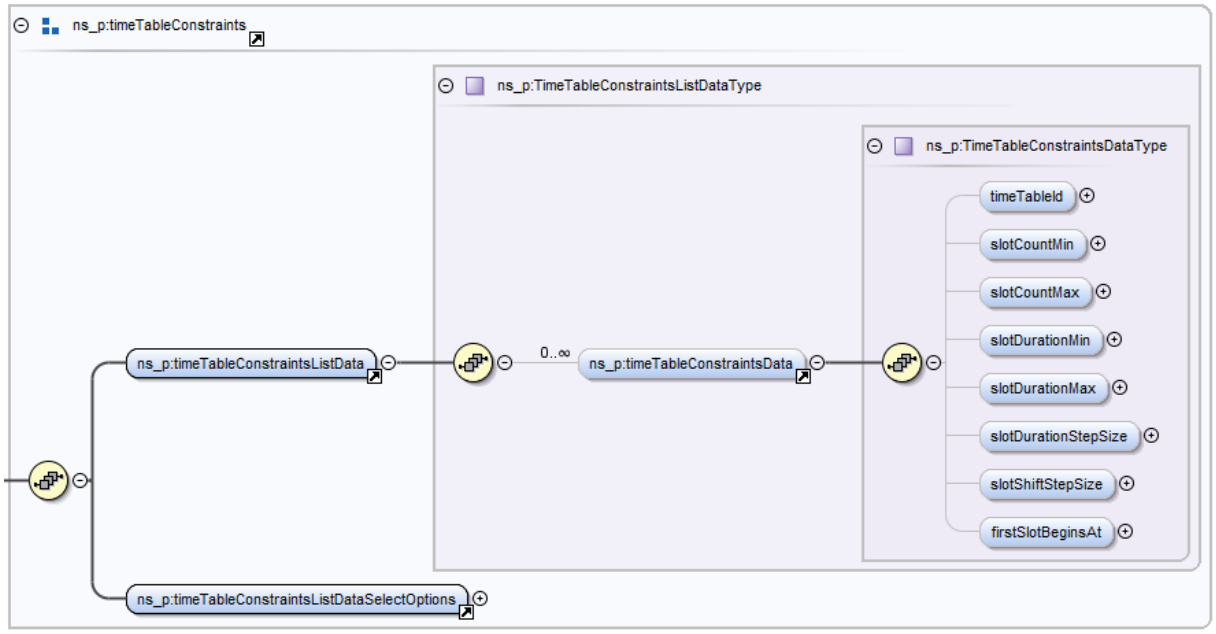


Figure 222: timeTableConstraintsListData function overview

5.3.29.3.2 Detailed description of elements

Element	Type	Description
timeTableId	*Identifier "TimeTableIdType" (see* Table 339*).*	The *timeTableId* is used for indentifying specific table information in the different functions and for relations in other classes like *Setpoint*.
slotCountMin	*Simple type "TimeSlotCountType" (restriction of "TimeSlotIdType", see* Table 328*).*	The minimum amount of slots for a specific time table is stated in this element. Recurring slots only count as one.
slotCountMax	*Simple type "TimeSlotCountType" (restriction of "TimeSlotIdType", see* Table 328*).*	The maximum amount of slots for a specific time table is stated in this element. Recurring slots only count as one.

slotDurationMin	xs:duration (W3C standard type)	If the slots of this time table have a minimum duration, it is denoted here.
slotDurationMax	xs:duration (W3C standard type)	If the slots of this time table have a maximum duration, it is denoted here.
slotDurationStepSize	xs:duration (W3C standard type)	Slots can have a specific duration step size. This element holds the corresponding value.
slotShiftStepSize	xs:duration (W3C standard type)	Slots can have a specific shift step size. This element holds the corresponding value.
firstSlotBeginsAt	xs:time (W3C standard type)	Especially time tables with fixed slot durations probably need this information to specify, where in time the slots are located (i.e. fixed 15 minute slots can start either at 00:00:00 (default) or e.g. at 00:10:00). This value only defines, when the time pattern (defined by the element *slotDurationStepSize*) begins on a day.

Table 329: timeTableConstraintsListData function detailed description of elements

5.3.29.3.3 Available selectors

- timeTableId

5.3.29.3.4 Examples

5.3.29.3.4.1 Notify

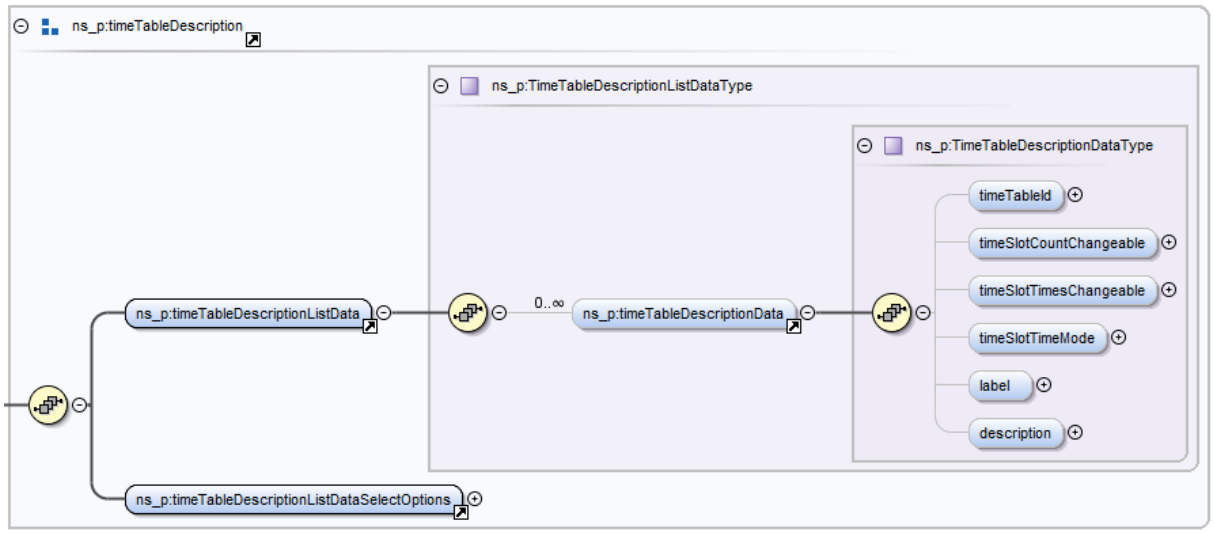
If some constraints data has changed, a device could send an update of its information with a *notify* classifier to all bound devices on the corresponding feature (the "filter/partial" part to announce the "restricted function exchange is simplified by "..." just to improve the readability"):

```
...
<timeTableConstraintsListData>
    <timeTableConstraintsData>
        <timeTableId>3</timeTableId>
        <slotCountMin>1</slotCountMin>
        <slotCountMax>10</slotCountMax>
        <slotDurationMin>PT10M</slotDurationMin>
        <slotDurationMax>PT20M</slotDurationMax>
        <slotDurationStepSize>PT5M</slotDurationStepSize>
        <slotShiftStepSize>PT5M</slotShiftStepSize>
        <firstSlotBeginsAt>00:00:00</firstSlotBeginsAt>
    </timeTableConstraintsData>
</timeTableConstraintsListData>
```

5.3.29.4 timeTableDescriptionListData

5.3.29.4.1 General

The more static information about the time tables are described in the *timeTableDescriptionData* function.



11113

11114 *Figure 223: timeTableDescriptionListData function overview*

11115

11116 *5.3.29.4.2 Detailed description of elements*

Element	Type	Description
timeTableId	*Identifier "TimeTableIdType" (see* Table 339*).*	The *timeTableId* is used for indentifying specific table information in the different functions and for relations in other classes like *Setpoint*.
timeSlotCountChangeable	xs:boolean (W3C standard type)	States whether the amount of slots is changeable by a client for a specific time table or not.
timeSlotTimesChangeable	xs:boolean (W3C standard type)	States whether the times of the slots of a specific time table are changeable by a client or not.
timeSlotTimeMode	*Union "*TimeSlotTimeModeType*":* *- Enum (see* Table 331*):* TimeSlotTimeModeEnumType - EnumExtendType (see section 3.10.1.5)	If either *absolute* or *recurring* time slots or *both* modes are supported is stated in this element.
label	*Common data type "LabelType". See section 3.10.1.2.*	A short label of the time table.
description	*Common data type "DescriptionType". See section 3.10.1.3.*	Descriptive information on the time table.

11117 *Table 330: timeTableDescriptionListData function detailed description of elements*

11118 Enumeration **TimeSlotTimeModeEnumType**:

Value	Description
absolute	Only absolute times are allowed for the time table.
recurring	Only recurring times are allowed for the time table.
both	Absolute and recurring times are allowed for the time table.

11119 *Table 331: Enumeration TimeSlotTimeModeEnumType*

11120

11121 *5.3.29.4.3 Available selectors*

11122 - timeTableId

11123

11124 *5.3.29.4.4 Examples*

11125 *5.3.29.4.4.1 Read-reply*

11126 If one is interested in the time table description of all available time tables, he would send the
11127 standard read command on the *timeTableDescriptionListData* resource. A response could look like
11128 this:

```
11129 <timeTableDescriptionListData>
11130     <timeTableDescriptionData>
11131         <timeTableId>1</timeTableId>
11132         <timeSlotCountChangeable>false</timeSlotCountChangeable>
11133         <timeSlotTimesChangeable>true</timeSlotTimesChangeable>
11134         <timeSlotTimeMode>absolute</timeSlotTimeMode>
11135     </timeTableDescriptionData>
11136     <timeTableDescriptionData>
11137         <timeTableId>2</timeTableId>
11138         <timeSlotCountChangeable>false</timeSlotCountChangeable>
11139         <timeSlotTimesChangeable>true</timeSlotTimesChangeable>
11140         <timeSlotTimeMode>both</timeSlotTimeMode>
11141     </timeTableDescriptionData>
11142 </timeTableDescriptionListData>
```

11143

11144 **5.3.30 UseCaseInformation**

11145 ***5.3.30.1 Introduction***

11146 To model which Use Cases are supported by a node the UseCaseInformation Class can be used.

11147 

11148 *Figure 224: UseCaseInformation function-group overview*

11149

11150 ***5.3.30.2 useCaseInformationListData***

11151 *5.3.30.2.1 General*

11152 This function provides information about the supported Use Cases together with a version
11153 information and further relevant details.

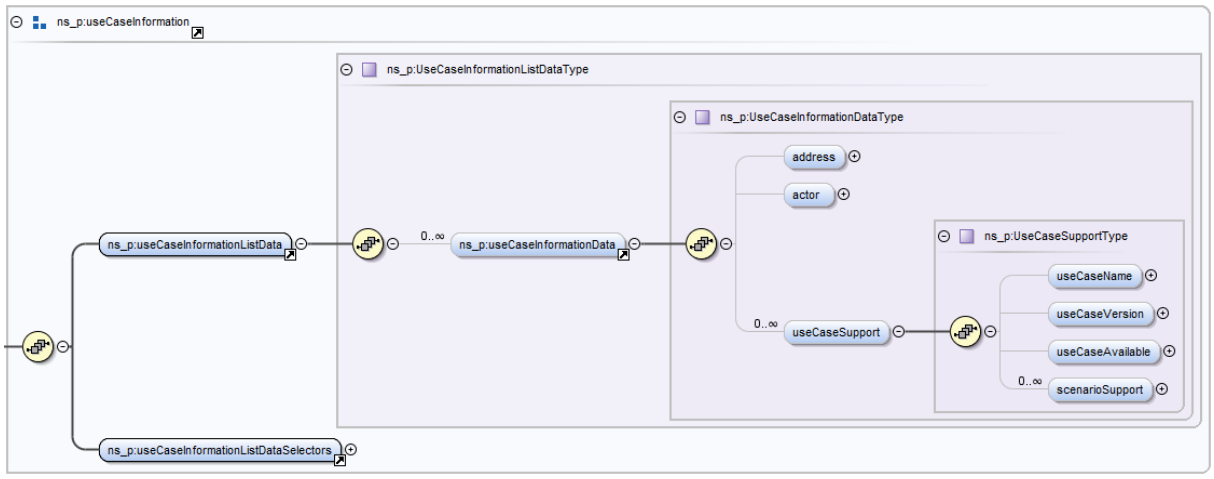


Figure 225: useCaseInformationListData function overview

5.3.30.2.2 Detailed description of elements

Element	Type	Description
address	*Common data type "FeatureAddressType". See section 3.10.3.6.*	The address where the supported Use Case is located. Note: not all sub-elements always need to be set here!
actor	*Union "UseCaseActorType":* *- Simple type "UseCaseActorEnumType" (restriction of xs:string)* - EnumExtendType (see section 3.10.1.5)	The actor of the supported Use Case located on the above stated address.
useCaseSupport (list)		List of supported Use Cases.
useCaseSupport. useCaseName	*Union "UseCaseNameType":* *- Simple type "UseCaseNameEnumType" (restriction of xs:string)* - EnumExtendType (see section 3.10.1.5)	The name of the supported Use Case.
useCaseSupport. useCaseVersion	*Common data type "SpecificationVersionType". See section 3.10.1.4.*	Version of the supported Use Case.
useCaseSupport. useCaseAvailable	xs:boolean (W3C standard type)	Stated whether the Use Case is available at the moment.
useCaseSupport. scenarioSupport (list)	*Simple type "UseCaseScenarioSupportType" (restriction of "xs:unsignedInt")*	List of supported scenarios of the Use Case.

Table 332: useCaseInformationListData function detailed description of elements

5.3.30.2.3 Available selectors

- address (with sub-elements, see above)
- actor
- useCaseSupport (with sub-elements, see above)

5.3.30.2.4 Examples

5.3.30.2.4.1 Read-reply

A *useCaseInformationListData* instance sent as reply to a standard read command:

```
<useCaseInformationListData>
    <useCaseInformationData>
        <address>
            <device>d:_i:46925_TestDevice_1</device>
            <entity>1</entity>
            <entity>1</entity>
        </address>
        <actor>Compressor</actor>
        <useCaseSupport>

<useCaseName>visualizationOfPowerConsumptionOfHeatPumpCompressor</useCaseNa
me>
            <useCaseVersion>1.0.0</useCaseVersion>
            <useCaseAvailable>true</useCaseAvailable>
            <scenarioSupport>1</scenarioSupport>
        </useCaseSupport>
        <useCaseSupport>

<useCaseName>optimizationForOptionalHeatPumpPowerConsumption</useCaseName>
            <useCaseVersion>1.0.0</useCaseVersion>
            <useCaseAvailable>false</useCaseAvailable>
            <scenarioSupport>1</scenarioSupport>
        </useCaseSupport>
    </useCaseInformationData>
    <useCaseInformationData>
        <address>
            <device>d:_i:46925_TestDevice_1</device>
            <entity>2</entity>
            <feature>1</feature>
        </address>
        <actor>GridConnectionPointOfPremises</actor>
        <useCaseSupport>

<useCaseName>remoteMonitoringOfElectricityGridConnectionPoint</useCaseName>
            <useCaseVersion>1.0.0</useCaseVersion>
            <useCaseAvailable>true</useCaseAvailable>
            <scenarioSupport>1</scenarioSupport>
            <scenarioSupport>2</scenarioSupport>
            <scenarioSupport>3</scenarioSupport>
            <scenarioSupport>4</scenarioSupport>
        </useCaseSupport>
    </useCaseInformationData>
</useCaseInformationListData>
```

5.3.31 Version

5.3.31.1 Introduction

This class can be used to model the version information of the SPINE specification. The version string is devided into *major*, *minor* and *revision*, separated by a dot (e.g. "1.0.0"), see section 5.3.31.2.1. The version information is also used within the “header” of “datagram” (see document

11217 [ProtocolSpecification]). This section, however, provides the definition and proper usage of “Version”
11218 as SPINE class.

11219 

11220 *Figure 226: Version function-group overview*

11221

11222 ### *5.3.31.2 specificationVersionListData*

11223 #### *5.3.31.2.1 General*

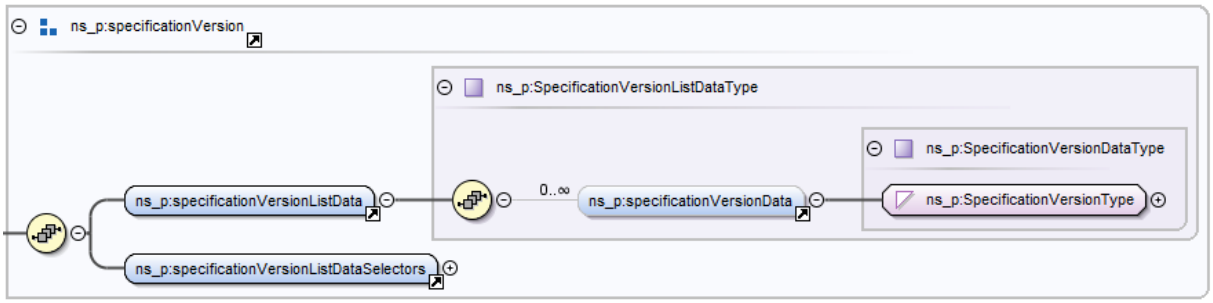
11224 The knowledge of the supported SPINE data model versions of a communication partner is essential
11225 for the proper interworking of devices that implemented different versions. Considering the
11226 compatibility rules defined in [ProtocolSpecification] is only possible with this kind of knowledge.
11227 Therefore, the *specificationVersionListData* function is used.

11228 The three parts of the version have the following meaning:

11229 **major**
11230 Typically, a specification's major part of the version number is increased by the proper authority in
11231 case of “fundamental changes”. Among others, changes that break mandatory compatibility
11232 requirements belong to such kind of changes.

11233 **minor**
11234 Typically, a specification's minor part of the version number is increased by the proper authority in
11235 case of optional additions to the pedecessor of the specification version.

11236 **revision**
11237 Typically, a specification's revision part of the version number is increased by the proper authority in
11238 case of “minor changes” where major and minor part can remain unchanged.

11239 

11240 *Figure 227: specificationVersionListData function overview*

11241

11242 #### *5.3.31.2.2 Detailed description of elements*

Element	Type	Description
specificationVersionData	*Common data type "SpecificationVersionType". See section 3.10.1.4.*	One of the supported SPINE specification versions.

11243 *Table 333: specificationVersionListData function detailed description of elements*

11244

11245 *5.3.31.2.3 Available selectors*
11246 None.

11247

11248 *5.3.31.2.4 Examples*

11249 *5.3.31.2.4.1 Notify*
11250 A device sends the *specificationVersionListData* function (as (complete) notification) to another
11251 device, including all supported versions of the SPINE specification:

```
11252 <specificationVersionListData>
11253     <specificationVersionData>1.0.0</specificationVersionData>
11254 </specificationVersionListData>
```

11255

Annex A - SPINE XSDs

All necessary XSDs defined for SPINE are included in this annex. The so-called "overview" XSDs are left out as they only reference to the functions defined within the "normal" XSDs. An example for an overview XSD is given in section A.3.6 Overview example.

The EEBus Initiative e.V. also provides the XSD files for download (please consider the corresponding website [EEBus]). These XSD files should be preferred as the the printed XSDs in this annex contain additional line breaks due to page layout reasons.

A.1 Complex Classes

A.1.1 IncentiveTable

File name: EEBus_SPINE_TS_IncentiveTable.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
    Smart Premises Interoperable Neutral-Message Exchange (SPINE)
    Version 1.1.1
    2018-12-21
    Copyright (c) 2018 EEBus Initiative e.V.  All Rights Reserved.
    Source: https://www.eebus.org/en/specifications/
-->
<xs:schema xmlns:ns_p="http://docs.eebus.org/spine/xsd/v1"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://docs.eebus.org/spine/xsd/v1" version="1.1.1"
    blockDefault="#all" elementFormDefault="qualified">
    <xs:annotation>
        <xs:documentation>EEBus SPINE Specification schema. Copyright 2018 EEBus Initiative
e.V.  All rights reserved.</xs:documentation>
    </xs:annotation>
    <xs:include schemaLocation="EEBus_SPINE_TS_CommonDataTypes.xsd"/>
    <xs:include schemaLocation="EEBus_SPINE_TS_TariffInformation.xsd"/>
    <xs:complexType name="IncentiveTableType">
        <xs:sequence>
            <xs:element minOccurs="0" name="tariff">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:restriction base="ns_p:TariffDataType">
                            <xs:sequence>
                                <xs:element minOccurs="0" name="tariffId"
type="ns_p:TariffIdType"/>
                            </xs:sequence>
                        </xs:restriction>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
            <xs:element maxOccurs="unbounded" minOccurs="0" name="incentiveSlot"
type="ns_p:IncentiveTableIncentiveSlotType"> </xs:element>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="IncentiveTableIncentiveSlotType">
        <xs:sequence>
            <xs:element minOccurs="0" name="timeInterval">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:restriction base="ns_p:TimeTableDataType">
                            <xs:sequence>
                                <xs:element minOccurs="0" name="timeSlotId"
type="ns_p:TimeSlotIdType"/>
                                <xs:element minOccurs="0" name="startTime">
                                    <xs:complexType>
                                        <xs:complexContent>
                                            <xs:restriction
base="ns_p:AbsoluteOrRecurringTimeType">
                                                <xs:sequence>
```

```
                                                    <xs:element minOccurs="0" name="dateTime"
type="xs:dateTime"/>
                                                    <xs:element minOccurs="0" name="relative"
type="xs:duration"/>
                                                </xs:sequence>
                                            </xs:restriction>
                                        </xs:complexContent>
                                    </xs:complexType>
                                </xs:element>
                                <xs:element minOccurs="0" name="endTime">
                                    <xs:complexType>
                                        <xs:complexContent>
                                            <xs:restriction
base="ns_p:AbsoluteOrRecurringTimeType">
                                                <xs:sequence>
                                                    <xs:element minOccurs="0" name="dateTime"
type="xs:dateTime"/>
                                                    <xs:element minOccurs="0" name="relative"
type="xs:duration"/>
                                                </xs:sequence>
                                            </xs:restriction>
                                        </xs:complexContent>
                                    </xs:complexType>
                                </xs:element>
                            </xs:sequence>
                        </xs:restriction>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
            <xs:element maxOccurs="unbounded" minOccurs="0" name="tier"
type="ns_p:IncentiveTableTierType"> </xs:element>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="IncentiveTableTierType">
        <xs:sequence>
            <xs:element minOccurs="0" name="tier">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:restriction base="ns_p:TierDataType">
                            <xs:sequence>
                                <xs:element minOccurs="0" name="tierId"
type="ns_p:TierIdType"/>
                            </xs:sequence>
                        </xs:restriction>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
            <xs:element maxOccurs="unbounded" minOccurs="0" name="boundary">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:restriction base="ns_p:TierBoundaryDataType">
                            <xs:sequence>
                                <xs:element minOccurs="0" name="boundaryId"
type="ns_p:TierBoundaryIdType"/>
                                <xs:element minOccurs="0" name="lowerBoundaryValue"
type="ns_p:ScaledNumberType"/>
                                <xs:element minOccurs="0" name="upperBoundaryValue"
type="ns_p:ScaledNumberType"/>
                            </xs:sequence>
                        </xs:restriction>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
            <xs:element minOccurs="0" name="incentive" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:restriction base="ns_p:IncentiveDataType">
                            <xs:sequence>
                                <xs:element minOccurs="0" name="incentiveId"
type="ns_p:IncentiveIdType"/>
                                <xs:element minOccurs="0" name="value"
type="ns_p:ScaledNumberType"/>
                            </xs:sequence>
                        </xs:restriction>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
```



```
11397       </xs:complexType>
11398       <xs:complexType name="IncentiveTableDataType">
11399           <xs:sequence>
11400               <xs:element maxOccurs="unbounded" minOccurs="0" name="incentiveTable"
11401   type="ns_p:IncentiveTableType"> </xs:element>
11402           </xs:sequence>
11403       </xs:complexType>
11404       <xs:element name="incentiveTableData" type="ns_p:IncentiveTableDataType"/>
11405       <xs:complexType name="IncentiveTableElementsType">
11406           <xs:sequence>
11407               <xs:element minOccurs="0" name="tariff">
11408                   <xs:complexType>
11409                       <xs:complexContent>
11410                           <xs:restriction base="ns_p:TariffDataElementsType">
11411                               <xs:sequence>
11412                                   <xs:element minOccurs="0" name="tariffId"
11413   type="ns_p:ElementTagType"/>
11414                               </xs:sequence>
11415                           </xs:restriction>
11416                       </xs:complexContent>
11417                   </xs:complexType>
11418               </xs:element>
11419               <xs:element minOccurs="0" name="incentiveSlot"
11420   type="ns_p:IncentiveTableIncentiveSlotElementsType"> </xs:element>
11421           </xs:sequence>
11422       </xs:complexType>
11423       <xs:complexType name="IncentiveTableIncentiveSlotElementsType">
11424           <xs:sequence>
11425               <xs:element minOccurs="0" name="timeInterval">
11426                   <xs:complexType>
11427                       <xs:complexContent>
11428                           <xs:restriction base="ns_p:TimeTableDataElementsType">
11429                               <xs:sequence>
11430                                   <xs:element minOccurs="0" name="timeSlotId"
11431   type="ns_p:ElementTagType"/>
11432                                   <xs:element minOccurs="0" name="startTime">
11433                                       <xs:complexType>
11434                                           <xs:complexContent>
11435                                               <xs:restriction
11436   base="ns_p:AbsoluteOrRecurringTimeElementsType">
11437                                                   <xs:sequence>
11438                                                       <xs:element minOccurs="0" name="dateTime"
11439   type="ns_p:ElementTagType"/>
11440                                                       <xs:element minOccurs="0" name="relative"
11441   type="ns_p:ElementTagType"/>
11442                                                   </xs:sequence>
11443                                               </xs:restriction>
11444                                           </xs:complexContent>
11445                                       </xs:complexType>
11446                                   </xs:element>
11447                                   <xs:element minOccurs="0" name="endTime">
11448                                       <xs:complexType>
11449                                           <xs:complexContent>
11450                                               <xs:restriction
11451   base="ns_p:AbsoluteOrRecurringTimeElementsType">
11452                                                   <xs:sequence>
11453                                                       <xs:element minOccurs="0" name="dateTime"
11454   type="ns_p:ElementTagType"/>
11455                                                       <xs:element minOccurs="0" name="relative"
11456   type="ns_p:ElementTagType"/>
11457                                                   </xs:sequence>
11458                                               </xs:restriction>
11459                                           </xs:complexContent>
11460                                       </xs:complexType>
11461                                   </xs:element>
11462                               </xs:sequence>
11463                           </xs:restriction>
11464                       </xs:complexContent>
11465                   </xs:complexType>
11466               </xs:element>
11467               <xs:element minOccurs="0" name="tier" type="ns_p:IncentiveTableTierElementsType">
11468   </xs:element>
11469           </xs:sequence>
11470       </xs:complexType>
11471       <xs:complexType name="IncentiveTableTierElementsType">
11472           <xs:sequence>
11473               <xs:element minOccurs="0" name="tier">
11474                   <xs:complexType>
```



```
                    <xs:complexContent>
                        <xs:restriction base="ns_p:TierDataElementsType">
                            <xs:sequence>
                                <xs:element minOccurs="0" name="tierId"
type="ns_p:ElementTagType"/>
                            </xs:sequence>
                        </xs:restriction>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
            <xs:element minOccurs="0" name="boundary">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:restriction base="ns_p:TierBoundaryDataElementsType">
                            <xs:sequence>
                                <xs:element minOccurs="0" name="boundaryId"
type="ns_p:ElementTagType"/>
                                <xs:element minOccurs="0" name="lowerBoundaryValue"
type="ns_p:ScaledNumberElementsType"/>
                                <xs:element minOccurs="0" name="upperBoundaryValue"
type="ns_p:ScaledNumberElementsType"/>
                            </xs:sequence>
                        </xs:restriction>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
            <xs:element minOccurs="0" name="incentive">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:restriction base="ns_p:IncentiveDataElementsType">
                            <xs:sequence>
                                <xs:element minOccurs="0" name="incentiveId"
type="ns_p:ElementTagType"/>
                                <xs:element minOccurs="0" name="value"
type="ns_p:ElementTagType"/>
                            </xs:sequence>
                        </xs:restriction>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="IncentiveTableDataElementsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="incentiveTable"
type="ns_p:IncentiveTableElementsType"> </xs:element>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="incentiveTableDataElements" type="ns_p:IncentiveTableDataElementsType"/>
    <xs:complexType name="IncentiveTableDataSelectorsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="tariff">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:restriction base="ns_p:TariffListDataSelectorsType">
                            <xs:sequence>
                                <xs:element minOccurs="0" name="tariffId"
type="ns_p:TariffIdType"/>
                            </xs:sequence>
                        </xs:restriction>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="incentiveTableDataSelectors"
type="ns_p:IncentiveTableDataSelectorsType"/>
    <xs:complexType name="IncentiveTableDescriptionType">
        <xs:sequence>
            <xs:element minOccurs="0" name="tariffDescription">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:restriction base="ns_p:TariffDescriptionDataType">
                            <xs:sequence>
                                <xs:element minOccurs="0" name="tariffId"
type="ns_p:TariffIdType"/>
                                <xs:element minOccurs="0" name="measurementId"
type="ns_p:MeasurementIdType"/>
```



```
                                <xs:element minOccurs="0" name="tariffWriteable"
type="xs:boolean"/>
                                <xs:element minOccurs="0" name="updateRequired"
type="xs:boolean"/>
                                <xs:element minOccurs="0" name="scopeType"
type="ns_p:ScopeTypeType"/>
                                <xs:element minOccurs="0" name="label" type="ns_p:LabelType"/>
                                <xs:element minOccurs="0" name="description"
type="ns_p:DescriptionType"/>
                                <xs:element minOccurs="0" name="slotIdSupport"
type="xs:boolean"/>
                            </xs:sequence>
                        </xs:restriction>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
            <xs:element maxOccurs="unbounded" minOccurs="0" name="tier"
type="ns_p:IncentiveTableDescriptionTierType"> </xs:element>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="IncentiveTableDescriptionTierType">
        <xs:sequence>
            <xs:element minOccurs="0" name="tierDescription">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:restriction base="ns_p:TierDescriptionDataType">
                            <xs:sequence>
                                <xs:element minOccurs="0" name="tierId"
type="ns_p:TierIdType"/>
                                <xs:element minOccurs="0" name="tierType"
type="ns_p:TierTypeType"/>
                                <xs:element minOccurs="0" name="label" type="ns_p:LabelType"/>
                                <xs:element minOccurs="0" name="description"
type="ns_p:DescriptionType"/>
                            </xs:sequence>
                        </xs:restriction>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
            <xs:element maxOccurs="unbounded" minOccurs="0" name="boundaryDescription">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:restriction base="ns_p:TierBoundaryDescriptionDataType">
                            <xs:sequence>
                                <xs:element minOccurs="0" name="boundaryId"
type="ns_p:TierBoundaryIdType"/>
                                <xs:element minOccurs="0" name="boundaryType"
type="ns_p:TierBoundaryTypeType"/>
                                <xs:element minOccurs="0" name="switchToTierIdWhenLower"
type="ns_p:TierIdType"/>
                                <xs:element minOccurs="0" name="switchToTierIdWhenHigher"
type="ns_p:TierIdType"/>
                                <xs:element minOccurs="0" name="boundaryUnit"
type="ns_p:UnitOfMeasurementType"/>
                            </xs:sequence>
                        </xs:restriction>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
            <xs:element minOccurs="0" name="incentiveDescription" maxOccurs="unbounded">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:restriction base="ns_p:IncentiveDescriptionDataType">
                            <xs:sequence>
                                <xs:element minOccurs="0" name="incentiveId"
type="ns_p:IncentiveIdType"/>
                                <xs:element minOccurs="0" name="incentiveType"
type="ns_p:IncentiveTypeType"/>
                                <xs:element minOccurs="0" name="incentivePriority"
type="ns_p:IncentivePriorityType"/>
                                <xs:element minOccurs="0" name="currency"
type="ns_p:CurrencyType"/>
                                <xs:element minOccurs="0" name="unit"
type="ns_p:UnitOfMeasurementType"/>
                                <xs:element minOccurs="0" name="label" type="ns_p:LabelType"/>
                                <xs:element minOccurs="0" name="description"
type="ns_p:DescriptionType"/>
                            </xs:sequence>
```



```
                        </xs:restriction>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="IncentiveTableDescriptionDataType">
        <xs:sequence>
            <xs:element maxOccurs="unbounded" minOccurs="0" name="incentiveTableDescription"
type="ns_p:IncentiveTableDescriptionType"> </xs:element>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="incentiveTableDescriptionData"
type="ns_p:IncentiveTableDescriptionDataType"/>
    <xs:complexType name="IncentiveTableDescriptionElementsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="tariffDescription">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:restriction base="ns_p:TariffDescriptionDataElementsType">
                            <xs:sequence>
                                <xs:element minOccurs="0" name="tariffId"
type="ns_p:ElementTagType"/>
                                <xs:element minOccurs="0" name="measurementId"
type="ns_p:ElementTagType"/>
                                <xs:element minOccurs="0" name="tariffWriteable"
type="ns_p:ElementTagType"/>
                                <xs:element minOccurs="0" name="updateRequired"
type="ns_p:ElementTagType"/>
                                <xs:element minOccurs="0" name="scopeType"
type="ns_p:ElementTagType"/>
                                <xs:element minOccurs="0" name="label"
type="ns_p:ElementTagType"/>
                                <xs:element minOccurs="0" name="description"
type="ns_p:ElementTagType"/>
                                <xs:element minOccurs="0" name="slotIdSupport"
type="ns_p:ElementTagType"/>
                            </xs:sequence>
                        </xs:restriction>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
            <xs:element minOccurs="0" name="tier"
type="ns_p:IncentiveTableDescriptionTierElementsType"> </xs:element>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="IncentiveTableDescriptionTierElementsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="tierDescription">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:restriction base="ns_p:TierDescriptionDataElementsType">
                            <xs:sequence>
                                <xs:element minOccurs="0" name="tierId"
type="ns_p:ElementTagType"/>
                                <xs:element minOccurs="0" name="tierType"
type="ns_p:ElementTagType"/>
                                <xs:element minOccurs="0" name="label"
type="ns_p:ElementTagType"/>
                                <xs:element minOccurs="0" name="description"
type="ns_p:ElementTagType"/>
                            </xs:sequence>
                        </xs:restriction>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
            <xs:element minOccurs="0" name="boundaryDescription">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:restriction base="ns_p:TierBoundaryDescriptionDataElementsType">
                            <xs:sequence>
                                <xs:element minOccurs="0" name="boundaryId"
type="ns_p:ElementTagType"/>
                                <xs:element minOccurs="0" name="boundaryType"
type="ns_p:ElementTagType"/>
                                <xs:element minOccurs="0" name="switchToTierIdWhenLower"
type="ns_p:ElementTagType"/>
```



```
                                <xs:element minOccurs="0" name="switchToTierIdWhenHigher"
type="ns_p:ElementTagType"/>
                                <xs:element minOccurs="0" name="boundaryUnit"
type="ns_p:ElementTagType"/>
                            </xs:sequence>
                        </xs:restriction>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
            <xs:element minOccurs="0" name="incentiveDescription">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:restriction base="ns_p:IncentiveDescriptionDataElementsType">
                            <xs:sequence>
                                <xs:element minOccurs="0" name="incentiveId"
type="ns_p:ElementTagType"/>
                                <xs:element minOccurs="0" name="incentiveType"
type="ns_p:ElementTagType"/>
                                <xs:element minOccurs="0" name="incentivePriority"
type="ns_p:ElementTagType"/>
                                <xs:element minOccurs="0" name="currency"
type="ns_p:ElementTagType"/>
                                <xs:element minOccurs="0" name="unit"
type="ns_p:ElementTagType"/>
                                <xs:element minOccurs="0" name="label"
type="ns_p:ElementTagType"/>
                                <xs:element minOccurs="0" name="description"
type="ns_p:ElementTagType"/>
                            </xs:sequence>
                        </xs:restriction>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="IncentiveTableDescriptionDataElementsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="incentiveTableDescription"
type="ns_p:IncentiveTableDescriptionElementsType"> </xs:element>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="incentiveTableDescriptionDataElements"
type="ns_p:IncentiveTableDescriptionDataElementsType"/>
    <xs:complexType name="IncentiveTableDescriptionDataSelectorsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="tariffDescription">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:restriction base="ns_p:TariffDescriptionListDataSelectorsType">
                            <xs:sequence>
                                <xs:element minOccurs="0" name="tariffId"
type="ns_p:TariffIdType"/>
                                <xs:element minOccurs="0" name="scopeType"
type="ns_p:ScopeTypeType"/>
                            </xs:sequence>
                        </xs:restriction>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="incentiveTableDescriptionDataSelectors"
type="ns_p:IncentiveTableDescriptionDataSelectorsType"/>
    <xs:complexType name="IncentiveTableConstraintsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="tariff">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:restriction base="ns_p:TariffDataType">
                            <xs:sequence>
                                <xs:element minOccurs="0" name="tariffId"
type="ns_p:TariffIdType"/>
                            </xs:sequence>
                        </xs:restriction>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
            <xs:element name="tariffConstraints" minOccurs="0">
```



```
                <xs:complexType>
                    <xs:complexContent>
                        <xs:restriction base="ns_p:TariffOverallConstraintsDataType">
                            <xs:sequence>
                                <xs:element minOccurs="0" name="maxTiersPerTariff"
type="ns_p:TierCountType"/>
                                <xs:element minOccurs="0" name="maxBoundariesPerTier"
type="ns_p:TierBoundaryCountType"/>
                                <xs:element minOccurs="0" name="maxIncentivesPerTier"
type="ns_p:IncentiveCountType"/>
                            </xs:sequence>
                        </xs:restriction>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
            <xs:element name="incentiveSlotConstraints" minOccurs="0">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:restriction base="ns_p:TimeTableConstraintsDataType">
                            <xs:sequence>
                                <xs:element minOccurs="0" name="slotCountMax"
type="ns_p:TimeSlotCountType"/>
                            </xs:sequence>
                        </xs:restriction>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="IncentiveTableConstraintsDataType">
        <xs:sequence>
            <xs:element maxOccurs="unbounded" minOccurs="0" name="incentiveTableConstraints"
type="ns_p:IncentiveTableConstraintsType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="incentiveTableConstraintsData"
type="ns_p:IncentiveTableConstraintsDataType"/>
    <xs:complexType name="IncentiveTableConstraintsElementsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="tariff">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:restriction base="ns_p:TariffDataElementsType">
                            <xs:sequence>
                                <xs:element minOccurs="0" name="tariffId"
type="ns_p:ElementTagType"/>
                            </xs:sequence>
                        </xs:restriction>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
            <xs:element name="tariffConstraints" minOccurs="0">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:restriction base="ns_p:TariffOverallConstraintsDataElementsType">
                            <xs:sequence>
                                <xs:element minOccurs="0" name="maxTiersPerTariff"
type="ns_p:ElementTagType"/>
                                <xs:element minOccurs="0" name="maxBoundariesPerTier"
type="ns_p:ElementTagType"/>
                                <xs:element minOccurs="0" name="maxIncentivesPerTier"
type="ns_p:ElementTagType"/>
                            </xs:sequence>
                        </xs:restriction>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
            <xs:element name="incentiveSlotConstraints" minOccurs="0">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:restriction base="ns_p:TimeTableConstraintsDataElementsType">
                            <xs:sequence>
                                <xs:element minOccurs="0" name="slotCountMax"
type="ns_p:ElementTagType"/>
                            </xs:sequence>
                        </xs:restriction>
                    </xs:complexContent>
                </xs:complexType>
```



```
            </xs:element>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="IncentiveTableConstraintsDataElementsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="incentiveTableConstraints"
type="ns_p:IncentiveTableConstraintsElementsType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="incentiveTableConstraintsDataElements"
type="ns_p:IncentiveTableConstraintsDataElementsType"/>
    <xs:complexType name="IncentiveTableConstraintsDataSelectorsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="tariff">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:restriction base="ns_p:TariffListDataSelectorsType">
                            <xs:sequence>
                                <xs:element minOccurs="0" name="tariffId"
type="ns_p:TariffIdType"/>
                            </xs:sequence>
                        </xs:restriction>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="incentiveTableConstraintsDataSelectors"
type="ns_p:IncentiveTableConstraintsDataSelectorsType"/>
</xs:schema>
```

A.1.2 NodeManagement

File name: EEBus_SPINE_TS_NodeManagement.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
    Smart Premises Interoperable Neutral-Message Exchange (SPINE)
    Version 1.1.1
    2018-12-21
    Copyright (c) 2018 EEBus Initiative e.V.  All Rights Reserved.
    Source: https://www.eebus.org/en/specifications/
-->
<xs:schema xmlns:ns_p="http://docs.eebus.org/spine/xsd/v1"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://docs.eebus.org/spine/xsd/v1" version="1.1.1" blockDefault="#all"
    elementFormDefault="qualified">
    <xs:annotation>
        <xs:documentation>EEBus SPINE Specification schema. Copyright 2018 EEBus Initiative
e.V.  All rights reserved.</xs:documentation>
    </xs:annotation>
    <xs:include schemaLocation="EEBus_SPINE_TS_BindingManagement.xsd"/>
    <xs:include schemaLocation="EEBus_SPINE_TS_CommonDataTypes.xsd"/>
    <xs:include schemaLocation="EEBus_SPINE_TS_NetworkManagement.xsd"/>
    <xs:include schemaLocation="EEBus_SPINE_TS_SubscriptionManagement.xsd"/>
    <xs:include schemaLocation="EEBus_SPINE_TS_UseCaseInformation.xsd"/>
    <xs:include schemaLocation="EEBus_SPINE_TS_Version.xsd"/>
    <xs:complexType name="NodeManagementSpecificationVersionListType">
        <xs:sequence>
            <xs:element maxOccurs="unbounded" minOccurs="0" name="specificationVersion"
type="ns_p:SpecificationVersionDataType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="NodeManagementDetailedDiscoveryDeviceInformationType">
        <xs:sequence>
            <xs:element name="description" minOccurs="0">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:restriction
base="ns_p:NetworkManagementDeviceDescriptionDataType">
                            <xs:sequence>
                                <xs:element name="deviceAddress" minOccurs="0">
```

```
                                    <xs:complexType>
                                        <xs:complexContent>
                                            <xs:restriction base="ns_p:DeviceAddressType">
                                                <xs:sequence>
                                                    <xs:element ref="ns_p:device"
minOccurs="0"/>
                                                </xs:sequence>
                                            </xs:restriction>
                                        </xs:complexContent>
                                    </xs:complexType>
                                </xs:element>
                                <xs:element name="deviceType" type="ns_p:DeviceTypeType"
minOccurs="0"/>
                                <xs:element minOccurs="0"
name="networkManagementResponsibleAddress" type="ns_p:FeatureAddressType"/>
                                <xs:element minOccurs="0" name="nativeSetup"
type="ns_p:NetworkManagementNativeSetupType"/>
                                <xs:element minOccurs="0" name="technologyAddress"
type="ns_p:NetworkManagementTechnologyAddressType"/>
                                <xs:element minOccurs="0"
name="communicationsTechnologyInformation"
type="ns_p:NetworkManagementCommunicationsTechnologyInformationType"/>
                                <xs:element minOccurs="0" name="networkFeatureSet"
type="ns_p:NetworkManagementFeatureSetType"/>
                                <xs:element minOccurs="0" name="lastStateChange"
type="ns_p:NetworkManagementStateChangeType"/>
                                <xs:element minOccurs="0" name="minimumTrustLevel"
type="ns_p:NetworkManagementMinimumTrustLevelType"/>
                                <xs:element minOccurs="0" name="label" type="ns_p:LabelType"/>
                                <xs:element minOccurs="0" name="description"
type="ns_p:DescriptionType"/>
                            </xs:sequence>
                        </xs:restriction>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="NodeManagementDetailedDiscoveryEntityInformationType">
        <xs:sequence>
            <xs:element name="description" minOccurs="0">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:restriction
base="ns_p:NetworkManagementEntityDescriptionDataType">
                            <xs:sequence>
                                <xs:element name="entityAddress" minOccurs="0">
                                    <xs:complexType>
                                        <xs:complexContent>
                                            <xs:restriction base="ns_p:EntityAddressType">
                                                <xs:sequence>
                                                    <xs:sequence>
                                                        <xs:element maxOccurs="unbounded"
ref="ns_p:entity" minOccurs="0"/>
                                                    </xs:sequence>
                                                </xs:sequence>
                                            </xs:restriction>
                                        </xs:complexContent>
                                    </xs:complexType>
                                </xs:element>
                                <xs:element name="entityType" type="ns_p:EntityTypeType"
minOccurs="0"/>
                                <xs:element minOccurs="0" name="lastStateChange"
type="ns_p:NetworkManagementStateChangeType"/>
                                <xs:element minOccurs="0" name="minimumTrustLevel"
type="ns_p:NetworkManagementMinimumTrustLevelType"/>
                                <xs:element minOccurs="0" name="label" type="ns_p:LabelType"/>
                                <xs:element minOccurs="0" name="description"
type="ns_p:DescriptionType"/>
                            </xs:sequence>
                        </xs:restriction>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="NodeManagementDetailedDiscoveryFeatureInformationType">
        <xs:sequence>
```



```
            <xs:element name="description" minOccurs="0">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:restriction
base="ns_p:NetworkManagementFeatureDescriptionDataType">
                            <xs:sequence>
                                <xs:element name="featureAddress" minOccurs="0">
                                    <xs:complexType>
                                        <xs:complexContent>
                                            <xs:restriction base="ns_p:FeatureAddressType">
                                                <xs:sequence>
                                                    <xs:sequence>
                                                        <xs:element maxOccurs="unbounded"
ref="ns_p:entity" minOccurs="0"/>
                                                    </xs:sequence>
                                                    <xs:sequence>
                                                        <xs:element ref="ns_p:feature"
minOccurs="0"/>
                                                    </xs:sequence>
                                                </xs:sequence>
                                            </xs:restriction>
                                        </xs:complexContent>
                                    </xs:complexType>
                                </xs:element>
                                <xs:element minOccurs="0" name="featureType"
type="ns_p:FeatureTypeType"/>
                                <xs:element maxOccurs="unbounded" minOccurs="0"
name="specificUsage" type="ns_p:FeatureSpecificUsageType"/>
                                <xs:element minOccurs="0" name="featureGroup"
type="ns_p:FeatureGroupType"/>
                                <xs:element name="role" type="ns_p:RoleType" minOccurs="0"/>
                                <xs:element minOccurs="0" name="supportedFunction"
type="ns_p:FunctionPropertyType" maxOccurs="unbounded"/>
                                <xs:element minOccurs="0" name="lastStateChange"
type="ns_p:NetworkManagementStateChangeType"/>
                                <xs:element minOccurs="0" name="minimumTrustLevel"
type="ns_p:NetworkManagementMinimumTrustLevelType"/>
                                <xs:element minOccurs="0" name="label" type="ns_p:LabelType"/>
                                <xs:element minOccurs="0" name="description"
type="ns_p:DescriptionType"/>
                                <xs:element minOccurs="0" name="maxResponseDelay"
type="ns_p:MaxResponseDelayType"/>
                            </xs:sequence>
                        </xs:restriction>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="NodeManagementDetailedDiscoveryDataType">
        <xs:sequence>
            <xs:element maxOccurs="1" name="specificationVersionList" minOccurs="0"
type="ns_p:NodeManagementSpecificationVersionListType"/>
            <xs:element name="deviceInformation"
type="ns_p:NodeManagementDetailedDiscoveryDeviceInformationType" minOccurs="0"/>
            <xs:element name="entityInformation" maxOccurs="unbounded" minOccurs="0"
type="ns_p:NodeManagementDetailedDiscoveryEntityInformationType"/>
            <xs:element name="featureInformation" maxOccurs="unbounded" minOccurs="0"
type="ns_p:NodeManagementDetailedDiscoveryFeatureInformationType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="nodeManagementDetailedDiscoveryData"
type="ns_p:NodeManagementDetailedDiscoveryDataType"/>
    <xs:complexType name="NodeManagementSpecificationVersionListElementsType">
        <xs:sequence>
            <xs:element maxOccurs="1" name="specificationVersion" minOccurs="0">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:restriction base="ns_p:SpecificationVersionDataElementsType"/>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="NodeManagementDetailedDiscoveryDeviceInformationElementsType">
        <xs:sequence>
            <xs:element name="description" minOccurs="0">
                <xs:complexType>
```



```
12093                     <xs:complexContent>
12094                         <xs:restriction
12095 base="ns_p:NetworkManagementDeviceDescriptionDataElementsType">
12096                             <xs:sequence>
12097                                 <xs:element name="deviceAddress" minOccurs="0">
12098                                     <xs:complexType>
12099                                         <xs:complexContent>
12100                                             <xs:restriction
12101 base="ns_p:DeviceAddressElementsType">
12102                                                 <xs:sequence>
12103                                                     <xs:element minOccurs="0" name="device"
12104 type="ns_p:ElementTagType"/>
12105                                                 </xs:sequence>
12106                                             </xs:restriction>
12107                                         </xs:complexContent>
12108                                     </xs:complexType>
12109                                 </xs:element>
12110                                 <xs:element name="deviceType" minOccurs="0"
12111 type="ns_p:ElementTagType"/>
12112                                 <xs:element minOccurs="0"
12113 name="networkManagementResponsibleAddress" type="ns_p:ElementTagType"/>
12114                                 <xs:element minOccurs="0" name="nativeSetup"
12115 type="ns_p:ElementTagType"/>
12116                                 <xs:element minOccurs="0" name="technologyAddress"
12117 type="ns_p:ElementTagType"/>
12118                                 <xs:element minOccurs="0"
12119 name="communicationsTechnologyInformation" type="ns_p:ElementTagType"/>
12120                                 <xs:element minOccurs="0" name="networkFeatureSet"
12121 type="ns_p:ElementTagType"/>
12122                                 <xs:element minOccurs="0" name="lastStateChange"
12123 type="ns_p:ElementTagType"/>
12124                                 <xs:element minOccurs="0" name="minimumTrustLevel"
12125 type="ns_p:ElementTagType"/>
12126                                 <xs:element minOccurs="0" name="label"
12127 type="ns_p:ElementTagType"/>
12128                                 <xs:element minOccurs="0" name="description"
12129 type="ns_p:ElementTagType"/>
12130                             </xs:sequence>
12131                         </xs:restriction>
12132                     </xs:complexContent>
12133                 </xs:complexType>
12134             </xs:element>
12135         </xs:sequence>
12136     </xs:complexType>
12137     <xs:complexType name="NodeManagementDetailedDiscoveryEntityInformationElementsType">
12138         <xs:sequence>
12139             <xs:element name="description" minOccurs="0">
12140                 <xs:complexType>
12141                     <xs:complexContent>
12142                         <xs:restriction
12143 base="ns_p:NetworkManagementEntityDescriptionDataElementsType">
12144                             <xs:sequence>
12145                                 <xs:element name="entityAddress" minOccurs="0">
12146                                     <xs:complexType>
12147                                         <xs:complexContent>
12148                                             <xs:restriction
12149 base="ns_p:EntityAddressElementsType">
12150                                                 <xs:sequence>
12151                                                     <xs:element minOccurs="0" name="entity"
12152 type="ns_p:ElementTagType"/>
12153                                                 </xs:sequence>
12154                                             </xs:restriction>
12155                                         </xs:complexContent>
12156                                     </xs:complexType>
12157                                 </xs:element>
12158                                 <xs:element name="entityType" minOccurs="0"
12159 type="ns_p:ElementTagType"/>
12160                                 <xs:element minOccurs="0" name="lastStateChange"
12161 type="ns_p:ElementTagType"/>
12162                                 <xs:element minOccurs="0" name="minimumTrustLevel"
12163 type="ns_p:ElementTagType"/>
12164                                 <xs:element minOccurs="0" name="label"
12165 type="ns_p:ElementTagType"/>
12166                                 <xs:element minOccurs="0" name="description"
12167 type="ns_p:ElementTagType"/>
12168                             </xs:sequence>
12169                         </xs:restriction>
12170                     </xs:complexContent>
```



```
12171                 </xs:complexType>
12172             </xs:element>
12173         </xs:sequence>
12174     </xs:complexType>
12175     <xs:complexType name="NodeManagementDetailedDiscoveryFeatureInformationElementsType">
12176         <xs:sequence>
12177             <xs:element name="description" minOccurs="0">
12178                 <xs:complexType>
12179                     <xs:complexContent>
12180                         <xs:restriction
12181 base="ns_p:NetworkManagementFeatureDescriptionDataElementsType">
12182                             <xs:sequence>
12183                                 <xs:element name="featureAddress" minOccurs="0">
12184                                     <xs:complexType>
12185                                         <xs:complexContent>
12186                                             <xs:restriction
12187 base="ns_p:FeatureAddressElementsType">
12188                                                 <xs:sequence>
12189                                                     <xs:element minOccurs="0" name="entity"
12190 type="ns_p:ElementTagType"/>
12191                                                     <xs:element minOccurs="0" name="feature"
12192 type="ns_p:ElementTagType"/>
12193                                                 </xs:sequence>
12194                                             </xs:restriction>
12195                                         </xs:complexContent>
12196                                     </xs:complexType>
12197                                 </xs:element>
12198                                 <xs:element minOccurs="0" name="featureType"
12199 type="ns_p:ElementTagType"/>
12200                                 <xs:element minOccurs="0" name="specificUsage"
12201 type="ns_p:ElementTagType"/>
12202                                 <xs:element minOccurs="0" name="featureGroup"
12203 type="ns_p:ElementTagType"/>
12204                                 <xs:element minOccurs="0" name="role"
12205 type="ns_p:ElementTagType"/>
12206                                 <xs:element minOccurs="0" name="supportedFunction"
12207 type="ns_p:FunctionPropertyElementsType"/>
12208                                 <xs:element minOccurs="0" name="lastStateChange"
12209 type="ns_p:ElementTagType"/>
12210                                 <xs:element minOccurs="0" name="minimumTrustLevel"
12211 type="ns_p:ElementTagType"/>
12212                                 <xs:element minOccurs="0" name="label"
12213 type="ns_p:ElementTagType"/>
12214                                 <xs:element minOccurs="0" name="description"
12215 type="ns_p:ElementTagType"/>
12216                                 <xs:element minOccurs="0" name="maxResponseDelay"
12217 type="ns_p:ElementTagType"/>
12218                             </xs:sequence>
12219                         </xs:restriction>
12220                     </xs:complexContent>
12221                 </xs:complexType>
12222             </xs:element>
12223         </xs:sequence>
12224     </xs:complexType>
12225     <xs:complexType name="NodeManagementDetailedDiscoveryDataElementsType">
12226         <xs:sequence>
12227             <xs:element maxOccurs="1" name="specificationVersionList" minOccurs="0"
12228 type="ns_p:NodeManagementSpecificationVersionListElementsType"/>
12229             <xs:element name="deviceInformation"
12230 type="ns_p:NodeManagementDetailedDiscoveryDeviceInformationElementsType" minOccurs="0"/>
12231             <xs:element name="entityInformation" minOccurs="0"
12232 type="ns_p:NodeManagementDetailedDiscoveryEntityInformationElementsType"/>
12233             <xs:element name="featureInformation" minOccurs="0"
12234 type="ns_p:NodeManagementDetailedDiscoveryFeatureInformationElementsType"/>
12235         </xs:sequence>
12236     </xs:complexType>
12237     <xs:element name="nodeManagementDetailedDiscoveryDataElements"
12238 type="ns_p:NodeManagementDetailedDiscoveryDataElementsType"/>
12239     <xs:complexType name="NodeManagementDetailedDiscoveryDataSelectorsType">
12240         <xs:sequence>
12241             <xs:element minOccurs="0" name="deviceInformation">
12242                 <xs:complexType>
12243                     <xs:complexContent>
12244                         <xs:restriction
12245 base="ns_p:NetworkManagementDeviceDescriptionListDataSelectorsType">
12246                             <xs:sequence>
12247                                 <xs:element minOccurs="0" name="deviceAddress"
12248 type="ns_p:DeviceAddressType"/>
```



```
                                <xs:element minOccurs="0" name="deviceType"
type="ns_p:DeviceTypeType"/>
                            </xs:sequence>
                        </xs:restriction>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
            <xs:element minOccurs="0" name="entityInformation">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:restriction
base="ns_p:NetworkManagementEntityDescriptionListDataSelectorsType">
                            <xs:sequence>
                                <xs:element minOccurs="0" name="entityAddress"
type="ns_p:EntityAddressType"/>
                                <xs:element minOccurs="0" name="entityType"
type="ns_p:EntityTypeType"/>
                            </xs:sequence>
                        </xs:restriction>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
            <xs:element minOccurs="0" name="featureInformation">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:restriction
base="ns_p:NetworkManagementFeatureDescriptionListDataSelectorsType">
                            <xs:sequence>
                                <xs:element minOccurs="0" name="featureAddress"
type="ns_p:FeatureAddressType"/>
                                <xs:element minOccurs="0" name="featureType"
type="ns_p:FeatureTypeType"/>
                            </xs:sequence>
                        </xs:restriction>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="nodeManagementDetailedDiscoveryDataSelectors"
type="ns_p:NodeManagementDetailedDiscoveryDataSelectorsType"/>
    <xs:complexType name="NodeManagementBindingDataType">
        <xs:sequence>
            <xs:element maxOccurs="unbounded" minOccurs="0" name="bindingEntry">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:restriction base="ns_p:BindingManagementEntryDataType">
                            <xs:sequence>
                                <xs:element name="bindingId" type="ns_p:BindingIdType"
minOccurs="0"/>
                                <xs:element name="clientAddress" minOccurs="0"
type="ns_p:FeatureAddressType"/>
                                <xs:element name="serverAddress" minOccurs="0"
type="ns_p:FeatureAddressType"/>
                                <xs:element minOccurs="0" name="label" type="ns_p:LabelType"/>
                                <xs:element minOccurs="0" name="description"
type="ns_p:DescriptionType"/>
                            </xs:sequence>
                        </xs:restriction>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="nodeManagementBindingData" type="ns_p:NodeManagementBindingDataType"/>
    <xs:complexType name="NodeManagementBindingDataElementsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="bindingEntry">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:restriction base="ns_p:BindingManagementEntryDataElementsType">
                            <xs:sequence>
                                <xs:element minOccurs="0" name="bindingId"
type="ns_p:ElementTagType"/>
                                <xs:element minOccurs="0" name="clientAddress"
type="ns_p:FeatureAddressElementsType"/>
                                <xs:element minOccurs="0" name="serverAddress"
type="ns_p:FeatureAddressElementsType"/>
```



```
                                <xs:element minOccurs="0" name="label"
type="ns_p:ElementTagType"/>
                                <xs:element minOccurs="0" name="description"
type="ns_p:ElementTagType"/>
                            </xs:sequence>
                        </xs:restriction>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="nodeManagementBindingDataElements"
type="ns_p:NodeManagementBindingDataElementsType"/>
    <xs:complexType name="NodeManagementBindingDataSelectorsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="bindingEntry">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:restriction
base="ns_p:BindingManagementEntryListDataSelectorsType"/>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="nodeManagementBindingDataSelectors"
type="ns_p:NodeManagementBindingDataSelectorsType"/>
    <xs:complexType name="NodeManagementBindingRequestCallType">
        <xs:sequence>
            <xs:element minOccurs="0" name="bindingRequest">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:restriction base="ns_p:BindingManagementRequestCallType">
                            <xs:sequence>
                                <xs:element minOccurs="0" name="clientAddress"
type="ns_p:FeatureAddressType"/>
                                <xs:element minOccurs="0" name="serverAddress"
type="ns_p:FeatureAddressType"/>
                                <xs:element minOccurs="0" name="serverFeatureType"
type="ns_p:FeatureTypeType"/>
                            </xs:sequence>
                        </xs:restriction>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="nodeManagementBindingRequestCall"
type="ns_p:NodeManagementBindingRequestCallType"/>
    <xs:complexType name="NodeManagementBindingRequestCallElementsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="bindingRequest">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:restriction base="ns_p:BindingManagementRequestCallElementsType">
                            <xs:sequence>
                                <xs:element minOccurs="0" name="clientAddress"
type="ns_p:FeatureAddressElementsType"/>
                                <xs:element minOccurs="0" name="serverAddress"
type="ns_p:FeatureAddressElementsType"/>
                                <xs:element minOccurs="0" name="serverFeatureType"
type="ns_p:ElementTagType"/>
                            </xs:sequence>
                        </xs:restriction>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="nodeManagementBindingRequestCallElements"
type="ns_p:NodeManagementBindingRequestCallElementsType"/>
    <xs:complexType name="NodeManagementBindingDeleteCallType">
        <xs:sequence>
            <xs:element minOccurs="0" name="bindingDelete">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:restriction base="ns_p:BindingManagementDeleteCallType">
                            <xs:sequence>
```



```
                                <xs:element minOccurs="0" name="bindingId"
type="ns_p:BindingIdType"/>
                                <xs:element minOccurs="0" name="clientAddress"
type="ns_p:FeatureAddressType"/>
                                <xs:element minOccurs="0" name="serverAddress"
type="ns_p:FeatureAddressType"/>
                            </xs:sequence>
                        </xs:restriction>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="nodeManagementBindingDeleteCall"
type="ns_p:NodeManagementBindingDeleteCallType"/>
    <xs:complexType name="NodeManagementBindingDeleteCallElementsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="bindingDelete">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:restriction base="ns_p:BindingManagementDeleteCallElementsType">
                            <xs:sequence>
                                <xs:element minOccurs="0" name="bindingId"
type="ns_p:ElementTagType"/>
                                <xs:element minOccurs="0" name="clientAddress"
type="ns_p:FeatureAddressElementsType"/>
                                <xs:element minOccurs="0" name="serverAddress"
type="ns_p:FeatureAddressElementsType"/>
                            </xs:sequence>
                        </xs:restriction>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="nodeManagementBindingDeleteCallElements"
type="ns_p:NodeManagementBindingDeleteCallElementsType"/>
    <xs:complexType name="NodeManagementSubscriptionDataType">
        <xs:sequence>
            <xs:element maxOccurs="unbounded" minOccurs="0" name="subscriptionEntry">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:restriction base="ns_p:SubscriptionManagementEntryDataType">
                            <xs:sequence>
                                <xs:element name="subscriptionId"
type="ns_p:SubscriptionIdType" minOccurs="0"/>
                                <xs:element name="clientAddress" minOccurs="0"
type="ns_p:FeatureAddressType"/>
                                <xs:element name="serverAddress" minOccurs="0"
type="ns_p:FeatureAddressType"/>
                                <xs:element minOccurs="0" name="label" type="ns_p:LabelType"/>
                                <xs:element minOccurs="0" name="description"
type="ns_p:DescriptionType"/>
                            </xs:sequence>
                        </xs:restriction>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="nodeManagementSubscriptionData"
type="ns_p:NodeManagementSubscriptionDataType"/>
    <xs:complexType name="NodeManagementSubscriptionDataElementsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="subscriptionEntry">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:restriction
base="ns_p:SubscriptionManagementEntryDataElementsType">
                            <xs:sequence>
                                <xs:element minOccurs="0" name="subscriptionId"
type="ns_p:ElementTagType"/>
                                <xs:element minOccurs="0" name="clientAddress"
type="ns_p:FeatureAddressElementsType"/>
                                <xs:element minOccurs="0" name="serverAddress"
type="ns_p:FeatureAddressElementsType"/>
                                <xs:element minOccurs="0" name="label"
type="ns_p:ElementTagType"/>
```



```
                                <xs:element minOccurs="0" name="description"
type="ns_p:ElementTagType"/>
                            </xs:sequence>
                        </xs:restriction>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="nodeManagementSubscriptionDataElements"
type="ns_p:NodeManagementSubscriptionDataElementsType"/>
    <xs:complexType name="NodeManagementSubscriptionDataSelectorsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="subscriptionEntry">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:restriction
base="ns_p:SubscriptionManagementEntryListDataSelectorsType">
                            <xs:sequence>
                                <xs:element minOccurs="0" name="subscriptionId"
type="ns_p:SubscriptionIdType"/>
                                <xs:element minOccurs="0" name="clientAddress"
type="ns_p:FeatureAddressType"/>
                                <xs:element minOccurs="0" name="serverAddress"
type="ns_p:FeatureAddressType"/>
                            </xs:sequence>
                        </xs:restriction>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="nodeManagementSubscriptionDataSelectors"
type="ns_p:NodeManagementSubscriptionDataSelectorsType"/>
    <xs:complexType name="NodeManagementSubscriptionRequestCallType">
        <xs:sequence>
            <xs:element minOccurs="0" name="subscriptionRequest">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:restriction base="ns_p:SubscriptionManagementRequestCallType">
                            <xs:sequence>
                                <xs:element minOccurs="0" name="clientAddress"
type="ns_p:FeatureAddressType"/>
                                <xs:element minOccurs="0" name="serverAddress"
type="ns_p:FeatureAddressType"/>
                                <xs:element minOccurs="0" name="serverFeatureType"
type="ns_p:FeatureTypeType"/>
                            </xs:sequence>
                        </xs:restriction>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="nodeManagementSubscriptionRequestCall"
type="ns_p:NodeManagementSubscriptionRequestCallType"/>
    <xs:complexType name="NodeManagementSubscriptionRequestCallElementsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="subscriptionRequest">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:restriction
base="ns_p:SubscriptionManagementRequestCallElementsType">
                            <xs:sequence>
                                <xs:element minOccurs="0" name="clientAddress"
type="ns_p:FeatureAddressElementsType"/>
                                <xs:element minOccurs="0" name="serverAddress"
type="ns_p:FeatureAddressElementsType"/>
                                <xs:element minOccurs="0" name="serverFeatureType"
type="ns_p:ElementTagType"/>
                            </xs:sequence>
                        </xs:restriction>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
```



```
12560       <xs:element name="nodeManagementSubscriptionRequestCallElements"
12561   type="ns_p:NodeManagementSubscriptionRequestCallElementsType"/>
12562       <xs:complexType name="NodeManagementSubscriptionDeleteCallType">
12563           <xs:sequence>
12564               <xs:element minOccurs="0" name="subscriptionDelete">
12565                   <xs:complexType>
12566                       <xs:complexContent>
12567                           <xs:restriction base="ns_p:SubscriptionManagementDeleteCallType">
12568                               <xs:sequence>
12569                                   <xs:element minOccurs="0" name="subscriptionId"
12570   type="ns_p:SubscriptionIdType"/>
12571                                   <xs:element minOccurs="0" name="clientAddress"
12572   type="ns_p:FeatureAddressType"/>
12573                                   <xs:element minOccurs="0" name="serverAddress"
12574   type="ns_p:FeatureAddressType"/>
12575                               </xs:sequence>
12576                           </xs:restriction>
12577                       </xs:complexContent>
12578                   </xs:complexType>
12579               </xs:element>
12580           </xs:sequence>
12581       </xs:complexType>
12582       <xs:element name="nodeManagementSubscriptionDeleteCall"
12583   type="ns_p:NodeManagementSubscriptionDeleteCallType"/>
12584       <xs:complexType name="NodeManagementSubscriptionDeleteCallElementsType">
12585           <xs:sequence>
12586               <xs:element minOccurs="0" name="subscriptionDelete">
12587                   <xs:complexType>
12588                       <xs:complexContent>
12589                           <xs:restriction
12590   base="ns_p:SubscriptionManagementDeleteCallElementsType">
12591                               <xs:sequence>
12592                                   <xs:element minOccurs="0" name="subscriptionId"
12593   type="ns_p:ElementTagType"/>
12594                                   <xs:element minOccurs="0" name="clientAddress"
12595   type="ns_p:FeatureAddressElementsType"/>
12596                                   <xs:element minOccurs="0" name="serverAddress"
12597   type="ns_p:FeatureAddressElementsType"/>
12598                               </xs:sequence>
12599                           </xs:restriction>
12600                       </xs:complexContent>
12601                   </xs:complexType>
12602               </xs:element>
12603           </xs:sequence>
12604       </xs:complexType>
12605       <xs:element name="nodeManagementSubscriptionDeleteCallElements"
12606   type="ns_p:NodeManagementSubscriptionDeleteCallElementsType"/>
12607       <xs:complexType name="NodeManagementDestinationDataType">
12608           <xs:sequence>
12609               <xs:element minOccurs="0" name="deviceDescription">
12610                   <xs:complexType>
12611                       <xs:complexContent>
12612                           <xs:restriction
12613   base="ns_p:NetworkManagementDeviceDescriptionDataType">
12614                               <xs:sequence>
12615                                   <xs:element minOccurs="0" name="deviceAddress"
12616   type="ns_p:DeviceAddressType"/>
12617                                   <xs:element minOccurs="0"
12618   name="communicationsTechnologyInformation"
12619   type="ns_p:NetworkManagementCommunicationsTechnologyInformationType"/>
12620                                   <xs:element minOccurs="0" name="networkFeatureSet"
12621   type="ns_p:NetworkManagementFeatureSetType"/>
12622                                   <xs:element minOccurs="0" name="lastStateChange"
12623   type="ns_p:NetworkManagementStateChangeType"/>
12624                                   <xs:element minOccurs="0" name="label" type="ns_p:LabelType"/>
12625                               </xs:sequence>
12626                           </xs:restriction>
12627                       </xs:complexContent>
12628                   </xs:complexType>
12629               </xs:element>
12630           </xs:sequence>
12631       </xs:complexType>
12632       <xs:element name="nodeManagementDestinationData"
12633   type="ns_p:NodeManagementDestinationDataType"/>
12634       <xs:complexType name="NodeManagementDestinationDataElementsType">
12635           <xs:sequence>
12636               <xs:element minOccurs="0" name="deviceDescription">
12637                   <xs:complexType>
```



```
                    <xs:complexContent>
                        <xs:restriction
base="ns_p:NetworkManagementDeviceDescriptionDataElementsType">
                            <xs:sequence>
                                <xs:element minOccurs="0" name="deviceAddress"
type="ns_p:DeviceAddressElementsType"/>
                                <xs:element minOccurs="0"
name="communicationsTechnologyInformation" type="ns_p:ElementTagType"/>
                                <xs:element minOccurs="0" name="networkFeatureSet"
type="ns_p:ElementTagType"/>
                                <xs:element minOccurs="0" name="lastStateChange"
type="ns_p:ElementTagType"/>
                                <xs:element minOccurs="0" name="label"
type="ns_p:ElementTagType"/>
                            </xs:sequence>
                        </xs:restriction>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="nodeManagementDestinationDataElements"
type="ns_p:NodeManagementDestinationDataElementsType"/>
    <xs:complexType name="NodeManagementDestinationListDataType">
        <xs:sequence>
            <xs:element maxOccurs="unbounded" minOccurs="0"
ref="ns_p:nodeManagementDestinationData"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="nodeManagementDestinationListData"
type="ns_p:NodeManagementDestinationListDataType"/>
    <xs:complexType name="NodeManagementDestinationListDataSelectorsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="deviceDescription">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:restriction
base="ns_p:NetworkManagementDeviceDescriptionListDataSelectorsType">
                            <xs:sequence>
                                <xs:element minOccurs="0" name="deviceAddress"
type="ns_p:DeviceAddressType"/>
                            </xs:sequence>
                        </xs:restriction>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="nodeManagementDestinationListDataSelectors"
type="ns_p:NodeManagementDestinationListDataSelectorsType"/>
    <xs:complexType name="NodeManagementUseCaseDataType">
        <xs:sequence>
            <xs:element maxOccurs="unbounded" minOccurs="0" name="useCaseInformation">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:restriction base="ns_p:UseCaseInformationDataType">
                            <xs:sequence>
                                <xs:element minOccurs="0" name="address"
type="ns_p:FeatureAddressType"/>
                                <xs:element minOccurs="0" name="actor"
type="ns_p:UseCaseActorType"/>
                                <xs:element maxOccurs="unbounded" minOccurs="0"
name="useCaseSupport">
                                    <xs:complexType>
                                        <xs:complexContent>
                                            <xs:restriction base="ns_p:UseCaseSupportType">
                                                <xs:sequence>
                                                    <xs:element minOccurs="0"
name="useCaseName" type="ns_p:UseCaseNameType"/>
                                                    <xs:element minOccurs="0"
name="useCaseVersion" type="ns_p:SpecificationVersionType"/>
                                                    <xs:element minOccurs="0"
name="useCaseAvailable" type="xs:boolean"/>
                                                    <xs:element maxOccurs="unbounded"
minOccurs="0" name="scenarioSupport" type="ns_p:UseCaseScenarioSupportType"/>
                                                </xs:sequence>
                                            </xs:restriction>
                                        </xs:complexContent>
```



```
                                    </xs:complexType>
                                </xs:element>
                            </xs:sequence>
                        </xs:restriction>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="nodeManagementUseCaseData" type="ns_p:NodeManagementUseCaseDataType"/>
    <xs:complexType name="NodeManagementUseCaseDataElementsType">
        <xs:sequence>
            <xs:element maxOccurs="1" minOccurs="0" name="useCaseInformation">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:restriction base="ns_p:UseCaseInformationDataElementsType">
                            <xs:sequence>
                                <xs:element minOccurs="0" name="address"
type="ns_p:FeatureAddressElementsType"/>
                                <xs:element minOccurs="0" name="actor"
type="ns_p:ElementTagType"/>
                                <xs:element minOccurs="0" name="useCaseSupport">
                                    <xs:complexType>
                                        <xs:complexContent>
                                            <xs:restriction
base="ns_p:UseCaseSupportElementsType">
                                                <xs:sequence>
                                                    <xs:element minOccurs="0"
name="useCaseName" type="ns_p:ElementTagType"/>
                                                    <xs:element minOccurs="0"
name="useCaseVersion" type="ns_p:ElementTagType"/>
                                                    <xs:element minOccurs="0"
name="useCaseAvailable" type="ns_p:ElementTagType"/>
                                                    <xs:element minOccurs="0"
name="scenarioSupport" type="ns_p:ElementTagType"/>
                                                </xs:sequence>
                                            </xs:restriction>
                                        </xs:complexContent>
                                    </xs:complexType>
                                </xs:element>
                            </xs:sequence>
                        </xs:restriction>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="nodeManagementUseCaseDataElements"
type="ns_p:NodeManagementUseCaseDataElementsType"/>
    <xs:complexType name="NodeManagementUseCaseDataSelectorsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="useCaseInformation">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:restriction base="ns_p:UseCaseInformationListDataSelectorsType">
                            <xs:sequence>
                                <xs:element minOccurs="0" name="address"
type="ns_p:FeatureAddressType"/>
                                <xs:element minOccurs="0" name="actor"
type="ns_p:UseCaseActorType"/>
                                <xs:element minOccurs="0" name="useCaseSupport">
                                    <xs:complexType>
                                        <xs:complexContent>
                                            <xs:restriction
base="ns_p:UseCaseSupportSelectorsType">
                                                <xs:sequence>
                                                    <xs:element minOccurs="0"
name="useCaseName" type="ns_p:UseCaseNameType"/>
                                                    <xs:element minOccurs="0"
name="useCaseVersion" type="ns_p:SpecificationVersionType"/>
                                                    <xs:element minOccurs="0"
name="scenarioSupport" type="ns_p:UseCaseScenarioSupportType"/>
                                                </xs:sequence>
                                            </xs:restriction>
                                        </xs:complexContent>
                                    </xs:complexType>
                                </xs:element>
                            </xs:sequence>
```



```
                        </xs:restriction>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="nodeManagementUseCaseDataSelectors"
type="ns_p:NodeManagementUseCaseDataSelectorsType"/>
</xs:schema>
```

A.1.3 SmartEnergyManagementPs

File name: EEBus_SPINE_TS_SmartEnergyManagementPs.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
    Smart Premises Interoperable Neutral-Message Exchange (SPINE)
    Version 1.1.1
    2018-12-21
    Copyright (c) 2018 EEBus Initiative e.V.  All Rights Reserved.
    Source: https://www.eebus.org/en/specifications/
-->
<xs:schema xmlns:ns_p="http://docs.eebus.org/spine/xsd/v1"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://docs.eebus.org/spine/xsd/v1" version="1.1.1" blockDefault="#all"
elementFormDefault="qualified">
    <xs:annotation>
        <xs:documentation>EEBus SPINE Specification schema. Copyright 2018 EEBus Initiative
e.V.  All rights reserved.</xs:documentation>
    </xs:annotation>
    <xs:include schemaLocation="EEBus_SPINE_TS_CommonDataTypes.xsd"/>
    <xs:include schemaLocation="EEBus_SPINE_TS_OperatingConstraints.xsd"/>
    <xs:include schemaLocation="EEBus_SPINE_TS_PowerSequences.xsd"/>
    <xs:complexType name="SmartEnergyManagementPsAlternativesRelationType">
        <xs:complexContent>
            <xs:restriction base="ns_p:PowerSequenceAlternativesRelationDataType">
                <xs:sequence>
                    <xs:element minOccurs="0" name="alternativesId"
type="ns_p:AlternativesIdType"/>
                </xs:sequence>
            </xs:restriction>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="SmartEnergyManagementPsAlternativesType">
        <xs:sequence>
            <xs:element minOccurs="0" name="relation"
type="ns_p:SmartEnergyManagementPsAlternativesRelationType"/>
            <xs:element name="powerSequence" maxOccurs="unbounded" minOccurs="0"
type="ns_p:SmartEnergyManagementPsPowerSequenceType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="SmartEnergyManagementPsPowerSequenceType">
        <xs:sequence>
            <xs:element name="description" minOccurs="0">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:restriction base="ns_p:PowerSequenceDescriptionDataType">
                            <xs:sequence>
                                <xs:element name="sequenceId" type="ns_p:PowerSequenceIdType"
minOccurs="0"/>
                                <xs:element minOccurs="0" name="description">
                                    <xs:simpleType>
                                        <xs:restriction base="ns_p:DescriptionType">
                                            <xs:minLength value="1"/>
                                            <xs:maxLength value="60"/>
                                        </xs:restriction>
                                    </xs:simpleType>
                                </xs:element>
                                <xs:element minOccurs="0" name="powerUnit"
type="ns_p:UnitOfMeasurementType"/>
                                <xs:element minOccurs="0" name="energyUnit"
type="ns_p:UnitOfMeasurementType"/>
```

```
                                <xs:element minOccurs="0" name="valueSource"
type="ns_p:MeasurementValueSourceType"/>
                                <xs:element minOccurs="0" name="taskIdentifier"
type="xs:unsignedInt"/>
                                <xs:element minOccurs="0" name="repetitionsTotal"
type="xs:unsignedInt"/>
                            </xs:sequence>
                        </xs:restriction>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
            <xs:element name="state" minOccurs="0">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:restriction base="ns_p:PowerSequenceStateDataType">
                            <xs:sequence>
                                <xs:element name="state" type="ns_p:PowerSequenceStateType"
minOccurs="0"/>
                                <xs:element minOccurs="0" name="activeSlotNumber"
type="ns_p:PowerTimeSlotNumberType"/>
                                <xs:element minOccurs="0" name="elapsedSlotTime"
type="xs:duration"/>
                                <xs:element minOccurs="0" name="remainingSlotTime"
type="xs:duration"/>
                                <xs:element name="sequenceRemoteControllable"
type="xs:boolean" minOccurs="0"/>
                                <xs:element minOccurs="0" name="activeRepetitionNumber"
type="xs:unsignedInt"/>
                                <xs:element minOccurs="0" name="remainingPauseTime"
type="xs:duration"/>
                            </xs:sequence>
                        </xs:restriction>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
            <xs:element minOccurs="0" name="schedule">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:restriction base="ns_p:PowerSequenceScheduleDataType">
                            <xs:sequence>
                                <xs:element name="startTime" type="xs:duration"
minOccurs="0"/>
                                <xs:element name="endTime" type="xs:duration" minOccurs="0"/>
                            </xs:sequence>
                        </xs:restriction>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
            <xs:element name="scheduleConstraints" minOccurs="0">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:restriction base="ns_p:PowerSequenceScheduleConstraintsDataType">
                            <xs:sequence>
                                <xs:element name="earliestStartTime" type="xs:duration"
minOccurs="0"/>
                                <xs:element name="latestEndTime" type="xs:duration"
minOccurs="0"/>
                            </xs:sequence>
                        </xs:restriction>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
            <xs:element minOccurs="0" name="schedulePreference">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:restriction base="ns_p:PowerSequenceSchedulePreferenceDataType">
                            <xs:sequence>
                                <xs:element minOccurs="0" name="greenest" type="xs:boolean"/>
                                <xs:element minOccurs="0" name="cheapest" type="xs:boolean"/>
                            </xs:sequence>
                        </xs:restriction>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
            <xs:element name="operatingConstraintsInterrupt" minOccurs="0">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:restriction base="ns_p:OperatingConstraintsInterruptDataType">
```



```
                            <xs:sequence>
                                <xs:element minOccurs="0" name="isPausable"
type="xs:boolean"/>
                                <xs:element minOccurs="0" name="isStoppable"
type="xs:boolean"/>
                                <xs:element minOccurs="0" name="maxCyclesPerDay"
type="xs:unsignedInt"/>
                            </xs:sequence>
                        </xs:restriction>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
            <xs:element name="operatingConstraintsDuration" minOccurs="0">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:restriction base="ns_p:OperatingConstraintsDurationDataType">
                            <xs:sequence>
                                <xs:element minOccurs="0" name="activeDurationMin"
type="xs:duration"/>
                                <xs:element minOccurs="0" name="activeDurationMax"
type="xs:duration"/>
                                <xs:element minOccurs="0" name="pauseDurationMin"
type="xs:duration"/>
                                <xs:element minOccurs="0" name="pauseDurationMax"
type="xs:duration"/>
                                <xs:element minOccurs="0" name="activeDurationSumMin"
type="xs:duration"/>
                                <xs:element minOccurs="0" name="activeDurationSumMax"
type="xs:duration"/>
                            </xs:sequence>
                        </xs:restriction>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
            <xs:element name="operatingConstraintsResumeImplication" minOccurs="0">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:restriction
base="ns_p:OperatingConstraintsResumeImplicationDataType">
                            <xs:sequence>
                                <xs:element minOccurs="0" name="resumeEnergyEstimated"
type="ns_p:ScaledNumberType"/>
                                <xs:element minOccurs="0" name="energyUnit"
type="ns_p:UnitOfMeasurementType"/>
                                <xs:element minOccurs="0" name="resumeCostEstimated"
type="ns_p:ScaledNumberType"/>
                                <xs:element minOccurs="0" name="currency"
type="ns_p:CurrencyType"/>
                            </xs:sequence>
                        </xs:restriction>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
            <xs:element maxOccurs="unbounded" minOccurs="0" name="powerTimeSlot"
type="ns_p:SmartEnergyManagementPsPowerTimeSlotType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="SmartEnergyManagementPsPowerTimeSlotType">
        <xs:sequence>
            <xs:element name="schedule" minOccurs="0">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:restriction base="ns_p:PowerTimeSlotScheduleDataType">
                            <xs:sequence>
                                <xs:element name="slotNumber"
type="ns_p:PowerTimeSlotNumberType" minOccurs="0"/>
                                <xs:element name="timePeriod" minOccurs="0">
                                    <xs:complexType>
                                        <xs:complexContent>
                                            <xs:restriction base="ns_p:TimePeriodType">
                                                <xs:sequence>
                                                    <xs:element name="startTime"
type="xs:duration" minOccurs="0"/>
                                                    <xs:element name="endTime"
type="xs:duration" minOccurs="0"/>
                                                </xs:sequence>
                                            </xs:restriction>
                                        </xs:complexContent>
```



```
                                    </xs:complexType>
                                </xs:element>
                                <xs:element name="defaultDuration" type="xs:duration"
minOccurs="0"/>
                                <xs:element minOccurs="0" name="durationUncertainty"
type="xs:duration"/>
                                <xs:element name="slotActivated" type="xs:boolean"
minOccurs="0"/>
                                <xs:element minOccurs="0" name="description">
                                    <xs:simpleType>
                                        <xs:restriction base="ns_p:DescriptionType">
                                            <xs:minLength value="1"/>
                                            <xs:maxLength value="60"/>
                                        </xs:restriction>
                                    </xs:simpleType>
                                </xs:element>
                            </xs:sequence>
                        </xs:restriction>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
            <xs:element minOccurs="0" name="valueList"
type="ns_p:SmartEnergyManagementPsPowerTimeSlotValueListType"/>
            <xs:element name="scheduleConstraints" minOccurs="0">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:restriction base="ns_p:PowerTimeSlotScheduleConstraintsDataType">
                            <xs:sequence>
                                <xs:element name="earliestStartTime" type="xs:duration"
minOccurs="0"/>
                                <xs:element name="latestEndTime" type="xs:duration"
minOccurs="0"/>
                                <xs:element minOccurs="0" name="minDuration"
type="xs:duration"/>
                                <xs:element minOccurs="0" name="maxDuration"
type="xs:duration"/>
                                <xs:element minOccurs="0" name="optionalSlot"
type="xs:boolean"/>
                            </xs:sequence>
                        </xs:restriction>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="SmartEnergyManagementPsPowerTimeSlotValueListType">
        <xs:sequence>
            <xs:element maxOccurs="unbounded" minOccurs="0" name="value">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:restriction base="ns_p:PowerTimeSlotValueDataType">
                            <xs:sequence>
                                <xs:element minOccurs="0" name="valueType"
type="ns_p:PowerTimeSlotValueTypeType"/>
                                <xs:element minOccurs="0" name="value"
type="ns_p:ScaledNumberType"/>
                            </xs:sequence>
                        </xs:restriction>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="SmartEnergyManagementPsDataType">
        <xs:sequence>
            <xs:element name="nodeScheduleInformation" minOccurs="0">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:restriction
base="ns_p:PowerSequenceNodeScheduleInformationDataType">
                            <xs:sequence>
                                <xs:element name="nodeRemoteControllable" type="xs:boolean"
minOccurs="0"/>
                                <xs:element name="supportsSingleSlotSchedulingOnly"
type="xs:boolean" minOccurs="0"/>
                                <xs:element name="alternativesCount" type="xs:unsignedInt"
minOccurs="0"/>
```



```
                                <xs:element name="totalSequencesCountMax"
type="xs:unsignedInt" minOccurs="0"/>
                                <xs:element name="supportsReselection" type="xs:boolean"
minOccurs="0"/>
                            </xs:sequence>
                        </xs:restriction>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
            <xs:element name="alternatives" maxOccurs="unbounded" minOccurs="0"
type="ns_p:SmartEnergyManagementPsAlternativesType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="smartEnergyManagementPsData"
type="ns_p:SmartEnergyManagementPsDataType"/>
    <xs:complexType name="SmartEnergyManagementPsAlternativesRelationElementsType">
        <xs:complexContent>
            <xs:restriction base="ns_p:PowerSequenceAlternativesRelationDataElementsType">
                <xs:sequence>
                    <xs:element minOccurs="0" name="alternativesId"
type="ns_p:ElementTagType"/>
                </xs:sequence>
            </xs:restriction>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="SmartEnergyManagementPsAlternativesElementsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="relation"
type="ns_p:SmartEnergyManagementPsAlternativesRelationElementsType"/>
            <xs:element name="powerSequence" minOccurs="0"
type="ns_p:SmartEnergyManagementPsPowerSequenceElementsType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="SmartEnergyManagementPsPowerSequenceElementsType">
        <xs:sequence>
            <xs:element name="description" minOccurs="0">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:restriction base="ns_p:PowerSequenceDescriptionDataElementsType">
                            <xs:sequence>
                                <xs:element name="sequenceId" minOccurs="0"
type="ns_p:ElementTagType"/>
                                <xs:element minOccurs="0" name="description"
type="ns_p:ElementTagType"/>
                                <xs:element minOccurs="0" name="powerUnit"
type="ns_p:ElementTagType"/>
                                <xs:element minOccurs="0" name="energyUnit"
type="ns_p:ElementTagType"/>
                                <xs:element minOccurs="0" name="valueSource"
type="ns_p:ElementTagType"/>
                                <xs:element minOccurs="0" name="taskIdentifier"
type="ns_p:ElementTagType"/>
                                <xs:element minOccurs="0" name="repetitionsTotal"
type="ns_p:ElementTagType"/>
                            </xs:sequence>
                        </xs:restriction>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
            <xs:element name="state" minOccurs="0">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:restriction base="ns_p:PowerSequenceStateDataElementsType">
                            <xs:sequence>
                                <xs:element name="state" minOccurs="0"
type="ns_p:ElementTagType"/>
                                <xs:element minOccurs="0" name="activeSlotNumber"
type="ns_p:ElementTagType"/>
                                <xs:element minOccurs="0" name="elapsedSlotTime"
type="ns_p:ElementTagType"/>
                                <xs:element minOccurs="0" name="remainingSlotTime"
type="ns_p:ElementTagType"/>
                                <xs:element name="sequenceRemoteControllable" minOccurs="0"
type="ns_p:ElementTagType"/>
                                <xs:element minOccurs="0" name="activeRepetitionNumber"
type="ns_p:ElementTagType"/>
                                <xs:element minOccurs="0" name="remainingPauseTime"
type="ns_p:ElementTagType"/>
```



```
                            </xs:sequence>
                        </xs:restriction>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
            <xs:element minOccurs="0" name="schedule">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:restriction base="ns_p:PowerSequenceScheduleDataElementsType">
                            <xs:sequence>
                                <xs:element name="startTime" minOccurs="0"
type="ns_p:ElementTagType"/>
                                <xs:element name="endTime" minOccurs="0"
type="ns_p:ElementTagType"/>
                            </xs:sequence>
                        </xs:restriction>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
            <xs:element name="scheduleConstraints" minOccurs="0">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:restriction
base="ns_p:PowerSequenceScheduleConstraintsDataElementsType">
                            <xs:sequence>
                                <xs:element name="earliestStartTime" minOccurs="0"
type="ns_p:ElementTagType"/>
                                <xs:element name="latestEndTime" minOccurs="0"
type="ns_p:ElementTagType"/>
                            </xs:sequence>
                        </xs:restriction>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
            <xs:element minOccurs="0" name="schedulePreference">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:restriction
base="ns_p:PowerSequenceSchedulePreferenceDataElementsType">
                            <xs:sequence>
                                <xs:element minOccurs="0" name="greenest"
type="ns_p:ElementTagType"/>
                                <xs:element minOccurs="0" name="cheapest"
type="ns_p:ElementTagType"/>
                            </xs:sequence>
                        </xs:restriction>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
            <xs:element name="operatingConstraintsInterrupt" minOccurs="0">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:restriction
base="ns_p:OperatingConstraintsInterruptDataElementsType">
                            <xs:sequence>
                                <xs:element minOccurs="0" name="isPausable"
type="ns_p:ElementTagType"/>
                                <xs:element minOccurs="0" name="isStoppable"
type="ns_p:ElementTagType"/>
                                <xs:element minOccurs="0" name="maxCyclesPerDay"
type="ns_p:ElementTagType"/>
                            </xs:sequence>
                        </xs:restriction>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
            <xs:element name="operatingConstraintsDuration" minOccurs="0">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:restriction
base="ns_p:OperatingConstraintsDurationDataElementsType">
                            <xs:sequence>
                                <xs:element minOccurs="0" name="activeDurationMin"
type="ns_p:ElementTagType"/>
                                <xs:element minOccurs="0" name="activeDurationMax"
type="ns_p:ElementTagType"/>
                                <xs:element minOccurs="0" name="pauseDurationMin"
type="ns_p:ElementTagType"/>
```



```
                                <xs:element minOccurs="0" name="pauseDurationMax"
type="ns_p:ElementTagType"/>
                                <xs:element minOccurs="0" name="activeDurationSumMin"
type="ns_p:ElementTagType"/>
                                <xs:element minOccurs="0" name="activeDurationSumMax"
type="ns_p:ElementTagType"/>
                            </xs:sequence>
                        </xs:restriction>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
            <xs:element name="operatingConstraintsResumeImplication" minOccurs="0">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:restriction
base="ns_p:OperatingConstraintsResumeImplicationDataElementsType">
                            <xs:sequence>
                                <xs:element minOccurs="0" name="resumeEnergyEstimated"
type="ns_p:ScaledNumberElementsType"/>
                                <xs:element minOccurs="0" name="energyUnit"
type="ns_p:ElementTagType"/>
                                <xs:element minOccurs="0" name="resumeCostEstimated"
type="ns_p:ScaledNumberElementsType"/>
                                <xs:element minOccurs="0" name="currency"
type="ns_p:ElementTagType"/>
                            </xs:sequence>
                        </xs:restriction>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
            <xs:element minOccurs="0" name="powerTimeSlot"
type="ns_p:SmartEnergyManagementPsPowerTimeSlotElementsType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="SmartEnergyManagementPsPowerTimeSlotElementsType">
        <xs:sequence>
            <xs:element name="schedule" minOccurs="0">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:restriction base="ns_p:PowerTimeSlotScheduleDataElementsType">
                            <xs:sequence>
                                <xs:element name="slotNumber" minOccurs="0"
type="ns_p:ElementTagType"/>
                                <xs:element name="timePeriod" minOccurs="0">
                                    <xs:complexType>
                                        <xs:complexContent>
                                            <xs:restriction
base="ns_p:TimePeriodElementsType">
                                                <xs:sequence>
                                                    <xs:element name="startTime" minOccurs="0"
type="ns_p:ElementTagType"/>
                                                    <xs:element name="endTime" minOccurs="0"
type="ns_p:ElementTagType"/>
                                                </xs:sequence>
                                            </xs:restriction>
                                        </xs:complexContent>
                                    </xs:complexType>
                                </xs:element>
                                <xs:element name="defaultDuration" minOccurs="0"
type="ns_p:ElementTagType"/>
                                <xs:element minOccurs="0" name="durationUncertainty"
type="ns_p:ElementTagType"/>
                                <xs:element name="slotActivated" minOccurs="0"
type="ns_p:ElementTagType"/>
                                <xs:element minOccurs="0" name="description"
type="ns_p:ElementTagType"/>
                            </xs:sequence>
                        </xs:restriction>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
            <xs:element name="valueList" minOccurs="0"
type="ns_p:SmartEnergyManagementPsPowerTimeSlotValueListElementsType"/>
            <xs:element name="scheduleConstraints" minOccurs="0">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:restriction
base="ns_p:PowerTimeSlotScheduleConstraintsDataElementsType">
```



```
                            <xs:sequence>
                                <xs:element name="earliestStartTime" minOccurs="0"
type="ns_p:ElementTagType"/>
                                <xs:element name="latestEndTime" minOccurs="0"
type="ns_p:ElementTagType"/>
                                <xs:element minOccurs="0" name="minDuration"
type="ns_p:ElementTagType"/>
                                <xs:element minOccurs="0" name="maxDuration"
type="ns_p:ElementTagType"/>
                                <xs:element minOccurs="0" name="optionalSlot"
type="ns_p:ElementTagType"/>
                            </xs:sequence>
                        </xs:restriction>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="SmartEnergyManagementPsPowerTimeSlotValueListElementsType">
        <xs:sequence>
            <xs:element name="value" minOccurs="0">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:restriction base="ns_p:PowerTimeSlotValueDataElementsType">
                            <xs:sequence>
                                <xs:element name="valueType" minOccurs="0"
type="ns_p:ElementTagType"/>
                                <xs:element name="value" minOccurs="0"
type="ns_p:ScaledNumberElementsType"/>
                            </xs:sequence>
                        </xs:restriction>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="SmartEnergyManagementPsDataElementsType">
        <xs:sequence>
            <xs:element name="nodeScheduleInformation" minOccurs="0">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:restriction
base="ns_p:PowerSequenceNodeScheduleInformationDataElementsType">
                            <xs:sequence>
                                <xs:element name="nodeRemoteControllable" minOccurs="0"
type="ns_p:ElementTagType"/>
                                <xs:element name="supportsSingleSlotSchedulingOnly"
minOccurs="0" type="ns_p:ElementTagType"/>
                                <xs:element name="alternativesCount" minOccurs="0"
type="ns_p:ElementTagType"/>
                                <xs:element name="totalSequencesCountMax" minOccurs="0"
type="ns_p:ElementTagType"/>
                                <xs:element name="supportsReselection" minOccurs="0"
type="ns_p:ElementTagType"/>
                            </xs:sequence>
                        </xs:restriction>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
            <xs:element name="alternatives" minOccurs="0"
type="ns_p:SmartEnergyManagementPsAlternativesElementsType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="smartEnergyManagementPsDataElements"
type="ns_p:SmartEnergyManagementPsDataElementsType"/>
    <xs:complexType name="SmartEnergyManagementPsDataSelectorsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="alternativesRelation">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:restriction
base="ns_p:PowerSequenceAlternativesRelationListDataSelectorsType">
                            <xs:sequence>
                                <xs:element minOccurs="0" name="alternativesId"
type="ns_p:AlternativesIdType"/>
                            </xs:sequence>
                        </xs:restriction>
                    </xs:complexContent>
```



```
                </xs:complexType>
            </xs:element>
            <xs:element minOccurs="0" name="powerSequenceDescription">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:restriction
base="ns_p:PowerSequenceDescriptionListDataSelectorsType">
                            <xs:sequence>
                                <xs:element minOccurs="0" name="sequenceId"
type="ns_p:PowerSequenceIdType"/>
                            </xs:sequence>
                        </xs:restriction>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
            <xs:element minOccurs="0" name="powerTimeSlotSchedule">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:restriction
base="ns_p:PowerTimeSlotScheduleListDataSelectorsType">
                            <xs:sequence>
                                <xs:element minOccurs="0" name="slotNumber"
type="ns_p:PowerTimeSlotNumberType"/>
                            </xs:sequence>
                        </xs:restriction>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
            <xs:element minOccurs="0" name="powerTimeSlotValue">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:restriction base="ns_p:PowerTimeSlotValueListDataSelectorsType">
                            <xs:sequence>
                                <xs:element minOccurs="0" name="valueType"
type="ns_p:PowerTimeSlotValueTypeType"/>
                            </xs:sequence>
                        </xs:restriction>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="smartEnergyManagementPsDataSelectors"
type="ns_p:SmartEnergyManagementPsDataSelectorsType"/>
    <xs:complexType name="SmartEnergyManagementPsPriceDataType">
        <xs:sequence>
            <xs:element maxOccurs="unbounded" minOccurs="0" name="price">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:restriction base="ns_p:PowerSequencePriceDataType">
                            <xs:sequence>
                                <xs:element minOccurs="0" name="sequenceId"
type="ns_p:PowerSequenceIdType"/>
                                <xs:element minOccurs="0" name="price"
type="ns_p:ScaledNumberType"/>
                                <xs:element minOccurs="0" name="currency"
type="ns_p:CurrencyType"/>
                            </xs:sequence>
                        </xs:restriction>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="smartEnergyManagementPsPriceData"
type="ns_p:SmartEnergyManagementPsPriceDataType"/>
    <xs:complexType name="SmartEnergyManagementPsPriceDataElementsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="price">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:restriction base="ns_p:PowerSequencePriceDataElementsType">
                            <xs:sequence>
                                <xs:element minOccurs="0" name="sequenceId"
type="ns_p:ElementTagType"/>
                                <xs:element minOccurs="0" name="price"
type="ns_p:ScaledNumberElementsType"/>
```



```
                                <xs:element minOccurs="0" name="currency"
type="ns_p:ElementTagType"/>
                            </xs:sequence>
                        </xs:restriction>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="smartEnergyManagementPsPriceDataElements"
type="ns_p:SmartEnergyManagementPsPriceDataElementsType"/>
    <xs:complexType name="SmartEnergyManagementPsPriceDataSelectorsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="price">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:restriction base="ns_p:PowerSequencePriceListDataSelectorsType">
                            <xs:sequence>
                                <xs:element minOccurs="0" name="sequenceId"
type="ns_p:PowerSequenceIdType"/>
                            </xs:sequence>
                        </xs:restriction>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="smartEnergyManagementPsPriceDataSelectors"
type="ns_p:SmartEnergyManagementPsPriceDataSelectorsType"/>
    <xs:complexType name="SmartEnergyManagementPsConfigurationRequestCallType">
        <xs:sequence>
            <xs:element minOccurs="0" name="scheduleConfigurationRequest">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:restriction
base="ns_p:PowerSequenceScheduleConfigurationRequestCallType">
                            <xs:sequence>
                                <xs:element minOccurs="0" name="sequenceId"
type="ns_p:PowerSequenceIdType"/>
                            </xs:sequence>
                        </xs:restriction>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="smartEnergyManagementPsConfigurationRequestCall"
type="ns_p:SmartEnergyManagementPsConfigurationRequestCallType"/>
    <xs:complexType name="SmartEnergyManagementPsConfigurationRequestCallElementsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="scheduleConfigurationRequest">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:restriction
base="ns_p:PowerSequenceScheduleConfigurationRequestCallElementsType">
                            <xs:sequence>
                                <xs:element minOccurs="0" name="sequenceId"
type="ns_p:ElementTagType"/>
                            </xs:sequence>
                        </xs:restriction>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="smartEnergyManagementPsConfigurationRequestCallElements"
type="ns_p:SmartEnergyManagementPsConfigurationRequestCallElementsType"/>
    <xs:complexType name="SmartEnergyManagementPsPriceCalculationRequestCallType">
        <xs:sequence>
            <xs:element minOccurs="0" name="priceCalculationRequest">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:restriction
base="ns_p:PowerSequencePriceCalculationRequestCallType">
                            <xs:sequence>
                                <xs:element minOccurs="0" name="sequenceId"
type="ns_p:PowerSequenceIdType"/>
                            </xs:sequence>
```



```
                        </xs:restriction>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="smartEnergyManagementPsPriceCalculationRequestCall"
type="ns_p:SmartEnergyManagementPsPriceCalculationRequestCallType"/>
    <xs:complexType name="SmartEnergyManagementPsPriceCalculationRequestCallElementsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="priceCalculationRequest">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:restriction
base="ns_p:PowerSequencePriceCalculationRequestCallElementsType">
                            <xs:sequence>
                                <xs:element minOccurs="0" name="sequenceId"
type="ns_p:ElementTagType"/>
                            </xs:sequence>
                        </xs:restriction>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="smartEnergyManagementPsPriceCalculationRequestCallElements"
type="ns_p:SmartEnergyManagementPsPriceCalculationRequestCallElementsType"/>
</xs:schema>
```

A.2 Standard Classes

A.2.1 ActuatorLevel

File name: EEBus_SPINE_TS_ActuatorLevel.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
    Smart Premises Interoperable Neutral-Message Exchange (SPINE)
    Version 1.1.1
    2018-12-21
    Copyright (c) 2018 EEBus Initiative e.V.  All Rights Reserved.
    Source: https://www.eebus.org/en/specifications/
-->
<xs:schema xmlns:ns_p="http://docs.eebus.org/spine/xsd/v1"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://docs.eebus.org/spine/xsd/v1" version="1.1.1"
    blockDefault="#all" elementFormDefault="qualified">
    <xs:annotation>
        <xs:documentation>EEBus SPINE Specification schema. Copyright 2018 EEBus Initiative
e.V.  All rights reserved.</xs:documentation>
    </xs:annotation>
    <xs:include schemaLocation="EEBus_SPINE_TS_CommonDataTypes.xsd"/>
    <xs:simpleType name="ActuatorLevelFctType">
        <xs:union memberTypes="ns_p:ActuatorLevelFctEnumType ns_p:EnumExtendType"/>
    </xs:simpleType>
    <xs:simpleType name="ActuatorLevelFctEnumType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="start"/>
            <xs:enumeration value="up"/>
            <xs:enumeration value="down"/>
            <xs:enumeration value="stop"/>
            <xs:enumeration value="percentageAbsolute"/>
            <xs:enumeration value="percentageRelative"/>
            <xs:enumeration value="absolute"/>
            <xs:enumeration value="relative"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:complexType name="ActuatorLevelDataType">
        <xs:sequence>
            <xs:element name="function" type="ns_p:ActuatorLevelFctType" minOccurs="0"/>
            <xs:element name="value" type="ns_p:ScaledNumberType" minOccurs="0"/>
```



```
        </xs:sequence>
    </xs:complexType>
    <xs:element name="actuatorLevelData" type="ns_p:ActuatorLevelDataType"/>
    <xs:complexType name="ActuatorLevelDataElementsType">
        <xs:sequence>
            <xs:element name="function" minOccurs="0" type="ns_p:ElementTagType"/>
            <xs:element name="value" type="ns_p:ScaledNumberElementsType" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="actuatorLevelDataElements" type="ns_p:ActuatorLevelDataElementsType"/>
    <xs:complexType name="ActuatorLevelDescriptionDataType">
        <xs:sequence>
            <xs:element minOccurs="0" name="label" type="ns_p:LabelType"/>
            <xs:element minOccurs="0" name="description" type="ns_p:DescriptionType"/>
            <xs:element minOccurs="0" name="levelDefaultUnit"
type="ns_p:UnitOfMeasurementType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="actuatorLevelDescriptionData"
type="ns_p:ActuatorLevelDescriptionDataType"/>
    <xs:complexType name="ActuatorLevelDescriptionDataElementsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="label" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="description" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="levelDefaultUnit" type="ns_p:ElementTagType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="actuatorLevelDescriptionDataElements"
type="ns_p:ActuatorLevelDescriptionDataElementsType"/>
</xs:schema>
```

A.2.2 ActuatorSwitch

File name: EEBus_SPINE_TS_ActuatorSwitch.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
    Smart Premises Interoperable Neutral-Message Exchange (SPINE)
    Version 1.1.1
    2018-12-21
    Copyright (c) 2018 EEBus Initiative e.V.  All Rights Reserved.
    Source: https://www.eebus.org/en/specifications/
-->
<xs:schema xmlns:ns_p="http://docs.eebus.org/spine/xsd/v1"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://docs.eebus.org/spine/xsd/v1" version="1.1.1"
    blockDefault="#all" elementFormDefault="qualified">
    <xs:annotation>
        <xs:documentation>EEBus SPINE Specification schema. Copyright 2018 EEBus Initiative
e.V.  All rights reserved.</xs:documentation>
    </xs:annotation>
    <xs:include schemaLocation="EEBus_SPINE_TS_CommonDataTypes.xsd"/>
    <xs:simpleType name="ActuatorSwitchFctType">
        <xs:union memberTypes="ns_p:ActuatorSwitchFctEnumType ns_p:EnumExtendType"/>
    </xs:simpleType>
    <xs:simpleType name="ActuatorSwitchFctEnumType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="on"/>
            <xs:enumeration value="off"/>
            <xs:enumeration value="toggle"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:complexType name="ActuatorSwitchDataType">
        <xs:sequence>
            <xs:element name="function" type="ns_p:ActuatorSwitchFctType" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="actuatorSwitchData" type="ns_p:ActuatorSwitchDataType"/>
    <xs:complexType name="ActuatorSwitchDataElementsType">
        <xs:sequence>
            <xs:element name="function" minOccurs="0" type="ns_p:ElementTagType"/>
        </xs:sequence>
```



```
    </xs:complexType>
    <xs:element name="actuatorSwitchDataElements" type="ns_p:ActuatorSwitchDataElementsType"/>
    <xs:complexType name="ActuatorSwitchDescriptionDataType">
        <xs:sequence>
            <xs:element minOccurs="0" name="label" type="ns_p:LabelType"/>
            <xs:element minOccurs="0" name="description" type="ns_p:DescriptionType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="actuatorSwitchDescriptionData"
type="ns_p:ActuatorSwitchDescriptionDataType"/>
    <xs:complexType name="ActuatorSwitchDescriptionDataElementsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="label" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="description" type="ns_p:ElementTagType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="actuatorSwitchDescriptionDataElements"
type="ns_p:ActuatorSwitchDescriptionDataElementsType"/>
</xs:schema>
```

A.2.3 Alarm

File name: EEBus_SPINE_TS_Alarm.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
    Smart Premises Interoperable Neutral-Message Exchange (SPINE)
    Version 1.1.1
    2018-12-21
    Copyright (c) 2018 EEBus Initiative e.V.  All Rights Reserved.
    Source: https://www.eebus.org/en/specifications/
-->
<xs:schema xmlns:ns_p="http://docs.eebus.org/spine/xsd/v1"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://docs.eebus.org/spine/xsd/v1" version="1.1.1"
    blockDefault="#all" elementFormDefault="qualified">
    <xs:annotation>
        <xs:documentation>EEBus SPINE Specification schema. Copyright 2018 EEBus Initiative
e.V.  All rights reserved.</xs:documentation>
    </xs:annotation>
    <xs:include schemaLocation="EEBus_SPINE_TS_CommonDataTypes.xsd"/>
    <xs:include schemaLocation="EEBus_SPINE_TS_Threshold.xsd"/>
    <xs:simpleType name="AlarmIdType">
        <xs:restriction base="xs:unsignedInt"/>
    </xs:simpleType>
    <xs:simpleType name="AlarmTypeType">
        <xs:union memberTypes="ns_p:AlarmTypeEnumType ns_p:EnumExtendType"/>
    </xs:simpleType>
    <xs:simpleType name="AlarmTypeEnumType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="alarmCancelled"/>
            <xs:enumeration value="underThreshold"/>
            <xs:enumeration value="overThreshold"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:complexType name="AlarmDataType">
        <xs:sequence>
            <xs:element name="alarmId" minOccurs="0" type="ns_p:AlarmIdType"/>
            <xs:element minOccurs="0" name="thresholdId" type="ns_p:ThresholdIdType"/>
            <xs:element minOccurs="0" name="timestamp"
type="ns_p:AbsoluteOrRelativeTimeType"/>
            <xs:element name="alarmType" minOccurs="0" type="ns_p:AlarmTypeType"/>
            <xs:element minOccurs="0" name="measuredValue" type="ns_p:ScaledNumberType"/>
            <xs:element minOccurs="0" name="evaluationPeriod" type="ns_p:TimePeriodType"/>
            <xs:element minOccurs="0" name="scopeType" type="ns_p:ScopeTypeType"/>
            <xs:element minOccurs="0" name="label" type="ns_p:LabelType"/>
            <xs:element minOccurs="0" name="description" type="ns_p:DescriptionType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="alarmData" type="ns_p:AlarmDataType"/>
    <xs:complexType name="AlarmDataElementsType">
        <xs:sequence>
```



```
            <xs:element name="alarmId" minOccurs="0" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="thresholdId" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="timestamp" type="ns_p:ElementTagType"/>
            <xs:element name="alarmType" minOccurs="0" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="measuredValue"
type="ns_p:ScaledNumberElementsType"/>
            <xs:element minOccurs="0" name="evaluationPeriod"
type="ns_p:TimePeriodElementsType"/>
            <xs:element minOccurs="0" name="scopeType" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="label" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="description" type="ns_p:ElementTagType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="alarmDataElements" type="ns_p:AlarmDataElementsType"/>
    <xs:complexType name="AlarmListDataType">
        <xs:sequence>
            <xs:element maxOccurs="unbounded" minOccurs="0" ref="ns_p:alarmData"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="alarmListData" type="ns_p:AlarmListDataType"/>
    <xs:complexType name="AlarmListDataSelectorsType">
        <xs:sequence>
            <xs:element name="alarmId" minOccurs="0" type="ns_p:AlarmIdType"/>
            <xs:element minOccurs="0" name="scopeType" type="ns_p:ScopeTypeType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="alarmListDataSelectors" type="ns_p:AlarmListDataSelectorsType"/>
</xs:schema>
```

A.2.4 Bill

File name: EEBus_SPINE_TS_Bill.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
    Smart Premises Interoperable Neutral-Message Exchange (SPINE)
    Version 1.1.1
    2018-12-21
    Copyright (c) 2018 EEBus Initiative e.V.  All Rights Reserved.
    Source: https://www.eebus.org/en/specifications/
-->
<xs:schema xmlns:ns_p="http://docs.eebus.org/spine/xsd/v1"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://docs.eebus.org/spine/xsd/v1" version="1.1.1"
    blockDefault="#all" elementFormDefault="qualified">
    <xs:annotation>
        <xs:documentation>EEBus SPINE Specification schema. Copyright 2018 EEBus Initiative
e.V.  All rights reserved.</xs:documentation>
    </xs:annotation>
    <xs:include schemaLocation="EEBus_SPINE_TS_CommonDataTypes.xsd"/>
    <xs:simpleType name="BillIdType">
        <xs:restriction base="xs:unsignedInt"/>
    </xs:simpleType>
    <xs:simpleType name="BillTypeType">
        <xs:union memberTypes="ns_p:BillTypeEnumType ns_p:EnumExtendType"/>
    </xs:simpleType>
    <xs:simpleType name="BillTypeEnumType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="chargingSummary"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="BillPositionIdType">
        <xs:restriction base="xs:unsignedInt"/>
    </xs:simpleType>
    <xs:simpleType name="BillPositionCountType">
        <xs:restriction base="ns_p:BillPositionIdType"/>
    </xs:simpleType>
    <xs:simpleType name="BillPositionTypeType">
        <xs:union memberTypes="ns_p:BillPositionTypeEnumType ns_p:EnumExtendType"/>
    </xs:simpleType>
    <xs:simpleType name="BillPositionTypeEnumType">
        <xs:restriction base="xs:string">
```

```
13857             <xs:enumeration value="gridElectricEnergy"/>
13858             <xs:enumeration value="selfProducedElectricEnergy"/>
13859         </xs:restriction>
13860     </xs:simpleType>
13861     <xs:simpleType name="BillValueIdType">
13862         <xs:restriction base="xs:unsignedInt"/>
13863     </xs:simpleType>
13864     <xs:simpleType name="BillCostIdType">
13865         <xs:restriction base="xs:unsignedInt"/>
13866     </xs:simpleType>
13867     <xs:simpleType name="BillCostTypeType">
13868         <xs:union memberTypes="ns_p:BillCostTypeEnumType ns_p:EnumExtendType"/>
13869     </xs:simpleType>
13870     <xs:simpleType name="BillCostTypeEnumType">
13871         <xs:restriction base="xs:string">
13872             <xs:enumeration value="absolutePrice"/>
13873             <xs:enumeration value="relativePrice"/>
13874             <xs:enumeration value="co2Emission"/>
13875             <xs:enumeration value="renewableEnergy"/>
13876             <xs:enumeration value="radioactiveWaste"/>
13877         </xs:restriction>
13878     </xs:simpleType>
13879     <xs:complexType name="BillValueType">
13880         <xs:sequence>
13881             <xs:element minOccurs="0" name="valueId" type="ns_p:BillValueIdType"/>
13882             <xs:element minOccurs="0" name="unit" type="ns_p:UnitOfMeasurementType"/>
13883             <xs:element minOccurs="0" name="value" type="ns_p:ScaledNumberType"/>
13884             <xs:element minOccurs="0" name="valuePercentage" type="ns_p:ScaledNumberType"/>
13885         </xs:sequence>
13886     </xs:complexType>
13887     <xs:complexType name="BillValueElementsType">
13888         <xs:sequence>
13889             <xs:element minOccurs="0" name="valueId" type="ns_p:ElementTagType"/>
13890             <xs:element minOccurs="0" name="unit" type="ns_p:ElementTagType"/>
13891             <xs:element minOccurs="0" name="value" type="ns_p:ScaledNumberElementsType"/>
13892             <xs:element minOccurs="0" name="valuePercentage"
13893 type="ns_p:ScaledNumberElementsType"/>
13894         </xs:sequence>
13895     </xs:complexType>
13896     <xs:complexType name="BillCostType">
13897         <xs:sequence>
13898             <xs:element minOccurs="0" name="costId" type="ns_p:BillCostIdType"/>
13899             <xs:element minOccurs="0" name="costType" type="ns_p:BillCostTypeType"/>
13900             <xs:element minOccurs="0" name="valueId" type="ns_p:BillValueIdType"/>
13901             <xs:element minOccurs="0" name="unit" type="ns_p:UnitOfMeasurementType"/>
13902             <xs:element minOccurs="0" name="currency" type="ns_p:CurrencyType"/>
13903             <xs:element minOccurs="0" name="cost" type="ns_p:ScaledNumberType"/>
13904             <xs:element minOccurs="0" name="costPercentage" type="ns_p:ScaledNumberType"/>
13905         </xs:sequence>
13906     </xs:complexType>
13907     <xs:complexType name="BillCostElementsType">
13908         <xs:sequence>
13909             <xs:element minOccurs="0" name="costId" type="ns_p:ElementTagType"/>
13910             <xs:element minOccurs="0" name="costType" type="ns_p:ElementTagType"/>
13911             <xs:element minOccurs="0" name="valueId" type="ns_p:ElementTagType"/>
13912             <xs:element minOccurs="0" name="unit" type="ns_p:ElementTagType"/>
13913             <xs:element minOccurs="0" name="currency" type="ns_p:ElementTagType"/>
13914             <xs:element minOccurs="0" name="cost" type="ns_p:ScaledNumberElementsType"/>
13915             <xs:element minOccurs="0" name="costPercentage"
13916 type="ns_p:ScaledNumberElementsType"/>
13917         </xs:sequence>
13918     </xs:complexType>
13919     <xs:complexType name="BillPositionType">
13920         <xs:sequence>
13921             <xs:element minOccurs="0" name="positionId" type="ns_p:BillPositionIdType"/>
13922             <xs:element minOccurs="0" name="positionType" type="ns_p:BillPositionTypeType"/>
13923             <xs:element minOccurs="0" name="timePeriod" type="ns_p:TimePeriodType"/>
13924             <xs:element maxOccurs="unbounded" minOccurs="0" name="value"
13925 type="ns_p:BillValueType"/>
13926             <xs:element minOccurs="0" name="cost" maxOccurs="unbounded"
13927 type="ns_p:BillCostType"/>
13928             <xs:element minOccurs="0" name="label" type="ns_p:LabelType"/>
13929             <xs:element minOccurs="0" name="description" type="ns_p:DescriptionType"/>
13930         </xs:sequence>
13931     </xs:complexType>
13932     <xs:complexType name="BillPositionElementsType">
13933         <xs:sequence>
13934             <xs:element minOccurs="0" name="positionId" type="ns_p:ElementTagType"/>
```



```
            <xs:element minOccurs="0" name="positionType" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="timePeriod" type="ns_p:TimePeriodElementsType"/>
            <xs:element minOccurs="0" name="value" type="ns_p:BillValueElementsType"/>
            <xs:element minOccurs="0" name="cost" type="ns_p:BillCostElementsType"/>
            <xs:element minOccurs="0" name="label" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="description" type="ns_p:ElementTagType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="BillDataType">
        <xs:sequence>
            <xs:element minOccurs="0" name="billId" type="ns_p:BillIdType"/>
            <xs:element minOccurs="0" name="billType" type="ns_p:BillTypeType"/>
            <xs:element minOccurs="0" name="scopeType" type="ns_p:ScopeTypeType"/>
            <xs:element minOccurs="0" name="total">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:restriction base="ns_p:BillPositionType">
                            <xs:sequence>
                                <xs:element minOccurs="0" name="timePeriod"
type="ns_p:TimePeriodType"/>
                                <xs:element maxOccurs="unbounded" minOccurs="0" name="value">
                                    <xs:complexType>
                                        <xs:complexContent>
                                            <xs:restriction base="ns_p:BillValueType">
                                                <xs:sequence>
                                                    <xs:element minOccurs="0" name="valueId"
type="ns_p:BillValueIdType"/>
                                                    <xs:element minOccurs="0" name="unit"
type="ns_p:UnitOfMeasurementType"/>
                                                    <xs:element minOccurs="0" name="value"
type="ns_p:ScaledNumberType"/>
                                                </xs:sequence>
                                            </xs:restriction>
                                        </xs:complexContent>
                                    </xs:complexType>
                                </xs:element>
                                <xs:element minOccurs="0" name="cost" maxOccurs="unbounded">
                                    <xs:complexType>
                                        <xs:complexContent>
                                            <xs:restriction base="ns_p:BillCostType">
                                                <xs:sequence>
                                                    <xs:element minOccurs="0" name="costId"
type="ns_p:BillCostIdType"/>
                                                    <xs:element minOccurs="0" name="costType"
type="ns_p:BillCostTypeType"/>
                                                    <xs:element minOccurs="0" name="valueId"
type="ns_p:BillValueIdType"/>
                                                    <xs:element minOccurs="0" name="unit"
type="ns_p:UnitOfMeasurementType"/>
                                                    <xs:element minOccurs="0" name="currency"
type="ns_p:CurrencyType"/>
                                                    <xs:element minOccurs="0" name="cost"
type="ns_p:ScaledNumberType"/>
                                                </xs:sequence>
                                            </xs:restriction>
                                        </xs:complexContent>
                                    </xs:complexType>
                                </xs:element>
                                <xs:element minOccurs="0" name="label" type="ns_p:LabelType"/>
                                <xs:element minOccurs="0" name="description"
type="ns_p:DescriptionType"/>
                            </xs:sequence>
                        </xs:restriction>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
            <xs:element maxOccurs="unbounded" minOccurs="0" name="position">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:restriction base="ns_p:BillPositionType">
                            <xs:sequence>
                                <xs:element minOccurs="0" name="positionId"
type="ns_p:BillPositionIdType"/>
                                <xs:element minOccurs="0" name="positionType"
type="ns_p:BillPositionTypeType"/>
                                <xs:element minOccurs="0" name="timePeriod"
type="ns_p:TimePeriodType"/>
                                <xs:element maxOccurs="unbounded" minOccurs="0" name="value">
```



```
                                    <xs:complexType>
                                        <xs:complexContent>
                                            <xs:restriction base="ns_p:BillValueType">
                                                <xs:sequence>
                                                    <xs:element minOccurs="0" name="valueId"
type="ns_p:BillValueIdType"/>
                                                    <xs:element minOccurs="0" name="value"
type="ns_p:ScaledNumberType"/>
                                                    <xs:element minOccurs="0"
name="valuePercentage" type="ns_p:ScaledNumberType"/>
                                                </xs:sequence>
                                            </xs:restriction>
                                        </xs:complexContent>
                                    </xs:complexType>
                                </xs:element>
                                <xs:element minOccurs="0" name="cost" maxOccurs="unbounded">
                                    <xs:complexType>
                                        <xs:complexContent>
                                            <xs:restriction base="ns_p:BillCostType">
                                                <xs:sequence>
                                                    <xs:element minOccurs="0" name="costId"
type="ns_p:BillCostIdType"/>
                                                    <xs:element minOccurs="0" name="cost"
type="ns_p:ScaledNumberType"/>
                                                    <xs:element minOccurs="0"
name="costPercentage" type="ns_p:ScaledNumberType"/>
                                                </xs:sequence>
                                            </xs:restriction>
                                        </xs:complexContent>
                                    </xs:complexType>
                                </xs:element>
                                <xs:element minOccurs="0" name="label" type="ns_p:LabelType"/>
                                <xs:element minOccurs="0" name="description"
type="ns_p:DescriptionType"/>
                            </xs:sequence>
                        </xs:restriction>
                    </xs:complexContent>
                </xs:complexType>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="billData" type="ns_p:BillDataType"/>
    <xs:complexType name="BillDataElementsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="billId" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="billType" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="scopeType" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="total">
                <xs:complexType>
                    <xs:complexContent>
                        <xs:restriction base="ns_p:BillPositionElementsType">
                            <xs:sequence>
                                <xs:element minOccurs="0" name="timePeriod"
type="ns_p:TimePeriodElementsType"/>
                                <xs:element minOccurs="0" name="value">
                                    <xs:complexType>
                                        <xs:complexContent>
                                            <xs:restriction base="ns_p:BillValueElementsType">
                                                <xs:sequence>
                                                    <xs:element minOccurs="0" name="valueId"
type="ns_p:ElementTagType"/>
                                                    <xs:element minOccurs="0" name="unit"
type="ns_p:ElementTagType"/>
                                                    <xs:element minOccurs="0" name="value"
type="ns_p:ScaledNumberElementsType"/>
                                                </xs:sequence>
                                            </xs:restriction>
                                        </xs:complexContent>
                                    </xs:complexType>
                                </xs:element>
                                <xs:element minOccurs="0" name="cost">
                                    <xs:complexType>
                                        <xs:complexContent>
                                            <xs:restriction base="ns_p:BillCostElementsType">
                                                <xs:sequence>
                                                    <xs:element minOccurs="0" name="costId"
type="ns_p:ElementTagType"/>
```



```
14090                                                   <xs:element minOccurs="0" name="costType"
14091   type="ns_p:ElementTagType"/>
14092                                                   <xs:element minOccurs="0" name="valueId"
14093   type="ns_p:ElementTagType"/>
14094                                                   <xs:element minOccurs="0" name="unit"
14095   type="ns_p:ElementTagType"/>
14096                                                   <xs:element minOccurs="0" name="currency"
14097   type="ns_p:ElementTagType"/>
14098                                                   <xs:element minOccurs="0" name="cost"
14099   type="ns_p:ScaledNumberElementsType"/>
14100                                               </xs:sequence>
14101                                           </xs:restriction>
14102                                       </xs:complexContent>
14103                                   </xs:complexType>
14104                               </xs:element>
14105                               <xs:element minOccurs="0" name="label"
14106   type="ns_p:ElementTagType"/>
14107                               <xs:element minOccurs="0" name="description"
14108   type="ns_p:ElementTagType"/>
14109                           </xs:sequence>
14110                       </xs:restriction>
14111                   </xs:complexContent>
14112               </xs:complexType>
14113           </xs:element>
14114           <xs:element minOccurs="0" name="position">
14115               <xs:complexType>
14116                   <xs:complexContent>
14117                       <xs:restriction base="ns_p:BillPositionElementsType">
14118                           <xs:sequence>
14119                               <xs:element minOccurs="0" name="positionId"
14120   type="ns_p:ElementTagType"/>
14121                               <xs:element minOccurs="0" name="positionType"
14122   type="ns_p:ElementTagType"/>
14123                               <xs:element minOccurs="0" name="timePeriod"
14124   type="ns_p:TimePeriodElementsType"/>
14125                               <xs:element minOccurs="0" name="value">
14126                                   <xs:complexType>
14127                                       <xs:complexContent>
14128                                           <xs:restriction base="ns_p:BillValueElementsType">
14129                                               <xs:sequence>
14130                                                   <xs:element minOccurs="0" name="valueId"
14131   type="ns_p:ElementTagType"/>
14132                                                   <xs:element minOccurs="0" name="value"
14133   type="ns_p:ScaledNumberElementsType"/>
14134                                                   <xs:element minOccurs="0"
14135   name="valuePercentage" type="ns_p:ScaledNumberElementsType"/>
14136                                               </xs:sequence>
14137                                           </xs:restriction>
14138                                       </xs:complexContent>
14139                                   </xs:complexType>
14140                               </xs:element>
14141                               <xs:element minOccurs="0" name="cost">
14142                                   <xs:complexType>
14143                                       <xs:complexContent>
14144                                           <xs:restriction base="ns_p:BillCostElementsType">
14145                                               <xs:sequence>
14146                                                   <xs:element minOccurs="0" name="costId"
14147   type="ns_p:ElementTagType"/>
14148                                                   <xs:element minOccurs="0" name="cost"
14149   type="ns_p:ScaledNumberElementsType"/>
14150                                                   <xs:element minOccurs="0"
14151   name="costPercentage" type="ns_p:ScaledNumberElementsType"/>
14152                                               </xs:sequence>
14153                                           </xs:restriction>
14154                                       </xs:complexContent>
14155                                   </xs:complexType>
14156                               </xs:element>
14157                               <xs:element minOccurs="0" name="label"
14158   type="ns_p:ElementTagType"/>
14159                               <xs:element minOccurs="0" name="description"
14160   type="ns_p:ElementTagType"/>
14161                           </xs:sequence>
14162                       </xs:restriction>
14163                   </xs:complexContent>
14164               </xs:complexType>
14165           </xs:element>
14166       </xs:sequence>
14167   </xs:complexType>
```



```
14168       <xs:element name="billDataElements" type="ns_p:BillDataElementsType"/>
14169       <xs:complexType name="BillListDataType">
14170           <xs:sequence>
14171               <xs:element maxOccurs="unbounded" minOccurs="0" ref="ns_p:billData"/>
14172           </xs:sequence>
14173       </xs:complexType>
14174       <xs:element name="billListData" type="ns_p:BillListDataType"/>
14175       <xs:complexType name="BillListDataSelectorsType">
14176           <xs:sequence>
14177               <xs:element minOccurs="0" name="billId" type="ns_p:BillIdType"/>
14178               <xs:element minOccurs="0" name="scopeType" type="ns_p:ScopeTypeType"/>
14179           </xs:sequence>
14180       </xs:complexType>
14181       <xs:element name="billListDataSelectors" type="ns_p:BillListDataSelectorsType"/>
14182       <xs:complexType name="BillConstraintsDataType">
14183           <xs:sequence>
14184               <xs:element minOccurs="0" name="billId" type="ns_p:BillIdType"/>
14185               <xs:element minOccurs="0" name="positionCountMin"
14186   type="ns_p:BillPositionCountType"/>
14187               <xs:element minOccurs="0" name="positionCountMax"
14188   type="ns_p:BillPositionCountType"/>
14189           </xs:sequence>
14190       </xs:complexType>
14191       <xs:element name="billConstraintsData" type="ns_p:BillConstraintsDataType"/>
14192       <xs:complexType name="BillConstraintsDataElementsType">
14193           <xs:sequence>
14194               <xs:element minOccurs="0" name="billId" type="ns_p:ElementTagType"/>
14195               <xs:element minOccurs="0" name="positionCountMin" type="ns_p:ElementTagType"/>
14196               <xs:element minOccurs="0" name="positionCountMax" type="ns_p:ElementTagType"/>
14197           </xs:sequence>
14198       </xs:complexType>
14199       <xs:element name="billConstraintsDataElements"
14200   type="ns_p:BillConstraintsDataElementsType"/>
14201       <xs:complexType name="BillConstraintsListDataType">
14202           <xs:sequence>
14203               <xs:element maxOccurs="unbounded" minOccurs="0" ref="ns_p:billConstraintsData"/>
14204           </xs:sequence>
14205       </xs:complexType>
14206       <xs:element name="billConstraintsListData" type="ns_p:BillConstraintsListDataType"/>
14207       <xs:complexType name="BillConstraintsListDataSelectorsType">
14208           <xs:sequence>
14209               <xs:element minOccurs="0" name="billId" type="ns_p:BillIdType"/>
14210           </xs:sequence>
14211       </xs:complexType>
14212       <xs:element name="billConstraintsListDataSelectors"
14213   type="ns_p:BillConstraintsListDataSelectorsType"/>
14214       <xs:complexType name="BillDescriptionDataType">
14215           <xs:sequence>
14216               <xs:element minOccurs="0" name="billId" type="ns_p:BillIdType"/>
14217               <xs:element minOccurs="0" name="billWriteable" type="xs:boolean"/>
14218               <xs:element minOccurs="0" name="updateRequired" type="xs:boolean"/>
14219               <xs:element maxOccurs="unbounded" minOccurs="0" name="supportedBillType"
14220   type="ns_p:BillTypeType"/>
14221           </xs:sequence>
14222       </xs:complexType>
14223       <xs:element name="billDescriptionData" type="ns_p:BillDescriptionDataType"/>
14224       <xs:complexType name="BillDescriptionDataElementsType">
14225           <xs:sequence>
14226               <xs:element minOccurs="0" name="billId" type="ns_p:ElementTagType"/>
14227               <xs:element minOccurs="0" name="billWriteable" type="ns_p:ElementTagType"/>
14228               <xs:element minOccurs="0" name="updateRequired" type="ns_p:ElementTagType"/>
14229               <xs:element minOccurs="0" name="supportedBillType" type="ns_p:ElementTagType"/>
14230           </xs:sequence>
14231       </xs:complexType>
14232       <xs:element name="billDescriptionDataElements"
14233   type="ns_p:BillDescriptionDataElementsType"/>
14234       <xs:complexType name="BillDescriptionListDataType">
14235           <xs:sequence>
14236               <xs:element maxOccurs="unbounded" minOccurs="0" ref="ns_p:billDescriptionData"/>
14237           </xs:sequence>
14238       </xs:complexType>
14239       <xs:element name="billDescriptionListData" type="ns_p:BillDescriptionListDataType"/>
14240       <xs:complexType name="BillDescriptionListDataSelectorsType">
14241           <xs:sequence>
14242               <xs:element minOccurs="0" name="billId" type="ns_p:BillIdType"/>
14243           </xs:sequence>
14244       </xs:complexType>
```



```
    <xs:element name="billDescriptionListDataSelectors"
type="ns_p:BillDescriptionListDataSelectorsType"/>
</xs:schema>
```

A.2.5 BindingManagement

File name: EEBus_SPINE_TS_BindingManagement.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
    Smart Premises Interoperable Neutral-Message Exchange (SPINE)
    Version 1.1.1
    2018-12-21
    Copyright (c) 2018 EEBus Initiative e.V.  All Rights Reserved.
    Source: https://www.eebus.org/en/specifications/
-->
<xs:schema xmlns:ns_p="http://docs.eebus.org/spine/xsd/v1"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://docs.eebus.org/spine/xsd/v1" version="1.1.1"
    blockDefault="#all" elementFormDefault="qualified">
    <xs:annotation>
        <xs:documentation>EEBus SPINE Specification schema. Copyright 2018 EEBus Initiative
e.V.  All rights reserved.</xs:documentation>
    </xs:annotation>
    <xs:include schemaLocation="EEBus_SPINE_TS_CommonDataTypes.xsd"/>
    <xs:simpleType name="BindingIdType">
        <xs:restriction base="xs:unsignedInt"/>
    </xs:simpleType>
    <xs:complexType name="BindingManagementEntryDataType">
        <xs:sequence>
            <xs:element minOccurs="0" name="bindingId" type="ns_p:BindingIdType"/>
            <xs:element minOccurs="0" name="clientAddress" type="ns_p:FeatureAddressType"/>
            <xs:element minOccurs="0" name="serverAddress" type="ns_p:FeatureAddressType"/>
            <xs:element minOccurs="0" name="label" type="ns_p:LabelType"/>
            <xs:element minOccurs="0" name="description" type="ns_p:DescriptionType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="bindingManagementEntryData" type="ns_p:BindingManagementEntryDataType"/>
    <xs:complexType name="BindingManagementEntryDataElementsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="bindingId" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="clientAddress"
type="ns_p:FeatureAddressElementsType"/>
            <xs:element minOccurs="0" name="serverAddress"
type="ns_p:FeatureAddressElementsType"/>
            <xs:element minOccurs="0" name="label" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="description" type="ns_p:ElementTagType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="bindingManagementEntryDataElements"
type="ns_p:BindingManagementEntryDataElementsType"/>
    <xs:complexType name="BindingManagementEntryListDataType">
        <xs:sequence>
            <xs:element maxOccurs="unbounded" minOccurs="0"
ref="ns_p:bindingManagementEntryData"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="bindingManagementEntryListData"
type="ns_p:BindingManagementEntryListDataType"/>
    <xs:complexType name="BindingManagementEntryListDataSelectorsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="bindingId" type="ns_p:BindingIdType"/>
            <xs:element minOccurs="0" name="clientAddress" type="ns_p:FeatureAddressType"/>
            <xs:element minOccurs="0" name="serverAddress" type="ns_p:FeatureAddressType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="bindingManagementEntryListDataSelectors"
type="ns_p:BindingManagementEntryListDataSelectorsType"/>
    <xs:complexType name="BindingManagementRequestCallType">
        <xs:sequence>
            <xs:element minOccurs="0" name="clientAddress" type="ns_p:FeatureAddressType"/>
            <xs:element minOccurs="0" name="serverAddress" type="ns_p:FeatureAddressType"/>
```

```
            <xs:element minOccurs="0" name="serverFeatureType" type="ns_p:FeatureTypeType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="bindingManagementRequestCall"
type="ns_p:BindingManagementRequestCallType"/>
    <xs:complexType name="BindingManagementRequestCallElementsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="clientAddress"
type="ns_p:FeatureAddressElementsType"/>
            <xs:element minOccurs="0" name="serverAddress"
type="ns_p:FeatureAddressElementsType"/>
            <xs:element minOccurs="0" name="serverFeatureType" type="ns_p:ElementTagType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="bindingManagementRequestCallElements"
type="ns_p:BindingManagementRequestCallElementsType"/>
    <xs:complexType name="BindingManagementDeleteCallType">
        <xs:sequence>
            <xs:element minOccurs="0" name="bindingId" type="ns_p:BindingIdType"/>
            <xs:element minOccurs="0" name="clientAddress" type="ns_p:FeatureAddressType"/>
            <xs:element minOccurs="0" name="serverAddress" type="ns_p:FeatureAddressType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="bindingManagementDeleteCall"
type="ns_p:BindingManagementDeleteCallType"/>
    <xs:complexType name="BindingManagementDeleteCallElementsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="bindingId" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="clientAddress"
type="ns_p:FeatureAddressElementsType"/>
            <xs:element minOccurs="0" name="serverAddress"
type="ns_p:FeatureAddressElementsType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="bindingManagementDeleteCallElements"
type="ns_p:BindingManagementDeleteCallElementsType"/>
</xs:schema>
```

A.2.6 DataTunneling

File name: EEBus_SPINE_TS_DataTunneling.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
    Smart Premises Interoperable Neutral-Message Exchange (SPINE)
    Version 1.1.1
    2018-12-21
    Copyright (c) 2018 EEBus Initiative e.V.  All Rights Reserved.
    Source: https://www.eebus.org/en/specifications/
-->
<xs:schema xmlns:ns_p="http://docs.eebus.org/spine/xsd/v1"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://docs.eebus.org/spine/xsd/v1" version="1.1.1"
    blockDefault="#all" elementFormDefault="qualified">
    <xs:annotation>
        <xs:documentation>EEBus SPINE Specification schema. Copyright 2018 EEBus Initiative
e.V.  All rights reserved.</xs:documentation>
    </xs:annotation>
    <xs:include schemaLocation="EEBus_SPINE_TS_CommonDataTypes.xsd"/>
    <xs:simpleType name="PurposeIdType">
        <xs:restriction base="xs:string"/>
    </xs:simpleType>
    <xs:simpleType name="ChannelIdType">
        <xs:restriction base="xs:unsignedInt"/>
    </xs:simpleType>
    <xs:complexType name="DataTunnelingHeaderType">
        <xs:sequence>
            <xs:element name="purposeId" type="ns_p:PurposeIdType" minOccurs="0"/>
            <xs:element name="channelId" type="ns_p:ChannelIdType" minOccurs="0"/>
            <xs:element name="sequenceId" type="xs:unsignedInt" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>
```



```
    <xs:complexType name="DataTunnelingHeaderElementsType">
        <xs:sequence>
            <xs:element name="purposeId" minOccurs="0" type="ns_p:ElementTagType"/>
            <xs:element name="channelId" minOccurs="0" type="ns_p:ElementTagType"/>
            <xs:element name="sequenceId" minOccurs="0" type="ns_p:ElementTagType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="DataTunnelingCallType">
        <xs:sequence>
            <xs:element name="header" minOccurs="0" type="ns_p:DataTunnelingHeaderType"/>
            <xs:element name="payload" type="xs:hexBinary" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="dataTunnelingCall" type="ns_p:DataTunnelingCallType"/>
    <xs:complexType name="DataTunnelingCallElementsType">
        <xs:sequence>
            <xs:element name="header" minOccurs="0"
type="ns_p:DataTunnelingHeaderElementsType"/>
            <xs:element name="payload" minOccurs="0" type="ns_p:ElementTagType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="dataTunnelingCallElements" type="ns_p:DataTunnelingCallElementsType"/>
</xs:schema>
```

A.2.7 DeviceClassification

File name: EEBus_SPINE_TS_DeviceClassification.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
    Smart Premises Interoperable Neutral-Message Exchange (SPINE)
    Version 1.1.1
    2018-12-21
    Copyright (c) 2018 EEBus Initiative e.V.  All Rights Reserved.
    Source: https://www.eebus.org/en/specifications/
-->
<xs:schema xmlns:ns_p="http://docs.eebus.org/spine/xsd/v1"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://docs.eebus.org/spine/xsd/v1" version="1.1.1"
    blockDefault="#all" elementFormDefault="qualified">
    <xs:annotation>
        <xs:documentation>EEBus SPINE Specification schema. Copyright 2018 EEBus Initiative
e.V.  All rights reserved.</xs:documentation>
    </xs:annotation>
    <xs:include schemaLocation="EEBus_SPINE_TS_CommonDataTypes.xsd"/>
    <xs:simpleType name="DeviceClassificationStringType">
        <xs:restriction base="xs:string"/>
    </xs:simpleType>
    <xs:simpleType name="PowerSourceType">
        <xs:union memberTypes="ns_p:PowerSourceEnumType ns_p:EnumExtendType"/>
    </xs:simpleType>
    <xs:simpleType name="PowerSourceEnumType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="unknown"/>
            <xs:enumeration value="mainsSinglePhase"/>
            <xs:enumeration value="mains3Phase"/>
            <xs:enumeration value="battery"/>
            <xs:enumeration value="dc"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:complexType name="DeviceClassificationManufacturerDataType">
        <xs:sequence>
            <xs:element minOccurs="0" name="deviceName"
type="ns_p:DeviceClassificationStringType"/>
            <xs:element minOccurs="0" name="deviceCode"
type="ns_p:DeviceClassificationStringType"/>
            <xs:element minOccurs="0" name="serialNumber"
type="ns_p:DeviceClassificationStringType"/>
            <xs:element minOccurs="0" name="softwareRevision"
type="ns_p:DeviceClassificationStringType"/>
            <xs:element minOccurs="0" name="hardwareRevision"
type="ns_p:DeviceClassificationStringType"/>
```



```
            <xs:element minOccurs="0" name="vendorName"
type="ns_p:DeviceClassificationStringType"/>
            <xs:element minOccurs="0" name="vendorCode"
type="ns_p:DeviceClassificationStringType"/>
            <xs:element minOccurs="0" name="brandName"
type="ns_p:DeviceClassificationStringType"/>
            <xs:element minOccurs="0" name="powerSource" type="ns_p:PowerSourceType"/>
            <xs:element minOccurs="0" name="manufacturerNodeIdentification"
type="ns_p:DeviceClassificationStringType"/>
            <xs:element minOccurs="0" name="manufacturerLabel" type="ns_p:LabelType"/>
            <xs:element minOccurs="0" name="manufacturerDescription"
type="ns_p:DescriptionType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="deviceClassificationManufacturerData"
type="ns_p:DeviceClassificationManufacturerDataType"/>
    <xs:complexType name="DeviceClassificationManufacturerDataElementsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="deviceName" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="deviceCode" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="serialNumber" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="softwareRevision" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="hardwareRevision" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="vendorName" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="vendorCode" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="brandName" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="powerSource" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="manufacturerNodeIdentification"
type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="manufacturerLabel" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="manufacturerDescription"
type="ns_p:ElementTagType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="deviceClassificationManufacturerDataElements"
type="ns_p:DeviceClassificationManufacturerDataElementsType"/>
    <xs:complexType name="DeviceClassificationUserDataType">
        <xs:sequence>
            <xs:element minOccurs="0" name="userNodeIdentification"
type="ns_p:DeviceClassificationStringType"/>
            <xs:element minOccurs="0" name="userLabel" type="ns_p:LabelType"/>
            <xs:element minOccurs="0" name="userDescription" type="ns_p:DescriptionType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="deviceClassificationUserData"
type="ns_p:DeviceClassificationUserDataType"/>
    <xs:complexType name="DeviceClassificationUserDataElementsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="userNodeIdentification"
type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="userLabel" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="userDescription" type="ns_p:ElementTagType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="deviceClassificationUserDataElements"
type="ns_p:DeviceClassificationUserDataElementsType"/>
</xs:schema>
```

A.2.8 DeviceConfiguration

File name: EEBus_SPINE_TS_DeviceConfiguration.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
    Smart Premises Interoperable Neutral-Message Exchange (SPINE)
    Version 1.1.1
    2018-12-21
    Copyright (c) 2018 EEBus Initiative e.V.  All Rights Reserved.
    Source: https://www.eebus.org/en/specifications/
-->
```



```
<xs:schema xmlns:ns_p="http://docs.eebus.org/spine/xsd/v1"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://docs.eebus.org/spine/xsd/v1" version="1.1.1"
    blockDefault="#all" elementFormDefault="qualified">
    <xs:annotation>
        <xs:documentation>EEBus SPINE Specification schema. Copyright 2018 EEBus Initiative
e.V.  All rights reserved.</xs:documentation>
    </xs:annotation>
    <xs:include schemaLocation="EEBus_SPINE_TS_CommonDataTypes.xsd"/>
    <xs:simpleType name="DeviceConfigurationKeyIdType">
        <xs:restriction base="xs:unsignedInt"/>
    </xs:simpleType>
    <xs:simpleType name="DeviceConfigurationKeyValueStringType">
        <xs:restriction base="xs:string"/>
    </xs:simpleType>
    <xs:simpleType name="DeviceConfigurationKeyNameType">
        <xs:union memberTypes="ns_p:DeviceConfigurationKeyNameEnumType ns_p:EnumExtendType"/>
    </xs:simpleType>
    <xs:simpleType name="DeviceConfigurationKeyNameEnumType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="peakPowerOfPvSystem"/>
            <xs:enumeration value="pvCurtailmentLimitFactor"/>
            <xs:enumeration value="asymmetricChargingSupported"/>
            <xs:enumeration value="communicationsStandard"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="DeviceConfigurationKeyValueTypeType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="boolean"/>
            <xs:enumeration value="date"/>
            <xs:enumeration value="dateTime"/>
            <xs:enumeration value="duration"/>
            <xs:enumeration value="string"/>
            <xs:enumeration value="time"/>
            <xs:enumeration value="scaledNumber"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:complexType name="DeviceConfigurationKeyValueValueType">
        <xs:sequence>
            <xs:element minOccurs="0" name="boolean" type="xs:boolean"/>
            <xs:element minOccurs="0" name="date" type="xs:date"/>
            <xs:element minOccurs="0" name="dateTime" type="xs:dateTime"/>
            <xs:element minOccurs="0" name="duration" type="xs:duration"/>
            <xs:element minOccurs="0" name="string"
type="ns_p:DeviceConfigurationKeyValueStringType"/>
            <xs:element minOccurs="0" name="time" type="xs:time"/>
            <xs:element minOccurs="0" name="scaledNumber" type="ns_p:ScaledNumberType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="DeviceConfigurationKeyValueValueElementsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="boolean" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="date" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="dateTime" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="duration" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="string" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="time" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="scaledNumber"
type="ns_p:ScaledNumberElementsType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="DeviceConfigurationKeyValueDataType">
        <xs:sequence>
            <xs:element minOccurs="0" name="keyId" type="ns_p:DeviceConfigurationKeyIdType"/>
            <xs:element minOccurs="0" name="value"
type="ns_p:DeviceConfigurationKeyValueValueType"/>
            <xs:element minOccurs="0" name="isValueChangeable" type="xs:boolean"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="deviceConfigurationKeyValueData"
type="ns_p:DeviceConfigurationKeyValueDataType"/>
    <xs:complexType name="DeviceConfigurationKeyValueDataElementsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="keyId" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="value"
type="ns_p:DeviceConfigurationKeyValueValueElementsType"/>
            <xs:element minOccurs="0" name="isValueChangeable" type="ns_p:ElementTagType"/>
        </xs:sequence>
```



```
14613       </xs:complexType>
14614       <xs:element name="deviceConfigurationKeyValueDataElements"
14615   type="ns_p:DeviceConfigurationKeyValueDataElementsType"/>
14616       <xs:complexType name="DeviceConfigurationKeyValueListDataType">
14617           <xs:sequence>
14618               <xs:element maxOccurs="unbounded" minOccurs="0"
14619   ref="ns_p:deviceConfigurationKeyValueData"/>
14620           </xs:sequence>
14621       </xs:complexType>
14622       <xs:element name="deviceConfigurationKeyValueListData"
14623   type="ns_p:DeviceConfigurationKeyValueListDataType"/>
14624       <xs:complexType name="DeviceConfigurationKeyValueListDataSelectorsType">
14625           <xs:sequence>
14626               <xs:element minOccurs="0" name="keyId" type="ns_p:DeviceConfigurationKeyIdType"/>
14627           </xs:sequence>
14628       </xs:complexType>
14629       <xs:element name="deviceConfigurationKeyValueListDataSelectors"
14630   type="ns_p:DeviceConfigurationKeyValueListDataSelectorsType"/>
14631       <xs:complexType name="DeviceConfigurationKeyValueDescriptionDataType">
14632           <xs:sequence>
14633               <xs:element minOccurs="0" name="keyId" type="ns_p:DeviceConfigurationKeyIdType"/>
14634               <xs:element minOccurs="0" name="keyName"
14635   type="ns_p:DeviceConfigurationKeyNameType"/>
14636               <xs:element minOccurs="0" name="valueType"
14637   type="ns_p:DeviceConfigurationKeyValueTypeType"/>
14638               <xs:element minOccurs="0" name="unit" type="ns_p:UnitOfMeasurementType"/>
14639               <xs:element minOccurs="0" name="label" type="ns_p:LabelType"/>
14640               <xs:element minOccurs="0" name="description" type="ns_p:DescriptionType"/>
14641           </xs:sequence>
14642       </xs:complexType>
14643       <xs:element name="deviceConfigurationKeyValueDescriptionData"
14644   type="ns_p:DeviceConfigurationKeyValueDescriptionDataType"/>
14645       <xs:complexType name="DeviceConfigurationKeyValueDescriptionDataElementsType">
14646           <xs:sequence>
14647               <xs:element minOccurs="0" name="keyId" type="ns_p:ElementTagType"/>
14648               <xs:element minOccurs="0" name="keyName" type="ns_p:ElementTagType"/>
14649               <xs:element minOccurs="0" name="valueType" type="ns_p:ElementTagType"/>
14650               <xs:element minOccurs="0" name="unit" type="ns_p:ElementTagType"/>
14651               <xs:element minOccurs="0" name="label" type="ns_p:ElementTagType"/>
14652               <xs:element minOccurs="0" name="description" type="ns_p:ElementTagType"/>
14653           </xs:sequence>
14654       </xs:complexType>
14655       <xs:element name="deviceConfigurationKeyValueDescriptionDataElements"
14656   type="ns_p:DeviceConfigurationKeyValueDescriptionDataElementsType"/>
14657       <xs:complexType name="DeviceConfigurationKeyValueDescriptionListDataType">
14658           <xs:sequence>
14659               <xs:element maxOccurs="unbounded" minOccurs="0"
14660   ref="ns_p:deviceConfigurationKeyValueDescriptionData"/>
14661           </xs:sequence>
14662       </xs:complexType>
14663       <xs:element name="deviceConfigurationKeyValueDescriptionListData"
14664   type="ns_p:DeviceConfigurationKeyValueDescriptionListDataType"/>
14665       <xs:complexType name="DeviceConfigurationKeyValueDescriptionListDataSelectorsType">
14666           <xs:sequence>
14667               <xs:element minOccurs="0" name="keyId" type="ns_p:DeviceConfigurationKeyIdType"/>
14668               <xs:element minOccurs="0" name="keyName"
14669   type="ns_p:DeviceConfigurationKeyNameType"/>
14670           </xs:sequence>
14671       </xs:complexType>
14672       <xs:element name="deviceConfigurationKeyValueDescriptionListDataSelectors"
14673   type="ns_p:DeviceConfigurationKeyValueDescriptionListDataSelectorsType"/>
14674       <xs:complexType name="DeviceConfigurationKeyValueConstraintsDataType">
14675           <xs:sequence>
14676               <xs:element minOccurs="0" name="keyId" type="ns_p:DeviceConfigurationKeyIdType"/>
14677               <xs:element minOccurs="0" name="valueRangeMin"
14678   type="ns_p:DeviceConfigurationKeyValueValueType"/>
14679               <xs:element minOccurs="0" name="valueRangeMax"
14680   type="ns_p:DeviceConfigurationKeyValueValueType"/>
14681               <xs:element minOccurs="0" name="valueStepSize"
14682   type="ns_p:DeviceConfigurationKeyValueValueType"/>
14683           </xs:sequence>
14684       </xs:complexType>
14685       <xs:element name="deviceConfigurationKeyValueConstraintsData"
14686   type="ns_p:DeviceConfigurationKeyValueConstraintsDataType"/>
14687       <xs:complexType name="DeviceConfigurationKeyValueConstraintsDataElementsType">
14688           <xs:sequence>
14689               <xs:element minOccurs="0" name="keyId" type="ns_p:ElementTagType"/>
```



```
            <xs:element minOccurs="0" name="valueRangeMin"
type="ns_p:DeviceConfigurationKeyValueValueElementsType"/>
            <xs:element minOccurs="0" name="valueRangeMax"
type="ns_p:DeviceConfigurationKeyValueValueElementsType"/>
            <xs:element minOccurs="0" name="valueStepSize"
type="ns_p:DeviceConfigurationKeyValueValueElementsType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="deviceConfigurationKeyValueConstraintsDataElements"
type="ns_p:DeviceConfigurationKeyValueConstraintsDataElementsType"/>
    <xs:complexType name="DeviceConfigurationKeyValueConstraintsListDataType">
        <xs:sequence>
            <xs:element maxOccurs="unbounded" minOccurs="0"
ref="ns_p:deviceConfigurationKeyValueConstraintsData"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="deviceConfigurationKeyValueConstraintsListData"
type="ns_p:DeviceConfigurationKeyValueConstraintsListDataType"/>
    <xs:complexType name="DeviceConfigurationKeyValueConstraintsListDataSelectorsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="keyId" type="ns_p:DeviceConfigurationKeyIdType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="deviceConfigurationKeyValueConstraintsListDataSelectors"
type="ns_p:DeviceConfigurationKeyValueConstraintsListDataSelectorsType"/>
</xs:schema>
```

A.2.9 DeviceDiagnosis

File name: EEBus_SPINE_TS_DeviceDiagnosis.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
    Smart Premises Interoperable Neutral-Message Exchange (SPINE)
    Version 1.1.1
    2018-12-21
    Copyright (c) 2018 EEBus Initiative e.V.  All Rights Reserved.
    Source: https://www.eebus.org/en/specifications/
-->
<xs:schema xmlns:ns_p="http://docs.eebus.org/spine/xsd/v1"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://docs.eebus.org/spine/xsd/v1" version="1.1.1"
    blockDefault="#all" elementFormDefault="qualified">
    <xs:annotation>
        <xs:documentation>EEBus SPINE Specification schema. Copyright 2018 EEBus Initiative
e.V.  All rights reserved.</xs:documentation>
    </xs:annotation>
    <xs:include schemaLocation="EEBus_SPINE_TS_CommonDataTypes.xsd"/>
    <xs:simpleType name="VendorStateCodeType">
        <xs:restriction base="xs:string"/>
    </xs:simpleType>
    <xs:simpleType name="LastErrorCodeType">
        <xs:restriction base="xs:string"/>
    </xs:simpleType>
    <xs:simpleType name="DeviceDiagnosisOperatingStateType">
        <xs:union memberTypes="ns_p:DeviceDiagnosisOperatingStateEnumType
ns_p:EnumExtendType"/>
    </xs:simpleType>
    <xs:simpleType name="DeviceDiagnosisOperatingStateEnumType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="normalOperation"/>
            <xs:enumeration value="standby"/>
            <xs:enumeration value="failure"/>
            <xs:enumeration value="serviceNeeded"/>
            <xs:enumeration value="overrideDetected"/>
            <xs:enumeration value="inAlarm"/>
            <xs:enumeration value="notReachable"/>
            <xs:enumeration value="finished"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="PowerSupplyConditionType">
        <xs:union memberTypes="ns_p:PowerSupplyConditionEnumType ns_p:EnumExtendType"/>
```

```
14763       </xs:simpleType>
14764       <xs:simpleType name="PowerSupplyConditionEnumType">
14765           <xs:restriction base="xs:string">
14766               <xs:enumeration value="good"/>
14767               <xs:enumeration value="low"/>
14768               <xs:enumeration value="critical"/>
14769               <xs:enumeration value="unknown"/>
14770               <xs:enumeration value="error"/>
14771           </xs:restriction>
14772       </xs:simpleType>
14773       <xs:complexType name="DeviceDiagnosisStateDataType">
14774           <xs:sequence>
14775               <xs:element name="timestamp" type="ns_p:AbsoluteOrRelativeTimeType"
14776   minOccurs="0"/>
14777               <xs:element name="operatingState" type="ns_p:DeviceDiagnosisOperatingStateType"
14778   minOccurs="0"/>
14779               <xs:element name="vendorStateCode" type="ns_p:VendorStateCodeType" minOccurs="0"/>
14780               <xs:element name="lastErrorCode" type="ns_p:LastErrorCodeType" minOccurs="0"/>
14781               <xs:element name="upTime" type="xs:duration" minOccurs="0"/>
14782               <xs:element name="totalUpTime" type="xs:duration" minOccurs="0"/>
14783               <xs:element name="powerSupplyCondition" minOccurs="0"
14784   type="ns_p:PowerSupplyConditionType"/>
14785           </xs:sequence>
14786       </xs:complexType>
14787       <xs:element name="deviceDiagnosisStateData" type="ns_p:DeviceDiagnosisStateDataType"/>
14788       <xs:complexType name="DeviceDiagnosisStateDataElementsType">
14789           <xs:sequence>
14790               <xs:element name="timestamp" minOccurs="0" type="ns_p:ElementTagType"/>
14791               <xs:element name="operatingState" minOccurs="0" type="ns_p:ElementTagType"/>
14792               <xs:element name="vendorStateCode" minOccurs="0" type="ns_p:ElementTagType"/>
14793               <xs:element name="lastErrorCode" minOccurs="0" type="ns_p:ElementTagType"/>
14794               <xs:element name="upTime" minOccurs="0" type="ns_p:ElementTagType"/>
14795               <xs:element name="totalUpTime" minOccurs="0" type="ns_p:ElementTagType"/>
14796               <xs:element name="powerSupplyCondition" minOccurs="0" type="ns_p:ElementTagType"/>
14797           </xs:sequence>
14798       </xs:complexType>
14799       <xs:element name="deviceDiagnosisStateDataElements"
14800   type="ns_p:DeviceDiagnosisStateDataElementsType"/>
14801       <xs:complexType name="DeviceDiagnosisHeartbeatDataType">
14802           <xs:sequence>
14803               <xs:element name="timestamp" type="ns_p:AbsoluteOrRelativeTimeType"
14804   minOccurs="0"/>
14805               <xs:element name="heartbeatCounter" type="xs:unsignedLong" minOccurs="0"/>
14806               <xs:element name="heartbeatTimeout" type="xs:duration" minOccurs="0"/>
14807           </xs:sequence>
14808       </xs:complexType>
14809       <xs:element name="deviceDiagnosisHeartbeatData"
14810   type="ns_p:DeviceDiagnosisHeartbeatDataType"/>
14811       <xs:complexType name="DeviceDiagnosisHeartbeatDataElementsType">
14812           <xs:sequence>
14813               <xs:element name="timestamp" minOccurs="0" type="ns_p:ElementTagType"/>
14814               <xs:element name="heartbeatCounter" minOccurs="0" type="ns_p:ElementTagType"/>
14815               <xs:element name="heartbeatTimeout" minOccurs="0" type="ns_p:ElementTagType"/>
14816           </xs:sequence>
14817       </xs:complexType>
14818       <xs:element name="deviceDiagnosisHeartbeatDataElements"
14819   type="ns_p:DeviceDiagnosisHeartbeatDataElementsType"/>
14820       <xs:complexType name="DeviceDiagnosisServiceDataType">
14821           <xs:sequence>
14822               <xs:element name="timestamp" type="ns_p:AbsoluteOrRelativeTimeType"
14823   minOccurs="0"/>
14824               <xs:element name="installationTime" type="ns_p:AbsoluteOrRelativeTimeType"
14825   minOccurs="0"/>
14826               <xs:element name="bootCounter" type="xs:unsignedLong" minOccurs="0"/>
14827               <xs:element name="nextService" type="ns_p:AbsoluteOrRelativeTimeType"
14828   minOccurs="0"/>
14829           </xs:sequence>
14830       </xs:complexType>
14831       <xs:element name="deviceDiagnosisServiceData" type="ns_p:DeviceDiagnosisServiceDataType"/>
14832       <xs:complexType name="DeviceDiagnosisServiceDataElementsType">
14833           <xs:sequence>
14834               <xs:element name="timestamp" minOccurs="0" type="ns_p:ElementTagType"/>
14835               <xs:element name="installationTime" minOccurs="0" type="ns_p:ElementTagType"/>
14836               <xs:element name="bootCounter" minOccurs="0" type="ns_p:ElementTagType"/>
14837               <xs:element name="nextService" minOccurs="0" type="ns_p:ElementTagType"/>
14838           </xs:sequence>
14839       </xs:complexType>
```



```
    <xs:element name="deviceDiagnosisServiceDataElements"
type="ns_p:DeviceDiagnosisServiceDataElementsType"/>
</xs:schema>
```

A.2.10 DirectControl

File name: EEBus_SPINE_TS_DirectControl.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
    Smart Premises Interoperable Neutral-Message Exchange (SPINE)
    Version 1.1.1
    2018-12-21
    Copyright (c) 2018 EEBus Initiative e.V.  All Rights Reserved.
    Source: https://www.eebus.org/en/specifications/
-->
<xs:schema xmlns:ns_p="http://docs.eebus.org/spine/xsd/v1"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://docs.eebus.org/spine/xsd/v1" version="1.1.1"
    blockDefault="#all" elementFormDefault="qualified">
    <xs:annotation>
        <xs:documentation>EEBus SPINE Specification schema. Copyright 2018 EEBus Initiative
e.V.  All rights reserved.</xs:documentation>
    </xs:annotation>
    <xs:include schemaLocation="EEBus_SPINE_TS_CommonDataTypes.xsd"/>
    <xs:include schemaLocation="EEBus_SPINE_TS_PowerSequences.xsd"/>
    <xs:simpleType name="DirectControlActivityStateType">
        <xs:union memberTypes="ns_p:DirectControlActivityStateEnumType ns_p:EnumExtendType"/>
    </xs:simpleType>
    <xs:simpleType name="DirectControlActivityStateEnumType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="running"/>
            <xs:enumeration value="paused"/>
            <xs:enumeration value="inactive"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:complexType name="DirectControlActivityDataType">
        <xs:sequence>
            <xs:element minOccurs="0" name="timestamp"
type="ns_p:AbsoluteOrRelativeTimeType"/>
            <xs:element minOccurs="0" name="activityState"
type="ns_p:DirectControlActivityStateType"/>
            <xs:element minOccurs="0" name="isActivityStateChangeable" type="xs:boolean"/>
            <xs:element minOccurs="0" name="energyMode" type="ns_p:EnergyModeType"/>
            <xs:element minOccurs="0" name="isEnergyModeChangeable" type="xs:boolean"/>
            <xs:element name="power" type="ns_p:ScaledNumberType" minOccurs="0"/>
            <xs:element minOccurs="0" name="isPowerChangeable" type="xs:boolean"/>
            <xs:element name="energy" type="ns_p:ScaledNumberType" minOccurs="0"/>
            <xs:element minOccurs="0" name="isEnergyChangeable" type="xs:boolean"/>
            <xs:element minOccurs="0" name="sequenceId" type="ns_p:PowerSequenceIdType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="directControlActivityData" type="ns_p:DirectControlActivityDataType"/>
    <xs:complexType name="DirectControlActivityDataElementsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="timestamp" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="activityState" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="isActivityStateChangeable"
type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="energyMode" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="isEnergyModeChangeable"
type="ns_p:ElementTagType"/>
            <xs:element name="power" minOccurs="0" type="ns_p:ScaledNumberElementsType"/>
            <xs:element minOccurs="0" name="isPowerChangeable" type="ns_p:ElementTagType"/>
            <xs:element name="energy" minOccurs="0" type="ns_p:ScaledNumberElementsType"/>
            <xs:element minOccurs="0" name="isEnergyChangeable" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="sequenceId" type="ns_p:ElementTagType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="directControlActivityDataElements"
type="ns_p:DirectControlActivityDataElementsType"/>
    <xs:complexType name="DirectControlActivityListDataType">
```



```
        <xs:sequence>
            <xs:element maxOccurs="unbounded" minOccurs="0"
ref="ns_p:directControlActivityData"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="directControlActivityListData"
type="ns_p:DirectControlActivityListDataType"/>
    <xs:complexType name="DirectControlActivityListDataSelectorsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="timestampInterval"
type="ns_p:TimestampIntervalType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="directControlActivityListDataSelectors"
type="ns_p:DirectControlActivityListDataSelectorsType"/>
    <xs:complexType name="DirectControlDescriptionDataType">
        <xs:sequence>
            <xs:element minOccurs="0" name="positiveEnergyDirection"
type="ns_p:EnergyDirectionType"/>
            <xs:element minOccurs="0" name="powerUnit" type="ns_p:UnitOfMeasurementType"/>
            <xs:element minOccurs="0" name="energyUnit" type="ns_p:UnitOfMeasurementType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="directControlDescriptionData"
type="ns_p:DirectControlDescriptionDataType"/>
    <xs:complexType name="DirectControlDescriptionDataElementsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="positiveEnergyDirection"
type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="powerUnit" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="energyUnit" type="ns_p:ElementTagType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="directControlDescriptionDataElements"
type="ns_p:DirectControlDescriptionDataElementsType"/>
</xs:schema>
```

A.2.11 ElectricalConnection

File name: EEBus_SPINE_TS_ElectricalConnection.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
    Smart Premises Interoperable Neutral-Message Exchange (SPINE)
    Version 1.1.1
    2018-12-21
    Copyright (c) 2018 EEBus Initiative e.V.  All Rights Reserved.
    Source: https://www.eebus.org/en/specifications/
-->
<xs:schema xmlns:ns_p="http://docs.eebus.org/spine/xsd/v1"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://docs.eebus.org/spine/xsd/v1" version="1.1.1"
    blockDefault="#all" elementFormDefault="qualified">
    <xs:annotation>
        <xs:documentation>EEBus SPINE Specification schema. Copyright 2018 EEBus Initiative
e.V.  All rights reserved.</xs:documentation>
    </xs:annotation>
    <xs:include schemaLocation="EEBus_SPINE_TS_CommonDataTypes.xsd"/>
    <xs:include schemaLocation="EEBus_SPINE_TS_Measurement.xsd"/>
    <xs:simpleType name="ElectricalConnectionIdType">
        <xs:restriction base="xs:unsignedInt"/>
    </xs:simpleType>
    <xs:simpleType name="ElectricalConnectionParameterIdType">
        <xs:restriction base="xs:unsignedInt"/>
    </xs:simpleType>
    <xs:simpleType name="ElectricalConnectionMeasurandVariantType">
        <xs:union memberTypes="ns_p:ElectricalConnectionMeasurandVariantEnumType
ns_p:EnumExtendType"/>
    </xs:simpleType>
    <xs:simpleType name="ElectricalConnectionMeasurandVariantEnumType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="amplitude"/>
```

```
14986             <xs:enumeration value="rms"/>
14987             <xs:enumeration value="instantaneous"/>
14988             <xs:enumeration value="angle"/>
14989             <xs:enumeration value="cosPhi"/>
14990         </xs:restriction>
14991     </xs:simpleType>
14992     <xs:simpleType name="ElectricalConnectionVoltageTypeType">
14993         <xs:union memberTypes="ns_p:ElectricalConnectionVoltageTypeEnumType
14994 ns_p:EnumExtendType"/>
14995     </xs:simpleType>
14996     <xs:simpleType name="ElectricalConnectionVoltageTypeEnumType">
14997         <xs:restriction base="xs:string">
14998             <xs:enumeration value="ac"/>
14999             <xs:enumeration value="dc"/>
15000         </xs:restriction>
15001     </xs:simpleType>
15002     <xs:simpleType name="ElectricalConnectionAcMeasurementTypeType">
15003         <xs:union memberTypes="ns_p:ElectricalConnectionAcMeasurementTypeEnumType
15004 ns_p:EnumExtendType"/>
15005     </xs:simpleType>
15006     <xs:simpleType name="ElectricalConnectionAcMeasurementTypeEnumType">
15007         <xs:restriction base="xs:string">
15008             <xs:enumeration value="real"/>
15009             <xs:enumeration value="reactive"/>
15010             <xs:enumeration value="apparent"/>
15011             <xs:enumeration value="phase"/>
15012         </xs:restriction>
15013     </xs:simpleType>
15014     <xs:simpleType name="ElectricalConnectionPhaseNameType">
15015         <xs:union memberTypes="ns_p:ElectricalConnectionPhaseNameEnumType
15016 ns_p:EnumExtendType"/>
15017     </xs:simpleType>
15018     <xs:simpleType name="ElectricalConnectionPhaseNameEnumType">
15019         <xs:restriction base="xs:string">
15020             <xs:enumeration value="a"/>
15021             <xs:enumeration value="b"/>
15022             <xs:enumeration value="c"/>
15023             <xs:enumeration value="ab"/>
15024             <xs:enumeration value="bc"/>
15025             <xs:enumeration value="ac"/>
15026             <xs:enumeration value="abc"/>
15027             <xs:enumeration value="neutral"/>
15028             <xs:enumeration value="ground"/>
15029             <xs:enumeration value="none"/>
15030         </xs:restriction>
15031     </xs:simpleType>
15032     <xs:simpleType name="ElectricalConnectionConnectionPointType">
15033         <xs:restriction base="xs:string">
15034             <xs:enumeration value="grid"/>
15035             <xs:enumeration value="home"/>
15036             <xs:enumeration value="pv"/>
15037             <xs:enumeration value="sd"/>
15038             <xs:enumeration value="other"/>
15039         </xs:restriction>
15040     </xs:simpleType>
15041     <xs:complexType name="ElectricalConnectionParameterDescriptionDataType">
15042         <xs:sequence>
15043             <xs:element name="electricalConnectionId" minOccurs="0"
15044 type="ns_p:ElectricalConnectionIdType"/>
15045             <xs:element name="parameterId" minOccurs="0"
15046 type="ns_p:ElectricalConnectionParameterIdType"/>
15047             <xs:element minOccurs="0" name="measurementId" type="ns_p:MeasurementIdType"/>
15048             <xs:element minOccurs="0" name="voltageType"
15049 type="ns_p:ElectricalConnectionVoltageTypeType"/>
15050             <xs:element name="acMeasuredPhases" minOccurs="0"
15051 type="ns_p:ElectricalConnectionPhaseNameType"/>
15052             <xs:element name="acMeasuredInReferenceTo" minOccurs="0"
15053 type="ns_p:ElectricalConnectionPhaseNameType"/>
15054             <xs:element name="acMeasurementType"
15055 type="ns_p:ElectricalConnectionAcMeasurementTypeType" minOccurs="0"/>
15056             <xs:element name="acMeasurementVariant" minOccurs="0"
15057 type="ns_p:ElectricalConnectionMeasurandVariantType"/>
15058             <xs:element minOccurs="0" name="acMeasuredHarmonic" type="xs:unsignedByte"/>
15059             <xs:element minOccurs="0" name="scopeType" type="ns_p:ScopeTypeType"/>
15060             <xs:element minOccurs="0" name="label" type="ns_p:LabelType"/>
15061             <xs:element minOccurs="0" name="description" type="ns_p:DescriptionType"/>
15062         </xs:sequence>
15063     </xs:complexType>
```



```
15064       <xs:element name="electricalConnectionParameterDescriptionData"
15065   type="ns_p:ElectricalConnectionParameterDescriptionDataType"/>
15066       <xs:complexType name="ElectricalConnectionParameterDescriptionDataElementsType">
15067           <xs:sequence>
15068               <xs:element name="electricalConnectionId" minOccurs="0"
15069   type="ns_p:ElementTagType"/>
15070               <xs:element name="parameterId" minOccurs="0" type="ns_p:ElementTagType"/>
15071               <xs:element minOccurs="0" name="measurementId" type="ns_p:ElementTagType"/>
15072               <xs:element minOccurs="0" name="voltageType" type="ns_p:ElementTagType"/>
15073               <xs:element name="acMeasuredPhases" minOccurs="0" type="ns_p:ElementTagType"/>
15074               <xs:element name="acMeasuredInReferenceTo" minOccurs="0"
15075   type="ns_p:ElementTagType"/>
15076               <xs:element name="acMeasurementType" minOccurs="0" type="ns_p:ElementTagType"/>
15077               <xs:element name="acMeasurementVariant" minOccurs="0" type="ns_p:ElementTagType"/>
15078               <xs:element minOccurs="0" name="acMeasuredHarmonic" type="ns_p:ElementTagType"/>
15079               <xs:element minOccurs="0" name="scopeType" type="ns_p:ElementTagType"/>
15080               <xs:element minOccurs="0" name="label" type="ns_p:ElementTagType"/>
15081               <xs:element minOccurs="0" name="description" type="ns_p:ElementTagType"/>
15082           </xs:sequence>
15083       </xs:complexType>
15084       <xs:element name="electricalConnectionParameterDescriptionDataElements"
15085   type="ns_p:ElectricalConnectionParameterDescriptionDataElementsType"/>
15086       <xs:complexType name="ElectricalConnectionParameterDescriptionListDataType">
15087           <xs:sequence>
15088               <xs:element maxOccurs="unbounded" minOccurs="0"
15089   ref="ns_p:electricalConnectionParameterDescriptionData"/>
15090           </xs:sequence>
15091       </xs:complexType>
15092       <xs:element name="electricalConnectionParameterDescriptionListData"
15093   type="ns_p:ElectricalConnectionParameterDescriptionListDataType"/>
15094       <xs:complexType name="ElectricalConnectionParameterDescriptionListDataSelectorsType">
15095           <xs:sequence>
15096               <xs:element name="electricalConnectionId" minOccurs="0"
15097   type="ns_p:ElectricalConnectionIdType"/>
15098               <xs:element name="parameterId" minOccurs="0"
15099   type="ns_p:ElectricalConnectionParameterIdType"/>
15100               <xs:element minOccurs="0" name="measurementId" type="ns_p:MeasurementIdType"/>
15101               <xs:element minOccurs="0" name="scopeType" type="ns_p:ScopeTypeType"/>
15102           </xs:sequence>
15103       </xs:complexType>
15104       <xs:element name="electricalConnectionParameterDescriptionListDataSelectors"
15105   type="ns_p:ElectricalConnectionParameterDescriptionListDataSelectorsType"/>
15106       <xs:complexType name="ElectricalConnectionPermittedValueSetDataType">
15107           <xs:sequence>
15108               <xs:element name="electricalConnectionId" minOccurs="0"
15109   type="ns_p:ElectricalConnectionIdType"/>
15110               <xs:element name="parameterId" minOccurs="0"
15111   type="ns_p:ElectricalConnectionParameterIdType"/>
15112               <xs:element minOccurs="0" name="permittedValueSet" type="ns_p:ScaledNumberSetType"
15113   maxOccurs="unbounded"/>
15114           </xs:sequence>
15115       </xs:complexType>
15116       <xs:element name="electricalConnectionPermittedValueSetData"
15117   type="ns_p:ElectricalConnectionPermittedValueSetDataType"/>
15118       <xs:complexType name="ElectricalConnectionPermittedValueSetDataElementsType">
15119           <xs:sequence>
15120               <xs:element name="electricalConnectionId" minOccurs="0"
15121   type="ns_p:ElementTagType"/>
15122               <xs:element name="parameterId" minOccurs="0" type="ns_p:ElementTagType"/>
15123               <xs:element minOccurs="0" name="permittedValueSet"
15124   type="ns_p:ScaledNumberSetElementsType"/>
15125           </xs:sequence>
15126       </xs:complexType>
15127       <xs:element name="electricalConnectionPermittedValueSetDataElements"
15128   type="ns_p:ElectricalConnectionPermittedValueSetDataElementsType"/>
15129       <xs:complexType name="ElectricalConnectionPermittedValueSetListDataType">
15130           <xs:sequence>
15131               <xs:element maxOccurs="unbounded" minOccurs="0"
15132   ref="ns_p:electricalConnectionPermittedValueSetData"/>
15133           </xs:sequence>
15134       </xs:complexType>
15135       <xs:element name="electricalConnectionPermittedValueSetListData"
15136   type="ns_p:ElectricalConnectionPermittedValueSetListDataType"/>
15137       <xs:complexType name="ElectricalConnectionPermittedValueSetListDataSelectorsType">
15138           <xs:sequence>
15139               <xs:element name="electricalConnectionId" minOccurs="0"
15140   type="ns_p:ElectricalConnectionIdType"/>
```



```
            <xs:element name="parameterId" minOccurs="0"
type="ns_p:ElectricalConnectionParameterIdType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="electricalConnectionPermittedValueSetListDataSelectors"
type="ns_p:ElectricalConnectionPermittedValueSetListDataSelectorsType"/>

    <xs:complexType name="ElectricalConnectionStateDataType">
        <xs:sequence>
            <xs:element name="electricalConnectionId" minOccurs="0"
type="ns_p:ElectricalConnectionIdType"/>
            <xs:element minOccurs="0" name="timestamp"
type="ns_p:AbsoluteOrRelativeTimeType"/>
            <xs:element name="currentEnergyMode" minOccurs="0" type="ns_p:EnergyModeType"/>
            <xs:element minOccurs="0" name="consumptionTime" type="xs:duration"/>
            <xs:element minOccurs="0" name="productionTime" type="xs:duration"/>
            <xs:element minOccurs="0" name="totalConsumptionTime" type="xs:duration"/>
            <xs:element minOccurs="0" name="totalProductionTime" type="xs:duration"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="electricalConnectionStateData"
type="ns_p:ElectricalConnectionStateDataType"/>
    <xs:complexType name="ElectricalConnectionStateDataElementsType">
        <xs:sequence>
            <xs:element name="electricalConnectionId" minOccurs="0"
type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="timestamp" type="ns_p:ElementTagType"/>
            <xs:element name="currentEnergyMode" minOccurs="0" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="consumptionTime" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="productionTime" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="totalConsumptionTime" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="totalProductionTime" type="ns_p:ElementTagType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="electricalConnectionStateDataElements"
type="ns_p:ElectricalConnectionStateDataElementsType"/>
    <xs:complexType name="ElectricalConnectionStateListDataType">
        <xs:sequence>
            <xs:element maxOccurs="unbounded" minOccurs="0"
ref="ns_p:electricalConnectionStateData"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="electricalConnectionStateListData"
type="ns_p:ElectricalConnectionStateListDataType"/>
    <xs:complexType name="ElectricalConnectionStateListDataSelectorsType">
        <xs:sequence>
            <xs:element name="electricalConnectionId" minOccurs="0"
type="ns_p:ElectricalConnectionIdType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="electricalConnectionStateListDataSelectors"
type="ns_p:ElectricalConnectionStateListDataSelectorsType"/>
    <xs:complexType name="ElectricalConnectionDescriptionDataType">
        <xs:sequence>
            <xs:element name="electricalConnectionId" minOccurs="0"
type="ns_p:ElectricalConnectionIdType"/>
            <xs:element name="powerSupplyType" type="ns_p:ElectricalConnectionVoltageTypeType"
minOccurs="0"/>
            <xs:element name="acConnectedPhases" minOccurs="0" type="xs:unsignedInt"/>
            <xs:element minOccurs="0" name="acRmsPeriodDuration" type="xs:duration"/>
            <xs:element minOccurs="0" name="positiveEnergyDirection"
type="ns_p:EnergyDirectionType"/>
            <xs:element minOccurs="0" name="scopeType" type="ns_p:ScopeTypeType"/>
            <xs:element minOccurs="0" name="label" type="ns_p:LabelType"/>
            <xs:element minOccurs="0" name="description" type="ns_p:DescriptionType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="electricalConnectionDescriptionData"
type="ns_p:ElectricalConnectionDescriptionDataType"/>
    <xs:complexType name="ElectricalConnectionDescriptionDataElementsType">
        <xs:sequence>
            <xs:element name="electricalConnectionId" minOccurs="0"
type="ns_p:ElementTagType"/>
            <xs:element name="powerSupplyType" minOccurs="0" type="ns_p:ElementTagType"/>
            <xs:element name="acConnectedPhases" minOccurs="0" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="acRmsPeriodDuration" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="positiveEnergyDirection"
type="ns_p:ElementTagType"/>
```



```
            <xs:element minOccurs="0" name="scopeType" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="label" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="description" type="ns_p:ElementTagType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="electricalConnectionDescriptionDataElements"
type="ns_p:ElectricalConnectionDescriptionDataElementsType"/>
    <xs:complexType name="ElectricalConnectionDescriptionListDataType">
        <xs:sequence>
            <xs:element maxOccurs="unbounded" minOccurs="0"
ref="ns_p:electricalConnectionDescriptionData"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="electricalConnectionDescriptionListData"
type="ns_p:ElectricalConnectionDescriptionListDataType"/>
    <xs:complexType name="ElectricalConnectionDescriptionListDataSelectorsType">
        <xs:sequence>
            <xs:element name="electricalConnectionId" minOccurs="0"
type="ns_p:ElectricalConnectionIdType"/>
            <xs:element minOccurs="0" name="scopeType" type="ns_p:ScopeTypeType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="electricalConnectionDescriptionListDataSelectors"
type="ns_p:ElectricalConnectionDescriptionListDataSelectorsType"/>
</xs:schema>
```

A.2.12 HVAC

File name: EEBus_SPINE_TS_HVAC.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
    Smart Premises Interoperable Neutral-Message Exchange (SPINE)
    Version 1.1.1
    2018-12-21
    Copyright (c) 2018 EEBus Initiative e.V.  All Rights Reserved.
    Source: https://www.eebus.org/en/specifications/
-->
<xs:schema xmlns:ns_p="http://docs.eebus.org/spine/xsd/v1"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://docs.eebus.org/spine/xsd/v1" version="1.1.1"
    blockDefault="#all" elementFormDefault="qualified">
    <xs:annotation>
        <xs:documentation>EEBus SPINE Specification schema. Copyright 2018 EEBus Initiative
e.V.  All rights reserved.</xs:documentation>
    </xs:annotation>
    <xs:include schemaLocation="EEBus_SPINE_TS_CommonDataTypes.xsd"/>
    <xs:include schemaLocation="EEBus_SPINE_TS_PowerSequences.xsd"/>
    <xs:include schemaLocation="EEBus_SPINE_TS_Setpoint.xsd"/>
    <xs:simpleType name="HvacSystemFunctionIdType">
        <xs:restriction base="xs:unsignedInt"/>
    </xs:simpleType>
    <xs:simpleType name="HvacSystemFunctionTypeType">
        <xs:union memberTypes="ns_p:HvacSystemFunctionTypeEnumType ns_p:EnumExtendType"/>
    </xs:simpleType>
    <xs:simpleType name="HvacSystemFunctionTypeEnumType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="heating"/>
            <xs:enumeration value="cooling"/>
            <xs:enumeration value="ventilation"/>
            <xs:enumeration value="dhw"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="HvacOperationModeIdType">
        <xs:restriction base="xs:unsignedInt"/>
    </xs:simpleType>
    <xs:simpleType name="HvacOperationModeTypeType">
        <xs:union memberTypes="ns_p:HvacOperationModeTypeEnumType ns_p:EnumExtendType"/>
    </xs:simpleType>
    <xs:simpleType name="HvacOperationModeTypeEnumType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="auto"/>
```

```
15292             <xs:enumeration value="on"/>
15293             <xs:enumeration value="off"/>
15294             <xs:enumeration value="eco"/>
15295         </xs:restriction>
15296     </xs:simpleType>
15297     <xs:simpleType name="HvacOverrunIdType">
15298         <xs:restriction base="xs:unsignedInt"/>
15299     </xs:simpleType>
15300     <xs:simpleType name="HvacOverrunTypeType">
15301         <xs:union memberTypes="ns_p:HvacOverrunTypeEnumType ns_p:EnumExtendType"/>
15302     </xs:simpleType>
15303     <xs:simpleType name="HvacOverrunTypeEnumType">
15304         <xs:restriction base="xs:string">
15305             <xs:enumeration value="oneTimeDhw"/>
15306             <xs:enumeration value="party"/>
15307             <xs:enumeration value="sgReadyCondition1"/>
15308             <xs:enumeration value="sgReadyCondition3"/>
15309             <xs:enumeration value="sgReadyCondition4"/>
15310             <xs:enumeration value="oneDayAway"/>
15311             <xs:enumeration value="oneDayAtHome"/>
15312             <xs:enumeration value="oneTimeVentilation"/>
15313             <xs:enumeration value="hvacSystemOff"/>
15314             <xs:enumeration value="valveKick"/>
15315         </xs:restriction>
15316     </xs:simpleType>
15317     <xs:simpleType name="HvacOverrunStatusType">
15318         <xs:union memberTypes="ns_p:HvacOverrunStatusEnumType ns_p:EnumExtendType"/>
15319     </xs:simpleType>
15320     <xs:simpleType name="HvacOverrunStatusEnumType">
15321         <xs:restriction base="xs:string">
15322             <xs:enumeration value="active"/>
15323             <xs:enumeration value="running"/>
15324             <xs:enumeration value="finished"/>
15325             <xs:enumeration value="inactive"/>
15326         </xs:restriction>
15327     </xs:simpleType>
15328     <xs:complexType name="HvacSystemFunctionDataType">
15329         <xs:sequence>
15330             <xs:element minOccurs="0" name="systemFunctionId"
15331 type="ns_p:HvacSystemFunctionIdType"/>
15332             <xs:element minOccurs="0" name="currentOperationModeId"
15333 type="ns_p:HvacOperationModeIdType"/>
15334             <xs:element minOccurs="0" name="isOperationModeIdChangeable" type="xs:boolean"/>
15335             <xs:element minOccurs="0" name="currentSetpointId" type="ns_p:SetpointIdType"/>
15336             <xs:element minOccurs="0" name="isSetpointIdChangeable" type="xs:boolean"/>
15337             <xs:element minOccurs="0" name="isOverrunActive" type="xs:boolean"/>
15338         </xs:sequence>
15339     </xs:complexType>
15340     <xs:element name="hvacSystemFunctionData" type="ns_p:HvacSystemFunctionDataType"/>
15341     <xs:complexType name="HvacSystemFunctionDataElementsType">
15342         <xs:sequence>
15343             <xs:element minOccurs="0" name="systemFunctionId" type="ns_p:ElementTagType"/>
15344             <xs:element minOccurs="0" name="currentOperationModeId"
15345 type="ns_p:ElementTagType"/>
15346             <xs:element minOccurs="0" name="isOperationModeIdChangeable"
15347 type="ns_p:ElementTagType"/>
15348             <xs:element minOccurs="0" name="currentSetpointId" type="ns_p:ElementTagType"/>
15349             <xs:element minOccurs="0" name="isSetpointIdChangeable"
15350 type="ns_p:ElementTagType"/>
15351             <xs:element minOccurs="0" name="isOverrunActive" type="ns_p:ElementTagType"/>
15352         </xs:sequence>
15353     </xs:complexType>
15354     <xs:element name="hvacSystemFunctionDataElements"
15355 type="ns_p:HvacSystemFunctionDataElementsType"/>
15356     <xs:complexType name="HvacSystemFunctionListDataType">
15357         <xs:sequence>
15358             <xs:element maxOccurs="unbounded" minOccurs="0"
15359 ref="ns_p:hvacSystemFunctionData"/>
15360         </xs:sequence>
15361     </xs:complexType>
15362     <xs:element name="hvacSystemFunctionListData" type="ns_p:HvacSystemFunctionListDataType"/>
15363     <xs:complexType name="HvacSystemFunctionListDataSelectorsType">
15364         <xs:sequence>
15365             <xs:element minOccurs="0" name="systemFunctionId"
15366 type="ns_p:HvacSystemFunctionIdType"/>
15367         </xs:sequence>
15368     </xs:complexType>
```



```
15369       <xs:element name="hvacSystemFunctionListDataSelectors"
15370   type="ns_p:HvacSystemFunctionListDataSelectorsType"/>
15371       <xs:complexType name="HvacSystemFunctionOperationModeRelationDataType">
15372           <xs:sequence>
15373               <xs:element minOccurs="0" name="systemFunctionId"
15374   type="ns_p:HvacSystemFunctionIdType"/>
15375               <xs:element minOccurs="0" name="operationModeId"
15376   type="ns_p:HvacOperationModeIdType" maxOccurs="unbounded"/>
15377           </xs:sequence>
15378       </xs:complexType>
15379       <xs:element name="hvacSystemFunctionOperationModeRelationData"
15380   type="ns_p:HvacSystemFunctionOperationModeRelationDataType"/>
15381       <xs:complexType name="HvacSystemFunctionOperationModeRelationDataElementsType">
15382           <xs:sequence>
15383               <xs:element minOccurs="0" name="systemFunctionId" type="ns_p:ElementTagType"/>
15384               <xs:element minOccurs="0" name="operationModeId" type="ns_p:ElementTagType"/>
15385           </xs:sequence>
15386       </xs:complexType>
15387       <xs:element name="hvacSystemFunctionOperationModeRelationDataElements"
15388   type="ns_p:HvacSystemFunctionOperationModeRelationDataElementsType"/>
15389       <xs:complexType name="HvacSystemFunctionOperationModeRelationListDataType">
15390           <xs:sequence>
15391               <xs:element maxOccurs="unbounded" minOccurs="0"
15392   ref="ns_p:hvacSystemFunctionOperationModeRelationData"/>
15393           </xs:sequence>
15394       </xs:complexType>
15395       <xs:element name="hvacSystemFunctionOperationModeRelationListData"
15396   type="ns_p:HvacSystemFunctionOperationModeRelationListDataType"/>
15397       <xs:complexType name="HvacSystemFunctionOperationModeRelationListDataSelectorsType">
15398           <xs:sequence>
15399               <xs:element minOccurs="0" name="systemFunctionId"
15400   type="ns_p:HvacSystemFunctionIdType"/>
15401           </xs:sequence>
15402       </xs:complexType>
15403       <xs:element name="hvacSystemFunctionOperationModeRelationListDataSelectors"
15404   type="ns_p:HvacSystemFunctionOperationModeRelationListDataSelectorsType"/>
15405       <xs:complexType name="HvacSystemFunctionSetpointRelationDataType">
15406           <xs:sequence>
15407               <xs:element minOccurs="0" name="systemFunctionId"
15408   type="ns_p:HvacSystemFunctionIdType"/>
15409               <xs:element minOccurs="0" name="operationModeId"
15410   type="ns_p:HvacOperationModeIdType"/>
15411               <xs:element minOccurs="0" name="setpointId" type="ns_p:SetpointIdType"
15412   maxOccurs="unbounded"/>
15413           </xs:sequence>
15414       </xs:complexType>
15415       <xs:element name="hvacSystemFunctionSetpointRelationData"
15416   type="ns_p:HvacSystemFunctionSetpointRelationDataType"/>
15417       <xs:complexType name="HvacSystemFunctionSetpointRelationDataElementsType">
15418           <xs:sequence>
15419               <xs:element minOccurs="0" name="systemFunctionId" type="ns_p:ElementTagType"/>
15420               <xs:element minOccurs="0" name="operationModeId" type="ns_p:ElementTagType"/>
15421               <xs:element minOccurs="0" name="setpointId" type="ns_p:ElementTagType"/>
15422           </xs:sequence>
15423       </xs:complexType>
15424       <xs:element name="hvacSystemFunctionSetpointRelationDataElements"
15425   type="ns_p:HvacSystemFunctionSetpointRelationDataElementsType"/>
15426       <xs:complexType name="HvacSystemFunctionSetpointRelationListDataType">
15427           <xs:sequence>
15428               <xs:element maxOccurs="unbounded" minOccurs="0"
15429   ref="ns_p:hvacSystemFunctionSetpointRelationData"/>
15430           </xs:sequence>
15431       </xs:complexType>
15432       <xs:element name="hvacSystemFunctionSetpointRelationListData"
15433   type="ns_p:HvacSystemFunctionSetpointRelationListDataType"/>
15434       <xs:complexType name="HvacSystemFunctionSetpointRelationListDataSelectorsType">
15435           <xs:sequence>
15436               <xs:element minOccurs="0" name="systemFunctionId"
15437   type="ns_p:HvacSystemFunctionIdType"/>
15438               <xs:element minOccurs="0" name="operationModeId"
15439   type="ns_p:HvacOperationModeIdType"/>
15440           </xs:sequence>
15441       </xs:complexType>
15442       <xs:element name="hvacSystemFunctionSetpointRelationListDataSelectors"
15443   type="ns_p:HvacSystemFunctionSetpointRelationListDataSelectorsType"/>
15444       <xs:complexType name="HvacSystemFunctionPowerSequenceRelationDataType">
15445           <xs:sequence>
```



```
15446             <xs:element minOccurs="0" name="systemFunctionId"
15447 type="ns_p:HvacSystemFunctionIdType"/>
15448             <xs:element minOccurs="0" name="sequenceId" type="ns_p:PowerSequenceIdType"
15449 maxOccurs="unbounded"/>
15450         </xs:sequence>
15451     </xs:complexType>
15452     <xs:element name="hvacSystemFunctionPowerSequenceRelationData"
15453 type="ns_p:HvacSystemFunctionPowerSequenceRelationDataType"/>
15454     <xs:complexType name="HvacSystemFunctionPowerSequenceRelationDataElementsType">
15455         <xs:sequence>
15456             <xs:element minOccurs="0" name="systemFunctionId" type="ns_p:ElementTagType"/>
15457             <xs:element minOccurs="0" name="sequenceId" type="ns_p:ElementTagType"/>
15458         </xs:sequence>
15459     </xs:complexType>
15460     <xs:element name="hvacSystemFunctionPowerSequenceRelationDataElements"
15461 type="ns_p:HvacSystemFunctionPowerSequenceRelationDataElementsType"/>
15462     <xs:complexType name="HvacSystemFunctionPowerSequenceRelationListDataType">
15463         <xs:sequence>
15464             <xs:element maxOccurs="unbounded" minOccurs="0"
15465 ref="ns_p:hvacSystemFunctionPowerSequenceRelationData"/>
15466         </xs:sequence>
15467     </xs:complexType>
15468     <xs:element name="hvacSystemFunctionPowerSequenceRelationListData"
15469 type="ns_p:HvacSystemFunctionPowerSequenceRelationListDataType"/>
15470     <xs:complexType name="HvacSystemFunctionPowerSequenceRelationListDataSelectorsType">
15471         <xs:sequence>
15472             <xs:element minOccurs="0" name="systemFunctionId"
15473 type="ns_p:HvacSystemFunctionIdType"/>
15474         </xs:sequence>
15475     </xs:complexType>
15476     <xs:element name="hvacSystemFunctionPowerSequenceRelationListDataSelectors"
15477 type="ns_p:HvacSystemFunctionPowerSequenceRelationListDataSelectorsType"/>
15478     <xs:complexType name="HvacSystemFunctionDescriptionDataType">
15479         <xs:sequence>
15480             <xs:element minOccurs="0" name="systemFunctionId"
15481 type="ns_p:HvacSystemFunctionIdType"/>
15482             <xs:element minOccurs="0" name="systemFunctionType"
15483 type="ns_p:HvacSystemFunctionTypeType"/>
15484             <xs:element minOccurs="0" name="label" type="ns_p:LabelType"/>
15485             <xs:element minOccurs="0" name="description" type="ns_p:DescriptionType"/>
15486         </xs:sequence>
15487     </xs:complexType>
15488     <xs:element name="hvacSystemFunctionDescriptionData"
15489 type="ns_p:HvacSystemFunctionDescriptionDataType"/>
15490     <xs:complexType name="HvacSystemFunctionDescriptionDataElementsType">
15491         <xs:sequence>
15492             <xs:element minOccurs="0" name="systemFunctionId" type="ns_p:ElementTagType"/>
15493             <xs:element minOccurs="0" name="systemFunctionType" type="ns_p:ElementTagType"/>
15494             <xs:element minOccurs="0" name="label" type="ns_p:ElementTagType"/>
15495             <xs:element minOccurs="0" name="description" type="ns_p:ElementTagType"/>
15496         </xs:sequence>
15497     </xs:complexType>
15498     <xs:element name="hvacSystemFunctionDescriptionDataElements"
15499 type="ns_p:HvacSystemFunctionDescriptionDataElementsType"/>
15500     <xs:complexType name="HvacSystemFunctionDescriptionListDataType">
15501         <xs:sequence>
15502             <xs:element maxOccurs="unbounded" minOccurs="0"
15503 ref="ns_p:hvacSystemFunctionDescriptionData"/>
15504         </xs:sequence>
15505     </xs:complexType>
15506     <xs:element name="hvacSystemFunctionDescriptionListData"
15507 type="ns_p:HvacSystemFunctionDescriptionListDataType"/>
15508     <xs:complexType name="HvacSystemFunctionDescriptionListDataSelectorsType">
15509         <xs:sequence>
15510             <xs:element minOccurs="0" name="systemFunctionId"
15511 type="ns_p:HvacSystemFunctionIdType"/>
15512         </xs:sequence>
15513     </xs:complexType>
15514     <xs:element name="hvacSystemFunctionDescriptionListDataSelectors"
15515 type="ns_p:HvacSystemFunctionDescriptionListDataSelectorsType"/>
15516     <xs:complexType name="HvacOperationModeDescriptionDataType">
15517         <xs:sequence>
15518             <xs:element minOccurs="0" name="operationModeId"
15519 type="ns_p:HvacOperationModeIdType"/>
15520             <xs:element minOccurs="0" name="operationModeType"
15521 type="ns_p:HvacOperationModeTypeType"/>
15522             <xs:element minOccurs="0" name="label" type="ns_p:LabelType"/>
15523             <xs:element minOccurs="0" name="description" type="ns_p:DescriptionType"/>
```



```
15524           </xs:sequence>
15525       </xs:complexType>
15526       <xs:element name="hvacOperationModeDescriptionData"
15527   type="ns_p:HvacOperationModeDescriptionDataType"/>
15528       <xs:complexType name="HvacOperationModeDescriptionDataElementsType">
15529           <xs:sequence>
15530               <xs:element minOccurs="0" name="operationModeId" type="ns_p:ElementTagType"/>
15531               <xs:element minOccurs="0" name="operationModeType" type="ns_p:ElementTagType"/>
15532               <xs:element minOccurs="0" name="label" type="ns_p:ElementTagType"/>
15533               <xs:element minOccurs="0" name="description" type="ns_p:ElementTagType"/>
15534           </xs:sequence>
15535       </xs:complexType>
15536       <xs:element name="hvacOperationModeDescriptionDataElements"
15537   type="ns_p:HvacOperationModeDescriptionDataElementsType"/>
15538       <xs:complexType name="HvacOperationModeDescriptionListDataType">
15539           <xs:sequence>
15540               <xs:element maxOccurs="unbounded" minOccurs="0"
15541   ref="ns_p:hvacOperationModeDescriptionData"/>
15542           </xs:sequence>
15543       </xs:complexType>
15544       <xs:element name="hvacOperationModeDescriptionListData"
15545   type="ns_p:HvacOperationModeDescriptionListDataType"/>
15546       <xs:complexType name="HvacOperationModeDescriptionListDataSelectorsType">
15547           <xs:sequence>
15548               <xs:element minOccurs="0" name="operationModeId"
15549   type="ns_p:HvacOperationModeIdType"/>
15550           </xs:sequence>
15551       </xs:complexType>
15552       <xs:element name="hvacOperationModeDescriptionListDataSelectors"
15553   type="ns_p:HvacOperationModeDescriptionListDataSelectorsType"/>
15554       <xs:complexType name="HvacOverrunDataType">
15555           <xs:sequence>
15556               <xs:element name="overrunId" minOccurs="0" type="ns_p:HvacOverrunIdType"/>
15557               <xs:element name="overrunStatus" minOccurs="0" type="ns_p:HvacOverrunStatusType"/>
15558               <xs:element minOccurs="0" name="timeTableId" type="ns_p:TimeTableIdType"/>
15559               <xs:element minOccurs="0" name="isOverrunStatusChangeable" type="xs:boolean"/>
15560           </xs:sequence>
15561       </xs:complexType>
15562       <xs:element name="hvacOverrunData" type="ns_p:HvacOverrunDataType"/>
15563       <xs:complexType name="HvacOverrunDataElementsType">
15564           <xs:sequence>
15565               <xs:element name="overrunId" minOccurs="0" type="ns_p:ElementTagType"/>
15566               <xs:element name="overrunStatus" minOccurs="0" type="ns_p:ElementTagType"/>
15567               <xs:element minOccurs="0" name="timeTableId" type="ns_p:ElementTagType"/>
15568               <xs:element minOccurs="0" name="isOverrunStatusChangeable"
15569   type="ns_p:ElementTagType"/>
15570           </xs:sequence>
15571       </xs:complexType>
15572       <xs:element name="hvacOverrunDataElements" type="ns_p:HvacOverrunDataElementsType"/>
15573       <xs:complexType name="HvacOverrunListDataType">
15574           <xs:sequence>
15575               <xs:element maxOccurs="unbounded" minOccurs="0" ref="ns_p:hvacOverrunData"/>
15576           </xs:sequence>
15577       </xs:complexType>
15578       <xs:element name="hvacOverrunListData" type="ns_p:HvacOverrunListDataType"/>
15579       <xs:complexType name="HvacOverrunListDataSelectorsType">
15580           <xs:sequence>
15581               <xs:element name="overrunId" minOccurs="0" type="ns_p:HvacOverrunIdType"/>
15582           </xs:sequence>
15583       </xs:complexType>
15584       <xs:element name="hvacOverrunListDataSelectors"
15585   type="ns_p:HvacOverrunListDataSelectorsType"/>
15586       <xs:complexType name="HvacOverrunDescriptionDataType">
15587           <xs:sequence>
15588               <xs:element name="overrunId" minOccurs="0" type="ns_p:HvacOverrunIdType"/>
15589               <xs:element minOccurs="0" name="overrunType" type="ns_p:HvacOverrunTypeType"/>
15590               <xs:element minOccurs="0" name="affectedSystemFunctionId"
15591   type="ns_p:HvacSystemFunctionIdType" maxOccurs="unbounded"/>
15592               <xs:element minOccurs="0" name="label" type="ns_p:LabelType"/>
15593               <xs:element minOccurs="0" name="description" type="ns_p:DescriptionType"/>
15594           </xs:sequence>
15595       </xs:complexType>
15596       <xs:element name="hvacOverrunDescriptionData" type="ns_p:HvacOverrunDescriptionDataType"/>
15597       <xs:complexType name="HvacOverrunDescriptionDataElementsType">
15598           <xs:sequence>
15599               <xs:element name="overrunId" minOccurs="0" type="ns_p:ElementTagType"/>
15600               <xs:element minOccurs="0" name="overrunType" type="ns_p:ElementTagType"/>
```



```
            <xs:element minOccurs="0" name="affectedSystemFunctionId"
type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="label" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="description" type="ns_p:ElementTagType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="hvacOverrunDescriptionDataElements"
type="ns_p:HvacOverrunDescriptionDataElementsType"/>
    <xs:complexType name="HvacOverrunDescriptionListDataType">
        <xs:sequence>
            <xs:element maxOccurs="unbounded" minOccurs="0"
ref="ns_p:hvacOverrunDescriptionData"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="hvacOverrunDescriptionListData"
type="ns_p:HvacOverrunDescriptionListDataType"/>
    <xs:complexType name="HvacOverrunDescriptionListDataSelectorsType">
        <xs:sequence>
            <xs:element name="overrunId" minOccurs="0" type="ns_p:HvacOverrunIdType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="hvacOverrunDescriptionListDataSelectors"
type="ns_p:HvacOverrunDescriptionListDataSelectorsType"/>
</xs:schema>
```

A.2.13 Identification

File name: EEBus_SPINE_TS_Identification.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
    Smart Premises Interoperable Neutral-Message Exchange (SPINE)
    Version 1.1.1
    2018-12-21
    Copyright (c) 2018 EEBus Initiative e.V.  All Rights Reserved.
    Source: https://www.eebus.org/en/specifications/
-->
<xs:schema xmlns:ns_p="http://docs.eebus.org/spine/xsd/v1"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://docs.eebus.org/spine/xsd/v1" version="1.1.1"
    blockDefault="#all" elementFormDefault="qualified">
    <xs:annotation>
        <xs:documentation>EEBus SPINE Specification schema. Copyright 2018 EEBus Initiative
e.V.  All rights reserved.</xs:documentation>
    </xs:annotation>
    <xs:include schemaLocation="EEBus_SPINE_TS_CommonDataTypes.xsd"/>
    <xs:simpleType name="IdentificationIdType">
        <xs:restriction base="xs:unsignedInt"/>
    </xs:simpleType>
    <xs:simpleType name="IdentificationTypeType">
        <xs:union memberTypes="ns_p:IdentificationTypeEnumType ns_p:EnumExtendType"/>
    </xs:simpleType>
    <xs:simpleType name="IdentificationTypeEnumType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="eui48"/>
            <xs:enumeration value="eui64"/>
            <xs:enumeration value="userRfidTag"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="IdentificationValueType">
        <xs:restriction base="xs:string"/>
    </xs:simpleType>
    <xs:complexType name="IdentificationDataType">
        <xs:sequence>
            <xs:element minOccurs="0" name="identificationId"
type="ns_p:IdentificationIdType"/>
            <xs:element minOccurs="0" name="identificationType"
type="ns_p:IdentificationTypeType"/>
            <xs:element minOccurs="0" name="identificationValue"
type="ns_p:IdentificationValueType"/>
            <xs:element minOccurs="0" name="authorized" type="xs:boolean"/>
        </xs:sequence>
```

```
    </xs:complexType>
    <xs:element name="identificationData" type="ns_p:IdentificationDataType"/>
    <xs:complexType name="IdentificationDataElementsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="identificationId" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="identificationType" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="identificationValue" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="authorized" type="ns_p:ElementTagType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="identificationDataElements" type="ns_p:IdentificationDataElementsType"/>
    <xs:complexType name="IdentificationListDataType">
        <xs:sequence>
            <xs:element maxOccurs="unbounded" minOccurs="0" ref="ns_p:identificationData"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="identificationListData" type="ns_p:IdentificationListDataType"/>
    <xs:complexType name="IdentificationListDataSelectorsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="identificationId"
type="ns_p:IdentificationIdType"/>
            <xs:element minOccurs="0" name="identificationType"
type="ns_p:IdentificationTypeType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="identificationListDataSelectors"
type="ns_p:IdentificationListDataSelectorsType"/>
</xs:schema>
```

A.2.14 LoadControl

File name: EEBus_SPINE_TS_LoadControl.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
    Smart Premises Interoperable Neutral-Message Exchange (SPINE)
    Version 1.1.1
    2018-12-21
    Copyright (c) 2018 EEBus Initiative e.V.  All Rights Reserved.
    Source: https://www.eebus.org/en/specifications/
-->
<xs:schema xmlns:ns_p="http://docs.eebus.org/spine/xsd/v1"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://docs.eebus.org/spine/xsd/v1" version="1.1.1"
    blockDefault="#all" elementFormDefault="qualified">
    <xs:annotation>
        <xs:documentation>EEBus SPINE Specification schema. Copyright 2018 EEBus Initiative
e.V.  All rights reserved.</xs:documentation>
    </xs:annotation>
    <xs:include schemaLocation="EEBus_SPINE_TS_CommonDataTypes.xsd"/>
    <xs:include schemaLocation="EEBus_SPINE_TS_Measurement.xsd"/>
    <xs:simpleType name="LoadControlEventIdType">
        <xs:restriction base="xs:unsignedInt"/>
    </xs:simpleType>
    <xs:simpleType name="LoadControlEventActionType">
        <xs:union memberTypes="ns_p:LoadControlEventActionEnumType ns_p:EnumExtendType"/>
    </xs:simpleType>
    <xs:simpleType name="LoadControlEventActionEnumType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="pause"/>
            <xs:enumeration value="resume"/>
            <xs:enumeration value="reduce"/>
            <xs:enumeration value="increase"/>
            <xs:enumeration value="emergency"/>
            <xs:enumeration value="normal"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="LoadControlEventStateType">
        <xs:union memberTypes="ns_p:LoadControlEventStateEnumType ns_p:EnumExtendType"/>
    </xs:simpleType>
    <xs:simpleType name="LoadControlEventStateEnumType">
        <xs:restriction base="xs:string">
```



```
15747             <xs:enumeration value="eventAccepted"/>
15748             <xs:enumeration value="eventStarted"/>
15749             <xs:enumeration value="eventStopped"/>
15750             <xs:enumeration value="eventRejected"/>
15751             <xs:enumeration value="eventCancelled"/>
15752             <xs:enumeration value="eventError"/>
15753         </xs:restriction>
15754     </xs:simpleType>
15755     <xs:simpleType name="LoadControlLimitIdType">
15756         <xs:restriction base="xs:unsignedInt"/>
15757     </xs:simpleType>
15758     <xs:simpleType name="LoadControlLimitTypeType">
15759         <xs:union memberTypes="ns_p:LoadControlLimitTypeEnumType ns_p:EnumExtendType"/>
15760     </xs:simpleType>
15761     <xs:simpleType name="LoadControlLimitTypeEnumType">
15762         <xs:restriction base="xs:string">
15763             <xs:enumeration value="minValueLimit"/>
15764             <xs:enumeration value="maxValueLimit"/>
15765         </xs:restriction>
15766     </xs:simpleType>
15767     <xs:simpleType name="LoadControlCategoryType">
15768         <xs:union memberTypes="ns_p:LoadControlCategoryEnumType ns_p:EnumExtendType"/>
15769     </xs:simpleType>
15770     <xs:simpleType name="LoadControlCategoryEnumType">
15771         <xs:restriction base="xs:string">
15772             <xs:enumeration value="obligation"/>
15773             <xs:enumeration value="recommendation"/>
15774             <xs:enumeration value="optimization"/>
15775         </xs:restriction>
15776     </xs:simpleType>
15777     <xs:complexType name="LoadControlNodeDataType">
15778         <xs:sequence>
15779             <xs:element minOccurs="0" name="isNodeRemoteControllable" type="xs:boolean"/>
15780         </xs:sequence>
15781     </xs:complexType>
15782     <xs:element name="loadControlNodeData" type="ns_p:LoadControlNodeDataType"/>
15783     <xs:complexType name="LoadControlNodeDataElementsType">
15784         <xs:sequence>
15785             <xs:element minOccurs="0" name="isNodeRemoteControllable"
15786 type="ns_p:ElementTagType"/>
15787         </xs:sequence>
15788     </xs:complexType>
15789     <xs:element name="loadControlNodeDataElements"
15790 type="ns_p:LoadControlNodeDataElementsType"/>
15791     <xs:complexType name="LoadControlEventDataType">
15792         <xs:sequence>
15793             <xs:element minOccurs="0" name="timestamp"
15794 type="ns_p:AbsoluteOrRelativeTimeType"/>
15795             <xs:element minOccurs="0" name="eventId" type="ns_p:LoadControlEventIdType"/>
15796             <xs:element minOccurs="0" name="eventActionConsume"
15797 type="ns_p:LoadControlEventActionType"/>
15798             <xs:element minOccurs="0" name="eventActionProduce"
15799 type="ns_p:LoadControlEventActionType"/>
15800             <xs:element minOccurs="0" name="timePeriod" type="ns_p:TimePeriodType"/>
15801         </xs:sequence>
15802     </xs:complexType>
15803     <xs:element name="loadControlEventData" type="ns_p:LoadControlEventDataType"/>
15804     <xs:complexType name="LoadControlEventDataElementsType">
15805         <xs:sequence>
15806             <xs:element minOccurs="0" name="timestamp" type="ns_p:ElementTagType"/>
15807             <xs:element minOccurs="0" name="eventId" type="ns_p:ElementTagType"/>
15808             <xs:element minOccurs="0" name="eventActionConsume" type="ns_p:ElementTagType"/>
15809             <xs:element minOccurs="0" name="eventActionProduce" type="ns_p:ElementTagType"/>
15810             <xs:element minOccurs="0" name="timePeriod" type="ns_p:TimePeriodElementsType"/>
15811         </xs:sequence>
15812     </xs:complexType>
15813     <xs:element name="loadControlEventDataElements"
15814 type="ns_p:LoadControlEventDataElementsType"/>
15815     <xs:complexType name="LoadControlEventListDataType">
15816         <xs:sequence>
15817             <xs:element maxOccurs="unbounded" minOccurs="0" ref="ns_p:loadControlEventData"/>
15818         </xs:sequence>
15819     </xs:complexType>
15820     <xs:element name="loadControlEventListData" type="ns_p:LoadControlEventListDataType"/>
15821     <xs:complexType name="LoadControlEventListDataSelectorsType">
15822         <xs:sequence>
15823             <xs:element minOccurs="0" name="timestampInterval"
15824 type="ns_p:TimestampIntervalType"/>
```



```
15825             <xs:element minOccurs="0" name="eventId" type="ns_p:LoadControlEventIdType"/>
15826         </xs:sequence>
15827     </xs:complexType>
15828     <xs:element name="loadControlEventListDataSelectors"
15829 type="ns_p:LoadControlEventListDataSelectorsType"/>
15830     <xs:complexType name="LoadControlStateDataType">
15831         <xs:sequence>
15832             <xs:element minOccurs="0" name="timestamp"
15833 type="ns_p:AbsoluteOrRelativeTimeType"/>
15834             <xs:element minOccurs="0" name="eventId" type="ns_p:LoadControlEventIdType"/>
15835             <xs:element minOccurs="0" name="eventStateConsume"
15836 type="ns_p:LoadControlEventStateType"/>
15837             <xs:element minOccurs="0" name="appliedEventActionConsume"
15838 type="ns_p:LoadControlEventActionType"/>
15839             <xs:element minOccurs="0" name="eventStateProduce"
15840 type="ns_p:LoadControlEventStateType"/>
15841             <xs:element minOccurs="0" name="appliedEventActionProduce"
15842 type="ns_p:LoadControlEventActionType"/>
15843         </xs:sequence>
15844     </xs:complexType>
15845     <xs:element name="loadControlStateData" type="ns_p:LoadControlStateDataType"/>
15846     <xs:complexType name="LoadControlStateDataElementsType">
15847         <xs:sequence>
15848             <xs:element minOccurs="0" name="timestamp" type="ns_p:ElementTagType"/>
15849             <xs:element minOccurs="0" name="eventId" type="ns_p:ElementTagType"/>
15850             <xs:element minOccurs="0" name="eventStateConsume" type="ns_p:ElementTagType"/>
15851             <xs:element minOccurs="0" name="appliedEventActionConsume"
15852 type="ns_p:ElementTagType"/>
15853             <xs:element minOccurs="0" name="eventStateProduce" type="ns_p:ElementTagType"/>
15854             <xs:element minOccurs="0" name="appliedEventActionProduce"
15855 type="ns_p:ElementTagType"/>
15856         </xs:sequence>
15857     </xs:complexType>
15858     <xs:element name="loadControlStateDataElements"
15859 type="ns_p:LoadControlStateDataElementsType"/>
15860     <xs:complexType name="LoadControlStateListDataType">
15861         <xs:sequence>
15862             <xs:element maxOccurs="unbounded" minOccurs="0" ref="ns_p:loadControlStateData"/>
15863         </xs:sequence>
15864     </xs:complexType>
15865     <xs:element name="loadControlStateListData" type="ns_p:LoadControlStateListDataType"/>
15866     <xs:complexType name="LoadControlStateListDataSelectorsType">
15867         <xs:sequence>
15868             <xs:element minOccurs="0" name="timestampInterval"
15869 type="ns_p:TimestampIntervalType"/>
15870             <xs:element minOccurs="0" name="eventId" type="ns_p:LoadControlEventIdType"/>
15871         </xs:sequence>
15872     </xs:complexType>
15873     <xs:element name="loadControlStateListDataSelectors"
15874 type="ns_p:LoadControlStateListDataSelectorsType"/>
15875     <xs:complexType name="LoadControlLimitDataType">
15876         <xs:sequence>
15877             <xs:element minOccurs="0" name="limitId" type="ns_p:LoadControlLimitIdType"/>
15878             <xs:element name="isLimitChangeable" minOccurs="0" type="xs:boolean"/>
15879             <xs:element name="isLimitActive" minOccurs="0" type="xs:boolean"/>
15880             <xs:element minOccurs="0" name="timePeriod" type="ns_p:TimePeriodType"/>
15881             <xs:element minOccurs="0" name="value" type="ns_p:ScaledNumberType"/>
15882         </xs:sequence>
15883     </xs:complexType>
15884     <xs:element name="loadControlLimitData" type="ns_p:LoadControlLimitDataType"/>
15885     <xs:complexType name="LoadControlLimitDataElementsType">
15886         <xs:sequence>
15887             <xs:element minOccurs="0" name="limitId" type="ns_p:ElementTagType"/>
15888             <xs:element name="isLimitChangeable" minOccurs="0" type="ns_p:ElementTagType"/>
15889             <xs:element name="isLimitActive" minOccurs="0" type="ns_p:ElementTagType"/>
15890             <xs:element minOccurs="0" name="timePeriod" type="ns_p:TimePeriodElementsType"/>
15891             <xs:element minOccurs="0" name="value" type="ns_p:ScaledNumberElementsType"/>
15892         </xs:sequence>
15893     </xs:complexType>
15894     <xs:element name="loadControlLimitDataElements"
15895 type="ns_p:LoadControlLimitDataElementsType"/>
15896     <xs:complexType name="LoadControlLimitListDataType">
15897         <xs:sequence>
15898             <xs:element maxOccurs="unbounded" minOccurs="0" ref="ns_p:loadControlLimitData"/>
15899         </xs:sequence>
15900     </xs:complexType>
15901     <xs:element name="loadControlLimitListData" type="ns_p:LoadControlLimitListDataType"/>
15902     <xs:complexType name="LoadControlLimitListDataSelectorsType">
```



```
15903           <xs:sequence>
15904               <xs:element minOccurs="0" name="limitId" type="ns_p:LoadControlLimitIdType"/>
15905           </xs:sequence>
15906       </xs:complexType>
15907       <xs:element name="loadControlLimitListDataSelectors"
15908   type="ns_p:LoadControlLimitListDataSelectorsType"/>
15909       <xs:complexType name="LoadControlLimitConstraintsDataType">
15910           <xs:sequence>
15911               <xs:element minOccurs="0" name="limitId" type="ns_p:LoadControlLimitIdType"/>
15912               <xs:element minOccurs="0" name="valueRangeMin" type="ns_p:ScaledNumberType"/>
15913               <xs:element minOccurs="0" name="valueRangeMax" type="ns_p:ScaledNumberType"/>
15914               <xs:element minOccurs="0" name="valueStepSize" type="ns_p:ScaledNumberType"/>
15915           </xs:sequence>
15916       </xs:complexType>
15917       <xs:element name="loadControlLimitConstraintsData"
15918   type="ns_p:LoadControlLimitConstraintsDataType"/>
15919       <xs:complexType name="LoadControlLimitConstraintsDataElementsType">
15920           <xs:sequence>
15921               <xs:element minOccurs="0" name="limitId" type="ns_p:ElementTagType"/>
15922               <xs:element minOccurs="0" name="valueRangeMin"
15923   type="ns_p:ScaledNumberElementsType"/>
15924               <xs:element minOccurs="0" name="valueRangeMax"
15925   type="ns_p:ScaledNumberElementsType"/>
15926               <xs:element minOccurs="0" name="valueStepSize"
15927   type="ns_p:ScaledNumberElementsType"/>
15928           </xs:sequence>
15929       </xs:complexType>
15930       <xs:element name="loadControlLimitConstraintsDataElements"
15931   type="ns_p:LoadControlLimitConstraintsDataElementsType"/>
15932       <xs:complexType name="LoadControlLimitConstraintsListDataType">
15933           <xs:sequence>
15934               <xs:element maxOccurs="unbounded" minOccurs="0"
15935   ref="ns_p:loadControlLimitConstraintsData"/>
15936           </xs:sequence>
15937       </xs:complexType>
15938       <xs:element name="loadControlLimitConstraintsListData"
15939   type="ns_p:LoadControlLimitConstraintsListDataType"/>
15940       <xs:complexType name="LoadControlLimitConstraintsListDataSelectorsType">
15941           <xs:sequence>
15942               <xs:element minOccurs="0" name="limitId" type="ns_p:LoadControlLimitIdType"/>
15943           </xs:sequence>
15944       </xs:complexType>
15945       <xs:element name="loadControlLimitConstraintsListDataSelectors"
15946   type="ns_p:LoadControlLimitConstraintsListDataSelectorsType"/>
15947       <xs:complexType name="LoadControlLimitDescriptionDataType">
15948           <xs:sequence>
15949               <xs:element minOccurs="0" name="limitId" type="ns_p:LoadControlLimitIdType"/>
15950               <xs:element minOccurs="0" name="limitType" type="ns_p:LoadControlLimitTypeType"/>
15951               <xs:element minOccurs="0" name="limitCategory"
15952   type="ns_p:LoadControlCategoryType"/>
15953               <xs:element minOccurs="0" name="limitDirection" type="ns_p:EnergyDirectionType"/>
15954               <xs:element minOccurs="0" name="measurementId" type="ns_p:MeasurementIdType"/>
15955               <xs:element minOccurs="0" name="unit" type="ns_p:UnitOfMeasurementType"/>
15956               <xs:element minOccurs="0" name="scopeType" type="ns_p:ScopeTypeType"/>
15957               <xs:element minOccurs="0" name="label" type="ns_p:LabelType"/>
15958               <xs:element minOccurs="0" name="description" type="ns_p:DescriptionType"/>
15959           </xs:sequence>
15960       </xs:complexType>
15961       <xs:element name="loadControlLimitDescriptionData"
15962   type="ns_p:LoadControlLimitDescriptionDataType"/>
15963       <xs:complexType name="LoadControlLimitDescriptionDataElementsType">
15964           <xs:sequence>
15965               <xs:element minOccurs="0" name="limitId" type="ns_p:ElementTagType"/>
15966               <xs:element minOccurs="0" name="limitType" type="ns_p:ElementTagType"/>
15967               <xs:element minOccurs="0" name="limitCategory" type="ns_p:ElementTagType"/>
15968               <xs:element minOccurs="0" name="limitDirection" type="ns_p:ElementTagType"/>
15969               <xs:element minOccurs="0" name="measurementId" type="ns_p:ElementTagType"/>
15970               <xs:element minOccurs="0" name="unit" type="ns_p:ElementTagType"/>
15971               <xs:element minOccurs="0" name="scopeType" type="ns_p:ElementTagType"/>
15972               <xs:element minOccurs="0" name="label" type="ns_p:ElementTagType"/>
15973               <xs:element minOccurs="0" name="description" type="ns_p:ElementTagType"/>
15974           </xs:sequence>
15975       </xs:complexType>
15976       <xs:element name="loadControlLimitDescriptionDataElements"
15977   type="ns_p:LoadControlLimitDescriptionDataElementsType"/>
15978       <xs:complexType name="LoadControlLimitDescriptionListDataType">
15979           <xs:sequence>
```



```
            <xs:element maxOccurs="unbounded" minOccurs="0"
ref="ns_p:loadControlLimitDescriptionData"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="loadControlLimitDescriptionListData"
type="ns_p:LoadControlLimitDescriptionListDataType"/>
    <xs:complexType name="LoadControlLimitDescriptionListDataSelectorsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="limitId" type="ns_p:LoadControlLimitIdType"/>
            <xs:element minOccurs="0" name="limitType" type="ns_p:LoadControlLimitTypeType"/>
            <xs:element minOccurs="0" name="limitDirection" type="ns_p:EnergyDirectionType"/>
            <xs:element minOccurs="0" name="measurementId" type="ns_p:MeasurementIdType"/>
            <xs:element minOccurs="0" name="scopeType" type="ns_p:ScopeTypeType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="loadControlLimitDescriptionListDataSelectors"
type="ns_p:LoadControlLimitDescriptionListDataSelectorsType"/>
</xs:schema>
```

A.2.15 Measurement

File name: EEBus_SPINE_TS_Measurement.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
    Smart Premises Interoperable Neutral-Message Exchange (SPINE)
    Version 1.1.1
    2018-12-21
    Copyright (c) 2018 EEBus Initiative e.V.  All Rights Reserved.
    Source: https://www.eebus.org/en/specifications/
-->
<xs:schema xmlns:ns_p="http://docs.eebus.org/spine/xsd/v1"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://docs.eebus.org/spine/xsd/v1" version="1.1.1"
    blockDefault="#all" elementFormDefault="qualified">
    <xs:annotation>
        <xs:documentation>EEBus SPINE Specification schema. Copyright 2018 EEBus Initiative
e.V.  All rights reserved.</xs:documentation>
    </xs:annotation>
    <xs:include schemaLocation="EEBus_SPINE_TS_CommonDataTypes.xsd"/>
    <xs:include schemaLocation="EEBus_SPINE_TS_Threshold.xsd"/>
    <xs:simpleType name="MeasurementIdType">
        <xs:restriction base="xs:unsignedInt"/>
    </xs:simpleType>
    <xs:simpleType name="MeasurementTypeType">
        <xs:union memberTypes="ns_p:MeasurementTypeEnumType ns_p:EnumExtendType"/>
    </xs:simpleType>
    <xs:simpleType name="MeasurementTypeEnumType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="acceleration"/>
            <xs:enumeration value="angle"/>
            <xs:enumeration value="angularVelocity"/>
            <xs:enumeration value="area"/>
            <xs:enumeration value="atmosphericPressure"/>
            <xs:enumeration value="capacity"/>
            <xs:enumeration value="concentration"/>
            <xs:enumeration value="count"/>
            <xs:enumeration value="current"/>
            <xs:enumeration value="density"/>
            <xs:enumeration value="distance"/>
            <xs:enumeration value="electricField"/>
            <xs:enumeration value="energy"/>
            <xs:enumeration value="force"/>
            <xs:enumeration value="frequency"/>
            <xs:enumeration value="harmonicDistortion"/>
            <xs:enumeration value="heat"/>
            <xs:enumeration value="heatFlux"/>
            <xs:enumeration value="illuminance"/>
            <xs:enumeration value="impulse"/>
            <xs:enumeration value="level"/>
            <xs:enumeration value="magneticField"/>
            <xs:enumeration value="mass"/>
```



```
16053             <xs:enumeration value="massFlow"/>
16054             <xs:enumeration value="particles"/>
16055             <xs:enumeration value="percentage"/>
16056             <xs:enumeration value="power"/>
16057             <xs:enumeration value="powerFactor"/>
16058             <xs:enumeration value="pressure"/>
16059             <xs:enumeration value="radonActivity"/>
16060             <xs:enumeration value="relativeHumidity"/>
16061             <xs:enumeration value="resistance"/>
16062             <xs:enumeration value="solarRadiation"/>
16063             <xs:enumeration value="speed"/>
16064             <xs:enumeration value="temperature"/>
16065             <xs:enumeration value="time"/>
16066             <xs:enumeration value="torque"/>
16067             <xs:enumeration value="unknown"/>
16068             <xs:enumeration value="velocity"/>
16069             <xs:enumeration value="voltage"/>
16070             <xs:enumeration value="volume"/>
16071             <xs:enumeration value="volumetricFlow"/>
16072         </xs:restriction>
16073     </xs:simpleType>
16074     <xs:simpleType name="MeasurementValueTypeType">
16075         <xs:union memberTypes="ns_p:MeasurementValueTypeEnumType ns_p:EnumExtendType"/>
16076     </xs:simpleType>
16077     <xs:simpleType name="MeasurementValueTypeEnumType">
16078         <xs:restriction base="xs:string">
16079             <xs:enumeration value="value"/>
16080             <xs:enumeration value="averageValue"/>
16081             <xs:enumeration value="minValue"/>
16082             <xs:enumeration value="maxValue"/>
16083             <xs:enumeration value="standardDeviation"/>
16084         </xs:restriction>
16085     </xs:simpleType>
16086     <xs:simpleType name="MeasurementValueSourceType">
16087         <xs:union memberTypes="ns_p:MeasurementValueSourceEnumType ns_p:EnumExtendType"/>
16088     </xs:simpleType>
16089     <xs:simpleType name="MeasurementValueSourceEnumType">
16090         <xs:restriction base="xs:string">
16091             <xs:enumeration value="measuredValue"/>
16092             <xs:enumeration value="calculatedValue"/>
16093             <xs:enumeration value="empiricalValue"/>
16094         </xs:restriction>
16095     </xs:simpleType>
16096     <xs:simpleType name="MeasurementValueTendencyType">
16097         <xs:union memberTypes="ns_p:MeasurementValueTendencyEnumType ns_p:EnumExtendType"/>
16098     </xs:simpleType>
16099     <xs:simpleType name="MeasurementValueTendencyEnumType">
16100         <xs:restriction base="xs:string">
16101             <xs:enumeration value="rising"/>
16102             <xs:enumeration value="stable"/>
16103             <xs:enumeration value="falling"/>
16104         </xs:restriction>
16105     </xs:simpleType>
16106     <xs:simpleType name="MeasurementValueStateType">
16107         <xs:union memberTypes="ns_p:MeasurementValueStateEnumType ns_p:EnumExtendType"/>
16108     </xs:simpleType>
16109     <xs:simpleType name="MeasurementValueStateEnumType">
16110         <xs:restriction base="xs:string">
16111             <xs:enumeration value="normal"/>
16112             <xs:enumeration value="outOfRange"/>
16113             <xs:enumeration value="error"/>
16114         </xs:restriction>
16115     </xs:simpleType>
16116     <xs:complexType name="MeasurementDataType">
16117         <xs:sequence>
16118             <xs:element minOccurs="0" name="measurementId" type="ns_p:MeasurementIdType"/>
16119             <xs:element minOccurs="0" name="valueType" type="ns_p:MeasurementValueTypeType"/>
16120             <xs:element minOccurs="0" name="timestamp"
16121 type="ns_p:AbsoluteOrRelativeTimeType"/>
16122             <xs:element minOccurs="0" name="value" type="ns_p:ScaledNumberType"/>
16123             <xs:element minOccurs="0" name="evaluationPeriod" type="ns_p:TimePeriodType"/>
16124             <xs:element minOccurs="0" name="valueSource"
16125 type="ns_p:MeasurementValueSourceType"/>
16126             <xs:element minOccurs="0" name="valueTendency"
16127 type="ns_p:MeasurementValueTendencyType"/>
16128             <xs:element minOccurs="0" name="valueState"
16129 type="ns_p:MeasurementValueStateType"/>
16130         </xs:sequence>
```



```
16131       </xs:complexType>
16132       <xs:element name="measurementData" type="ns_p:MeasurementDataType"/>
16133       <xs:complexType name="MeasurementDataElementsType">
16134           <xs:sequence>
16135               <xs:element minOccurs="0" name="measurementId" type="ns_p:ElementTagType"/>
16136               <xs:element minOccurs="0" name="valueType" type="ns_p:ElementTagType"/>
16137               <xs:element minOccurs="0" name="timestamp" type="ns_p:ElementTagType"/>
16138               <xs:element minOccurs="0" name="value" type="ns_p:ScaledNumberElementsType"/>
16139               <xs:element minOccurs="0" name="evaluationPeriod"
16140   type="ns_p:TimePeriodElementsType"/>
16141               <xs:element minOccurs="0" name="valueSource" type="ns_p:ElementTagType"/>
16142               <xs:element minOccurs="0" name="valueTendency" type="ns_p:ElementTagType"/>
16143               <xs:element minOccurs="0" name="valueState" type="ns_p:ElementTagType"/>
16144           </xs:sequence>
16145       </xs:complexType>
16146       <xs:element name="measurementDataElements" type="ns_p:MeasurementDataElementsType"/>
16147       <xs:complexType name="MeasurementListDataType">
16148           <xs:sequence>
16149               <xs:element maxOccurs="unbounded" minOccurs="0" ref="ns_p:measurementData"/>
16150           </xs:sequence>
16151       </xs:complexType>
16152       <xs:element name="measurementListData" type="ns_p:MeasurementListDataType"/>
16153       <xs:complexType name="MeasurementListDataSelectorsType">
16154           <xs:sequence>
16155               <xs:element minOccurs="0" name="measurementId" type="ns_p:MeasurementIdType"/>
16156               <xs:element minOccurs="0" name="valueType" type="ns_p:MeasurementValueTypeType"/>
16157               <xs:element minOccurs="0" name="timestampInterval"
16158   type="ns_p:TimestampIntervalType"/>
16159           </xs:sequence>
16160       </xs:complexType>
16161       <xs:element name="measurementListDataSelectors"
16162   type="ns_p:MeasurementListDataSelectorsType"/>
16163       <xs:complexType name="MeasurementConstraintsDataType">
16164           <xs:sequence>
16165               <xs:element minOccurs="0" name="measurementId" type="ns_p:MeasurementIdType"/>
16166               <xs:element minOccurs="0" name="valueRangeMin" type="ns_p:ScaledNumberType"/>
16167               <xs:element minOccurs="0" name="valueRangeMax" type="ns_p:ScaledNumberType"/>
16168               <xs:element minOccurs="0" name="valueStepSize" type="ns_p:ScaledNumberType"/>
16169           </xs:sequence>
16170       </xs:complexType>
16171       <xs:element name="measurementConstraintsData" type="ns_p:MeasurementConstraintsDataType"/>
16172       <xs:complexType name="MeasurementConstraintsDataElementsType">
16173           <xs:sequence>
16174               <xs:element minOccurs="0" name="measurementId" type="ns_p:ElementTagType"/>
16175               <xs:element minOccurs="0" name="valueRangeMin"
16176   type="ns_p:ScaledNumberElementsType"/>
16177               <xs:element minOccurs="0" name="valueRangeMax"
16178   type="ns_p:ScaledNumberElementsType"/>
16179               <xs:element minOccurs="0" name="valueStepSize"
16180   type="ns_p:ScaledNumberElementsType"/>
16181           </xs:sequence>
16182       </xs:complexType>
16183       <xs:element name="measurementConstraintsDataElements"
16184   type="ns_p:MeasurementConstraintsDataElementsType"/>
16185       <xs:complexType name="MeasurementConstraintsListDataType">
16186           <xs:sequence>
16187               <xs:element maxOccurs="unbounded" minOccurs="0"
16188   ref="ns_p:measurementConstraintsData"/>
16189           </xs:sequence>
16190       </xs:complexType>
16191       <xs:element name="measurementConstraintsListData"
16192   type="ns_p:MeasurementConstraintsListDataType"/>
16193       <xs:complexType name="MeasurementConstraintsListDataSelectorsType">
16194           <xs:sequence>
16195               <xs:element minOccurs="0" name="measurementId" type="ns_p:MeasurementIdType"/>
16196           </xs:sequence>
16197       </xs:complexType>
16198       <xs:element name="measurementConstraintsListDataSelectors"
16199   type="ns_p:MeasurementConstraintsListDataSelectorsType"/>
16200       <xs:complexType name="MeasurementDescriptionDataType">
16201           <xs:sequence>
16202               <xs:element minOccurs="0" name="measurementId" type="ns_p:MeasurementIdType"/>
16203               <xs:element minOccurs="0" name="measurementType" type="ns_p:MeasurementTypeType"/>
16204               <xs:element minOccurs="0" name="commodityType" type="ns_p:CommodityTypeType"/>
16205               <xs:element minOccurs="0" name="unit" type="ns_p:UnitOfMeasurementType"/>
16206               <xs:element minOccurs="0" name="calibrationValue" type="ns_p:ScaledNumberType"/>
16207               <xs:element minOccurs="0" name="scopeType" type="ns_p:ScopeTypeType"/>
16208               <xs:element minOccurs="0" name="label" type="ns_p:LabelType"/>
```



```
            <xs:element minOccurs="0" name="description" type="ns_p:DescriptionType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="measurementDescriptionData" type="ns_p:MeasurementDescriptionDataType"/>
    <xs:complexType name="MeasurementDescriptionDataElementsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="measurementId" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="measurementType" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="commodityType" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="unit" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="calibrationValue"
type="ns_p:ScaledNumberElementsType"/>
            <xs:element minOccurs="0" name="scopeType" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="label" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="description" type="ns_p:ElementTagType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="measurementDescriptionDataElements"
type="ns_p:MeasurementDescriptionDataElementsType"/>
    <xs:complexType name="MeasurementDescriptionListDataType">
        <xs:sequence>
            <xs:element maxOccurs="unbounded" minOccurs="0"
ref="ns_p:measurementDescriptionData"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="measurementDescriptionListData"
type="ns_p:MeasurementDescriptionListDataType"/>
    <xs:complexType name="MeasurementDescriptionListDataSelectorsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="measurementId" type="ns_p:MeasurementIdType"/>
            <xs:element minOccurs="0" name="measurementType" type="ns_p:MeasurementTypeType"/>
            <xs:element minOccurs="0" name="commodityType" type="ns_p:CommodityTypeType"/>
            <xs:element minOccurs="0" name="scopeType" type="ns_p:ScopeTypeType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="measurementDescriptionListDataSelectors"
type="ns_p:MeasurementDescriptionListDataSelectorsType"/>
    <xs:complexType name="MeasurementThresholdRelationDataType">
        <xs:sequence>
            <xs:element minOccurs="0" name="measurementId" type="ns_p:MeasurementIdType"/>
            <xs:element maxOccurs="unbounded" minOccurs="0" name="thresholdId"
type="ns_p:ThresholdIdType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="measurementThresholdRelationData"
type="ns_p:MeasurementThresholdRelationDataType"/>
    <xs:complexType name="MeasurementThresholdRelationDataElementsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="measurementId" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="thresholdId" type="ns_p:ElementTagType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="measurementThresholdRelationDataElements"
type="ns_p:MeasurementThresholdRelationDataElementsType"/>
    <xs:complexType name="MeasurementThresholdRelationListDataType">
        <xs:sequence>
            <xs:element maxOccurs="unbounded" minOccurs="0"
ref="ns_p:measurementThresholdRelationData"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="measurementThresholdRelationListData"
type="ns_p:MeasurementThresholdRelationListDataType"/>
    <xs:complexType name="MeasurementThresholdRelationListDataSelectorsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="measurementId" type="ns_p:MeasurementIdType"/>
            <xs:element minOccurs="0" name="thresholdId" type="ns_p:ThresholdIdType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="measurementThresholdRelationListDataSelectors"
type="ns_p:MeasurementThresholdRelationListDataSelectorsType"/>
</xs:schema>
```


A.2.16 Messaging

File name: EEBus_SPINE_TS_Messaging.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
    Smart Premises Interoperable Neutral-Message Exchange (SPINE)
    Version 1.1.1
    2018-12-21
    Copyright (c) 2018 EEBus Initiative e.V.  All Rights Reserved.
    Source: https://www.eebus.org/en/specifications/
-->
<xs:schema xmlns:ns_p="http://docs.eebus.org/spine/xsd/v1"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://docs.eebus.org/spine/xsd/v1" version="1.1.1"
    blockDefault="#all" elementFormDefault="qualified">
    <xs:annotation>
        <xs:documentation>EEBus SPINE Specification schema. Copyright 2018 EEBus Initiative
e.V.  All rights reserved.</xs:documentation>
    </xs:annotation>
    <xs:include schemaLocation="EEBus_SPINE_TS_CommonDataTypes.xsd"/>
    <xs:simpleType name="MessagingNumberType">
        <xs:restriction base="xs:unsignedInt"/>
    </xs:simpleType>
    <xs:simpleType name="MessagingTypeType">
        <xs:union memberTypes="ns_p:MessagingTypeEnumType ns_p:EnumExtendType"/>
    </xs:simpleType>
    <xs:simpleType name="MessagingDataTextType">
        <xs:restriction base="xs:string"/>
    </xs:simpleType>
    <xs:simpleType name="MessagingTypeEnumType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="logging"/>
            <xs:enumeration value="information"/>
            <xs:enumeration value="warning"/>
            <xs:enumeration value="alarm"/>
            <xs:enumeration value="emergency"/>
            <xs:enumeration value="obsolete"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:complexType name="MessagingDataType">
        <xs:sequence>
            <xs:element minOccurs="0" name="timestamp"
type="ns_p:AbsoluteOrRelativeTimeType"/>
            <xs:element minOccurs="0" name="messagingNumber" type="ns_p:MessagingNumberType"/>
            <xs:element minOccurs="0" name="type" type="ns_p:MessagingTypeType"/>
            <xs:element minOccurs="0" name="text" type="ns_p:MessagingDataTextType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="messagingData" type="ns_p:MessagingDataType"/>
    <xs:complexType name="MessagingDataElementsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="timestamp" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="messagingNumber" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="type" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="text" type="ns_p:ElementTagType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="messagingDataElements" type="ns_p:MessagingDataElementsType"/>
    <xs:complexType name="MessagingListDataType">
        <xs:sequence>
            <xs:element maxOccurs="unbounded" minOccurs="0" ref="ns_p:messagingData"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="messagingListData" type="ns_p:MessagingListDataType"/>
    <xs:complexType name="MessagingListDataSelectorsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="timestampInterval"
type="ns_p:TimestampIntervalType"/>
            <xs:element minOccurs="0" name="messagingNumber" type="ns_p:MessagingNumberType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="messagingListDataSelectors" type="ns_p:MessagingListDataSelectorsType"/>
</xs:schema>
```


A.2.17 NetworkManagement

File name: EEBus_SPINE_TS_NetworkManagement.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
    Smart Premises Interoperable Neutral-Message Exchange (SPINE)
    Version 1.1.1
    2018-12-21
    Copyright (c) 2018 EEBus Initiative e.V.  All Rights Reserved.
    Source: https://www.eebus.org/en/specifications/
-->
<xs:schema xmlns:ns_p="http://docs.eebus.org/spine/xsd/v1"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://docs.eebus.org/spine/xsd/v1" version="1.1.1"
    blockDefault="#all" elementFormDefault="qualified">
    <xs:annotation>
        <xs:documentation>EEBus SPINE Specification schema. Copyright 2018 EEBus Initiative
e.V.  All rights reserved.</xs:documentation>
    </xs:annotation>
    <xs:include schemaLocation="EEBus_SPINE_TS_CommonDataTypes.xsd"/>
    <xs:simpleType name="NetworkManagementNativeSetupType">
        <xs:restriction base="xs:string"/>
    </xs:simpleType>
    <xs:simpleType name="NetworkManagementScanSetupType">
        <xs:restriction base="xs:string"/>
    </xs:simpleType>
    <xs:simpleType name="NetworkManagementSetupType">
        <xs:restriction base="xs:string"/>
    </xs:simpleType>
    <xs:simpleType name="NetworkManagementCandidateSetupType">
        <xs:restriction base="xs:string"/>
    </xs:simpleType>
    <xs:simpleType name="NetworkManagementTechnologyAddressType">
        <xs:restriction base="xs:string"/>
    </xs:simpleType>
    <xs:simpleType name="NetworkManagementCommunicationsTechnologyInformationType">
        <xs:restriction base="xs:string"/>
    </xs:simpleType>
    <xs:simpleType name="NetworkManagementMinimumTrustLevelType">
        <xs:restriction base="xs:string"/>
    </xs:simpleType>
    <xs:simpleType name="NetworkManagementProcessTimeoutType">
        <xs:restriction base="xs:duration"/>
    </xs:simpleType>
    <xs:simpleType name="NetworkManagementFeatureSetType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="gateway"/>
            <xs:enumeration value="router"/>
            <xs:enumeration value="smart"/>
            <xs:enumeration value="simple"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="NetworkManagementProcessStateStateType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="succeeded"/>
            <xs:enumeration value="failed"/>
            <xs:enumeration value="aborted"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="NetworkManagementStateChangeType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="added"/>
            <xs:enumeration value="removed"/>
            <xs:enumeration value="modified"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:complexType name="NetworkManagementAddNodeCallType">
        <xs:sequence>
            <xs:element minOccurs="0" name="nodeAddress" type="ns_p:FeatureAddressType"/>
            <xs:element minOccurs="0" name="nativeSetup"
type="ns_p:NetworkManagementNativeSetupType"/>
            <xs:element minOccurs="0" name="timeout"
type="ns_p:NetworkManagementProcessTimeoutType"/>
```

```
16432             <xs:element minOccurs="0" name="label" type="ns_p:LabelType"/>
16433             <xs:element minOccurs="0" name="description" type="ns_p:DescriptionType"/>
16434         </xs:sequence>
16435     </xs:complexType>
16436     <xs:element name="networkManagementAddNodeCall"
16437 type="ns_p:NetworkManagementAddNodeCallType"/>
16438     <xs:complexType name="NetworkManagementAddNodeCallElementsType">
16439         <xs:sequence>
16440             <xs:element minOccurs="0" name="nodeAddress"
16441 type="ns_p:FeatureAddressElementsType"/>
16442             <xs:element minOccurs="0" name="nativeSetup" type="ns_p:ElementTagType"/>
16443             <xs:element minOccurs="0" name="timeout" type="ns_p:ElementTagType"/>
16444             <xs:element minOccurs="0" name="label" type="ns_p:ElementTagType"/>
16445             <xs:element minOccurs="0" name="description" type="ns_p:ElementTagType"/>
16446         </xs:sequence>
16447     </xs:complexType>
16448     <xs:element name="networkManagementAddNodeCallElements"
16449 type="ns_p:NetworkManagementAddNodeCallElementsType"/>
16450     <xs:complexType name="NetworkManagementRemoveNodeCallType">
16451         <xs:sequence>
16452             <xs:element minOccurs="0" name="nodeAddress" type="ns_p:FeatureAddressType"/>
16453             <xs:element minOccurs="0" name="timeout"
16454 type="ns_p:NetworkManagementProcessTimeoutType"/>
16455         </xs:sequence>
16456     </xs:complexType>
16457     <xs:element name="networkManagementRemoveNodeCall"
16458 type="ns_p:NetworkManagementRemoveNodeCallType"/>
16459     <xs:complexType name="NetworkManagementRemoveNodeCallElementsType">
16460         <xs:sequence>
16461             <xs:element minOccurs="0" name="nodeAddress"
16462 type="ns_p:FeatureAddressElementsType"/>
16463             <xs:element minOccurs="0" name="timeout" type="ns_p:ElementTagType"/>
16464         </xs:sequence>
16465     </xs:complexType>
16466     <xs:element name="networkManagementRemoveNodeCallElements"
16467 type="ns_p:NetworkManagementRemoveNodeCallElementsType"/>
16468     <xs:complexType name="NetworkManagementModifyNodeCallType">
16469         <xs:sequence>
16470             <xs:element minOccurs="0" name="nodeAddress" type="ns_p:FeatureAddressType"/>
16471             <xs:element minOccurs="0" name="nativeSetup"
16472 type="ns_p:NetworkManagementNativeSetupType"/>
16473             <xs:element minOccurs="0" name="timeout"
16474 type="ns_p:NetworkManagementProcessTimeoutType"/>
16475             <xs:element minOccurs="0" name="label" type="ns_p:LabelType"/>
16476             <xs:element minOccurs="0" name="description" type="ns_p:DescriptionType"/>
16477         </xs:sequence>
16478     </xs:complexType>
16479     <xs:element name="networkManagementModifyNodeCall"
16480 type="ns_p:NetworkManagementModifyNodeCallType"/>
16481     <xs:complexType name="NetworkManagementModifyNodeCallElementsType">
16482         <xs:sequence>
16483             <xs:element minOccurs="0" name="nodeAddress"
16484 type="ns_p:FeatureAddressElementsType"/>
16485             <xs:element minOccurs="0" name="nativeSetup" type="ns_p:ElementTagType"/>
16486             <xs:element minOccurs="0" name="timeout" type="ns_p:ElementTagType"/>
16487             <xs:element minOccurs="0" name="label" type="ns_p:ElementTagType"/>
16488             <xs:element minOccurs="0" name="description" type="ns_p:ElementTagType"/>
16489         </xs:sequence>
16490     </xs:complexType>
16491     <xs:element name="networkManagementModifyNodeCallElements"
16492 type="ns_p:NetworkManagementModifyNodeCallElementsType"/>
16493     <xs:complexType name="NetworkManagementScanNetworkCallType">
16494         <xs:sequence>
16495             <xs:element minOccurs="0" name="scanSetup"
16496 type="ns_p:NetworkManagementScanSetupType"/>
16497             <xs:element minOccurs="0" name="timeout"
16498 type="ns_p:NetworkManagementProcessTimeoutType"/>
16499         </xs:sequence>
16500     </xs:complexType>
16501     <xs:element name="networkManagementScanNetworkCall"
16502 type="ns_p:NetworkManagementScanNetworkCallType"/>
16503     <xs:complexType name="NetworkManagementScanNetworkCallElementsType">
16504         <xs:sequence>
16505             <xs:element minOccurs="0" name="scanSetup" type="ns_p:ElementTagType"/>
16506             <xs:element minOccurs="0" name="timeout" type="ns_p:ElementTagType"/>
16507         </xs:sequence>
16508     </xs:complexType>
```



```
16509       <xs:element name="networkManagementScanNetworkCallElements"
16510    type="ns_p:NetworkManagementScanNetworkCallElementsType"/>
16511       <xs:complexType name="NetworkManagementDiscoverCallType">
16512           <xs:sequence>
16513               <xs:element minOccurs="0" name="discoverAddress" type="ns_p:FeatureAddressType"/>
16514           </xs:sequence>
16515       </xs:complexType>
16516       <xs:element name="networkManagementDiscoverCall"
16517    type="ns_p:NetworkManagementDiscoverCallType"/>
16518       <xs:complexType name="NetworkManagementDiscoverCallElementsType">
16519           <xs:sequence>
16520               <xs:element minOccurs="0" name="discoverAddress"
16521    type="ns_p:FeatureAddressElementsType"/>
16522           </xs:sequence>
16523       </xs:complexType>
16524       <xs:element name="networkManagementDiscoverCallElements"
16525    type="ns_p:NetworkManagementDiscoverCallElementsType"/>
16526       <xs:complexType name="NetworkManagementAbortCallType"/>
16527       <xs:element name="networkManagementAbortCall" type="ns_p:NetworkManagementAbortCallType"/>
16528       <xs:complexType name="NetworkManagementAbortCallElementsType"/>
16529       <xs:element name="networkManagementAbortCallElements"
16530    type="ns_p:NetworkManagementAbortCallElementsType"/>
16531       <xs:complexType name="NetworkManagementProcessStateDataType">
16532           <xs:sequence>
16533               <xs:element minOccurs="0" name="state"
16534    type="ns_p:NetworkManagementProcessStateStateType"/>
16535               <xs:element minOccurs="0" name="description" type="ns_p:DescriptionType"/>
16536           </xs:sequence>
16537       </xs:complexType>
16538       <xs:element name="networkManagementProcessStateData"
16539    type="ns_p:NetworkManagementProcessStateDataType"/>
16540       <xs:complexType name="NetworkManagementProcessStateDataElementsType">
16541           <xs:sequence>
16542               <xs:element minOccurs="0" name="state" type="ns_p:ElementTagType"/>
16543               <xs:element minOccurs="0" name="description" type="ns_p:ElementTagType"/>
16544           </xs:sequence>
16545       </xs:complexType>
16546       <xs:element name="networkManagementProcessStateDataElements"
16547    type="ns_p:NetworkManagementProcessStateDataElementsType"/>
16548       <xs:complexType name="NetworkManagementJoiningModeDataType">
16549           <xs:sequence>
16550               <xs:element minOccurs="0" name="setup" type="ns_p:NetworkManagementSetupType"/>
16551           </xs:sequence>
16552       </xs:complexType>
16553       <xs:element name="networkManagementJoiningModeData"
16554    type="ns_p:NetworkManagementJoiningModeDataType"/>
16555       <xs:complexType name="NetworkManagementJoiningModeDataElementsType">
16556           <xs:sequence>
16557               <xs:element minOccurs="0" name="setup" type="ns_p:ElementTagType"/>
16558           </xs:sequence>
16559       </xs:complexType>
16560       <xs:element name="networkManagementJoiningModeDataElements"
16561    type="ns_p:NetworkManagementJoiningModeDataElementsType"/>
16562       <xs:complexType name="NetworkManagementReportCandidateDataType">
16563           <xs:sequence>
16564               <xs:element minOccurs="0" name="candidateSetup"
16565    type="ns_p:NetworkManagementCandidateSetupType"/>
16566               <xs:element minOccurs="0" name="setupUsableForAdd" type="xs:boolean"/>
16567               <xs:element minOccurs="0" name="label" type="ns_p:LabelType"/>
16568               <xs:element minOccurs="0" name="description" type="ns_p:DescriptionType"/>
16569           </xs:sequence>
16570       </xs:complexType>
16571       <xs:element name="networkManagementReportCandidateData"
16572    type="ns_p:NetworkManagementReportCandidateDataType"/>
16573       <xs:complexType name="NetworkManagementReportCandidateDataElementsType">
16574           <xs:sequence>
16575               <xs:element minOccurs="0" name="candidateSetup" type="ns_p:ElementTagType"/>
16576               <xs:element minOccurs="0" name="setupUsableForAdd" type="ns_p:ElementTagType"/>
16577               <xs:element minOccurs="0" name="label" type="ns_p:ElementTagType"/>
16578               <xs:element minOccurs="0" name="description" type="ns_p:ElementTagType"/>
16579           </xs:sequence>
16580       </xs:complexType>
16581       <xs:element name="networkManagementReportCandidateDataElements"
16582    type="ns_p:NetworkManagementReportCandidateDataElementsType"/>
16583       <xs:complexType name="NetworkManagementDeviceDescriptionDataType">
16584           <xs:sequence>
16585               <xs:element minOccurs="0" name="deviceAddress" type="ns_p:DeviceAddressType"/>
16586               <xs:element minOccurs="0" name="deviceType" type="ns_p:DeviceTypeType"/>
```



```
16587             <xs:element minOccurs="0" name="networkManagementResponsibleAddress"
16588 type="ns_p:FeatureAddressType"/>
16589             <xs:element minOccurs="0" name="nativeSetup"
16590 type="ns_p:NetworkManagementNativeSetupType"/>
16591             <xs:element minOccurs="0" name="technologyAddress"
16592 type="ns_p:NetworkManagementTechnologyAddressType"/>
16593             <xs:element minOccurs="0" name="communicationsTechnologyInformation"
16594 type="ns_p:NetworkManagementCommunicationsTechnologyInformationType"/>
16595             <xs:element minOccurs="0" name="networkFeatureSet"
16596 type="ns_p:NetworkManagementFeatureSetType"/>
16597             <xs:element minOccurs="0" name="lastStateChange"
16598 type="ns_p:NetworkManagementStateChangeType"/>
16599             <xs:element minOccurs="0" name="minimumTrustLevel"
16600 type="ns_p:NetworkManagementMinimumTrustLevelType"/>
16601             <xs:element minOccurs="0" name="label" type="ns_p:LabelType"/>
16602             <xs:element minOccurs="0" name="description" type="ns_p:DescriptionType"/>
16603         </xs:sequence>
16604     </xs:complexType>
16605     <xs:element name="networkManagementDeviceDescriptionData"
16606 type="ns_p:NetworkManagementDeviceDescriptionDataType"/>
16607     <xs:complexType name="NetworkManagementDeviceDescriptionDataElementsType">
16608         <xs:sequence>
16609             <xs:element minOccurs="0" name="deviceAddress"
16610 type="ns_p:DeviceAddressElementsType"/>
16611             <xs:element minOccurs="0" name="deviceType" type="ns_p:ElementTagType"/>
16612             <xs:element minOccurs="0" name="networkManagementResponsibleAddress"
16613 type="ns_p:ElementTagType"/>
16614             <xs:element minOccurs="0" name="nativeSetup" type="ns_p:ElementTagType"/>
16615             <xs:element minOccurs="0" name="technologyAddress" type="ns_p:ElementTagType"/>
16616             <xs:element minOccurs="0" name="communicationsTechnologyInformation"
16617 type="ns_p:ElementTagType"/>
16618             <xs:element minOccurs="0" name="networkFeatureSet" type="ns_p:ElementTagType"/>
16619             <xs:element minOccurs="0" name="lastStateChange" type="ns_p:ElementTagType"/>
16620             <xs:element minOccurs="0" name="minimumTrustLevel" type="ns_p:ElementTagType"/>
16621             <xs:element minOccurs="0" name="label" type="ns_p:ElementTagType"/>
16622             <xs:element minOccurs="0" name="description" type="ns_p:ElementTagType"/>
16623         </xs:sequence>
16624     </xs:complexType>
16625     <xs:element name="networkManagementDeviceDescriptionDataElements"
16626 type="ns_p:NetworkManagementDeviceDescriptionDataElementsType"/>
16627     <xs:complexType name="NetworkManagementDeviceDescriptionListDataType">
16628         <xs:sequence>
16629             <xs:element maxOccurs="unbounded" minOccurs="0"
16630 ref="ns_p:networkManagementDeviceDescriptionData"/>
16631         </xs:sequence>
16632     </xs:complexType>
16633     <xs:element name="networkManagementDeviceDescriptionListData"
16634 type="ns_p:NetworkManagementDeviceDescriptionListDataType"/>
16635     <xs:complexType name="NetworkManagementDeviceDescriptionListDataSelectorsType">
16636         <xs:sequence>
16637             <xs:element minOccurs="0" name="deviceAddress" type="ns_p:DeviceAddressType"/>
16638             <xs:element minOccurs="0" name="deviceType" type="ns_p:DeviceTypeType"/>
16639         </xs:sequence>
16640     </xs:complexType>
16641     <xs:element name="networkManagementDeviceDescriptionListDataSelectors"
16642 type="ns_p:NetworkManagementDeviceDescriptionListDataSelectorsType"/>
16643     <xs:complexType name="NetworkManagementEntityDescriptionDataType">
16644         <xs:sequence>
16645             <xs:element minOccurs="0" name="entityAddress" type="ns_p:EntityAddressType"/>
16646             <xs:element minOccurs="0" name="entityType" type="ns_p:EntityTypeType"/>
16647             <xs:element minOccurs="0" name="lastStateChange"
16648 type="ns_p:NetworkManagementStateChangeType"/>
16649             <xs:element minOccurs="0" name="minimumTrustLevel"
16650 type="ns_p:NetworkManagementMinimumTrustLevelType"/>
16651             <xs:element minOccurs="0" name="label" type="ns_p:LabelType"/>
16652             <xs:element minOccurs="0" name="description" type="ns_p:DescriptionType"/>
16653         </xs:sequence>
16654     </xs:complexType>
16655     <xs:element name="networkManagementEntityDescriptionData"
16656 type="ns_p:NetworkManagementEntityDescriptionDataType"/>
16657     <xs:complexType name="NetworkManagementEntityDescriptionDataElementsType">
16658         <xs:sequence>
16659             <xs:element minOccurs="0" name="entityAddress"
16660 type="ns_p:EntityAddressElementsType"/>
16661             <xs:element minOccurs="0" name="entityType" type="ns_p:ElementTagType"/>
16662             <xs:element minOccurs="0" name="lastStateChange" type="ns_p:ElementTagType"/>
16663             <xs:element minOccurs="0" name="minimumTrustLevel" type="ns_p:ElementTagType"/>
16664             <xs:element minOccurs="0" name="label" type="ns_p:ElementTagType"/>
```



```
            <xs:element minOccurs="0" name="description" type="ns_p:ElementTagType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="networkManagementEntityDescriptionDataElements"
type="ns_p:NetworkManagementEntityDescriptionDataElementsType"/>
    <xs:complexType name="NetworkManagementEntityDescriptionListDataType">
        <xs:sequence>
            <xs:element maxOccurs="unbounded" minOccurs="0"
ref="ns_p:networkManagementEntityDescriptionData"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="networkManagementEntityDescriptionListData"
type="ns_p:NetworkManagementEntityDescriptionListDataType"/>
    <xs:complexType name="NetworkManagementEntityDescriptionListDataSelectorsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="entityAddress" type="ns_p:EntityAddressType"/>
            <xs:element minOccurs="0" name="entityType" type="ns_p:EntityTypeType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="networkManagementEntityDescriptionListDataSelectors"
type="ns_p:NetworkManagementEntityDescriptionListDataSelectorsType"/>
    <xs:complexType name="NetworkManagementFeatureDescriptionDataType">
        <xs:sequence>
            <xs:element minOccurs="0" name="featureAddress" type="ns_p:FeatureAddressType"/>
            <xs:element minOccurs="0" name="featureType" type="ns_p:FeatureTypeType"/>
            <xs:element maxOccurs="unbounded" minOccurs="0" name="specificUsage"
type="ns_p:FeatureSpecificUsageType"/>
            <xs:element minOccurs="0" name="featureGroup" type="ns_p:FeatureGroupType"/>
            <xs:element minOccurs="0" name="role" type="ns_p:RoleType"/>
            <xs:element maxOccurs="unbounded" minOccurs="0" name="supportedFunction"
type="ns_p:FunctionPropertyType"/>
            <xs:element minOccurs="0" name="lastStateChange"
type="ns_p:NetworkManagementStateChangeType"/>
            <xs:element minOccurs="0" name="minimumTrustLevel"
type="ns_p:NetworkManagementMinimumTrustLevelType"/>
            <xs:element minOccurs="0" name="label" type="ns_p:LabelType"/>
            <xs:element minOccurs="0" name="description" type="ns_p:DescriptionType"/>
            <xs:element minOccurs="0" name="maxResponseDelay"
type="ns_p:MaxResponseDelayType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="networkManagementFeatureDescriptionData"
type="ns_p:NetworkManagementFeatureDescriptionDataType"/>
    <xs:complexType name="NetworkManagementFeatureDescriptionDataElementsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="featureAddress"
type="ns_p:FeatureAddressElementsType"/>
            <xs:element minOccurs="0" name="featureType" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="specificUsage" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="featureGroup" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="role" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="supportedFunction"
type="ns_p:FunctionPropertyElementsType"/>
            <xs:element minOccurs="0" name="lastStateChange" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="minimumTrustLevel" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="label" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="description" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="maxResponseDelay" type="ns_p:ElementTagType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="networkManagementFeatureDescriptionDataElements"
type="ns_p:NetworkManagementFeatureDescriptionDataElementsType"/>
    <xs:complexType name="NetworkManagementFeatureDescriptionListDataType">
        <xs:sequence>
            <xs:element maxOccurs="unbounded" minOccurs="0"
ref="ns_p:networkManagementFeatureDescriptionData"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="networkManagementFeatureDescriptionListData"
type="ns_p:NetworkManagementFeatureDescriptionListDataType"/>
    <xs:complexType name="NetworkManagementFeatureDescriptionListDataSelectorsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="featureAddress" type="ns_p:FeatureAddressType"/>
            <xs:element minOccurs="0" name="featureType" type="ns_p:FeatureTypeType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="networkManagementFeatureDescriptionListDataSelectors"
type="ns_p:NetworkManagementFeatureDescriptionListDataSelectorsType"/>
```



```
</xs:schema>
```

A.2.18 OperatingConstraints

File name: EEBus_SPINE_TS_OperatingConstraints.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
    Smart Premises Interoperable Neutral-Message Exchange (SPINE)
    Version 1.1.1
    2018-12-21
    Copyright (c) 2018 EEBus Initiative e.V.  All Rights Reserved.
    Source: https://www.eebus.org/en/specifications/
-->
<xs:schema xmlns:ns_p="http://docs.eebus.org/spine/xsd/v1"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://docs.eebus.org/spine/xsd/v1" version="1.1.1"
    blockDefault="#all" elementFormDefault="qualified">
    <xs:annotation>
        <xs:documentation>EEBus SPINE Specification schema. Copyright 2018 EEBus Initiative
e.V.  All rights reserved.</xs:documentation>
    </xs:annotation>
    <xs:include schemaLocation="EEBus_SPINE_TS_PowerSequences.xsd"/>
    <xs:complexType name="OperatingConstraintsInterruptDataType">
        <xs:sequence>
            <xs:element minOccurs="0" name="sequenceId" type="ns_p:PowerSequenceIdType"/>
            <xs:element name="isPausable" type="xs:boolean" minOccurs="0"/>
            <xs:element name="isStoppable" type="xs:boolean" minOccurs="0"/>
            <xs:element name="notInterruptibleAtHighPower" type="xs:boolean" minOccurs="0"/>
            <xs:element name="maxCyclesPerDay" type="xs:unsignedInt" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="operatingConstraintsInterruptData"
type="ns_p:OperatingConstraintsInterruptDataType"/>
    <xs:complexType name="OperatingConstraintsInterruptDataElementsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="sequenceId" type="ns_p:ElementTagType"/>
            <xs:element name="isPausable" minOccurs="0" type="ns_p:ElementTagType"/>
            <xs:element name="isStoppable" minOccurs="0" type="ns_p:ElementTagType"/>
            <xs:element name="notInterruptibleAtHighPower" minOccurs="0"
type="ns_p:ElementTagType"/>
            <xs:element name="maxCyclesPerDay" minOccurs="0" type="ns_p:ElementTagType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="operatingConstraintsInterruptDataElements"
type="ns_p:OperatingConstraintsInterruptDataElementsType"/>
    <xs:complexType name="OperatingConstraintsInterruptListDataType">
        <xs:sequence>
            <xs:element maxOccurs="unbounded" minOccurs="0"
ref="ns_p:operatingConstraintsInterruptData"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="operatingConstraintsInterruptListData"
type="ns_p:OperatingConstraintsInterruptListDataType"/>
    <xs:complexType name="OperatingConstraintsInterruptListDataSelectorsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="sequenceId" type="ns_p:PowerSequenceIdType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="operatingConstraintsInterruptListDataSelectors"
type="ns_p:OperatingConstraintsInterruptListDataSelectorsType"/>
    <xs:complexType name="OperatingConstraintsDurationDataType">
        <xs:sequence>
            <xs:element minOccurs="0" name="sequenceId" type="ns_p:PowerSequenceIdType"/>
            <xs:element name="activeDurationMin" type="xs:duration" minOccurs="0"/>
            <xs:element name="activeDurationMax" type="xs:duration" minOccurs="0"/>
            <xs:element name="pauseDurationMin" type="xs:duration" minOccurs="0"/>
            <xs:element name="pauseDurationMax" type="xs:duration" minOccurs="0"/>
            <xs:element name="activeDurationSumMin" type="xs:duration" minOccurs="0"/>
            <xs:element name="activeDurationSumMax" type="xs:duration" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>
```

```
16816       <xs:element name="operatingConstraintsDurationData"
16817   type="ns_p:OperatingConstraintsDurationDataType"/>
16818       <xs:complexType name="OperatingConstraintsDurationDataElementsType">
16819           <xs:sequence>
16820               <xs:element minOccurs="0" name="sequenceId" type="ns_p:ElementTagType"/>
16821               <xs:element name="activeDurationMin" minOccurs="0" type="ns_p:ElementTagType"/>
16822               <xs:element name="activeDurationMax" minOccurs="0" type="ns_p:ElementTagType"/>
16823               <xs:element name="pauseDurationMin" minOccurs="0" type="ns_p:ElementTagType"/>
16824               <xs:element name="pauseDurationMax" minOccurs="0" type="ns_p:ElementTagType"/>
16825               <xs:element name="activeDurationSumMin" minOccurs="0" type="ns_p:ElementTagType"/>
16826               <xs:element name="activeDurationSumMax" minOccurs="0" type="ns_p:ElementTagType"/>
16827           </xs:sequence>
16828       </xs:complexType>
16829       <xs:element name="operatingConstraintsDurationDataElements"
16830   type="ns_p:OperatingConstraintsDurationDataElementsType"/>
16831       <xs:complexType name="OperatingConstraintsDurationListDataType">
16832           <xs:sequence>
16833               <xs:element maxOccurs="unbounded" minOccurs="0"
16834   ref="ns_p:operatingConstraintsDurationData"/>
16835           </xs:sequence>
16836       </xs:complexType>
16837       <xs:element name="operatingConstraintsDurationListData"
16838   type="ns_p:OperatingConstraintsDurationListDataType"/>
16839       <xs:complexType name="OperatingConstraintsDurationListDataSelectorsType">
16840           <xs:sequence>
16841               <xs:element minOccurs="0" name="sequenceId" type="ns_p:PowerSequenceIdType"/>
16842           </xs:sequence>
16843       </xs:complexType>
16844       <xs:element name="operatingConstraintsDurationListDataSelectors"
16845   type="ns_p:OperatingConstraintsDurationListDataSelectorsType"/>
16846       <xs:complexType name="OperatingConstraintsPowerDescriptionDataType">
16847           <xs:sequence>
16848               <xs:element minOccurs="0" name="sequenceId" type="ns_p:PowerSequenceIdType"/>
16849               <xs:element minOccurs="0" name="positiveEnergyDirection"
16850   type="ns_p:EnergyDirectionType"/>
16851               <xs:element minOccurs="0" name="powerUnit" type="ns_p:UnitOfMeasurementType"/>
16852               <xs:element minOccurs="0" name="energyUnit" type="ns_p:UnitOfMeasurementType"/>
16853               <xs:element minOccurs="0" name="description" type="ns_p:DescriptionType"/>
16854           </xs:sequence>
16855       </xs:complexType>
16856       <xs:element name="operatingConstraintsPowerDescriptionData"
16857   type="ns_p:OperatingConstraintsPowerDescriptionDataType"/>
16858       <xs:complexType name="OperatingConstraintsPowerDescriptionDataElementsType">
16859           <xs:sequence>
16860               <xs:element minOccurs="0" name="sequenceId" type="ns_p:ElementTagType"/>
16861               <xs:element minOccurs="0" name="positiveEnergyDirection"
16862   type="ns_p:ElementTagType"/>
16863               <xs:element minOccurs="0" name="powerUnit" type="ns_p:ElementTagType"/>
16864               <xs:element minOccurs="0" name="energyUnit" type="ns_p:ElementTagType"/>
16865               <xs:element minOccurs="0" name="description" type="ns_p:ElementTagType"/>
16866           </xs:sequence>
16867       </xs:complexType>
16868       <xs:element name="operatingConstraintsPowerDescriptionDataElements"
16869   type="ns_p:OperatingConstraintsPowerDescriptionDataElementsType"/>
16870       <xs:complexType name="OperatingConstraintsPowerDescriptionListDataType">
16871           <xs:sequence>
16872               <xs:element maxOccurs="unbounded" minOccurs="0"
16873   ref="ns_p:operatingConstraintsPowerDescriptionData"/>
16874           </xs:sequence>
16875       </xs:complexType>
16876       <xs:element name="operatingConstraintsPowerDescriptionListData"
16877   type="ns_p:OperatingConstraintsPowerDescriptionListDataType"/>
16878       <xs:complexType name="OperatingConstraintsPowerDescriptionListDataSelectorsType">
16879           <xs:sequence>
16880               <xs:element minOccurs="0" name="sequenceId" type="ns_p:PowerSequenceIdType"/>
16881           </xs:sequence>
16882       </xs:complexType>
16883       <xs:element name="operatingConstraintsPowerDescriptionListDataSelectors"
16884   type="ns_p:OperatingConstraintsPowerDescriptionListDataSelectorsType"/>
16885       <xs:complexType name="OperatingConstraintsPowerRangeDataType">
16886           <xs:sequence>
16887               <xs:element minOccurs="0" name="sequenceId" type="ns_p:PowerSequenceIdType"/>
16888               <xs:element name="powerMin" type="ns_p:ScaledNumberType" minOccurs="0"/>
16889               <xs:element name="powerMax" type="ns_p:ScaledNumberType" minOccurs="0"/>
16890               <xs:element name="energyMin" type="ns_p:ScaledNumberType" minOccurs="0"/>
16891               <xs:element name="energyMax" type="ns_p:ScaledNumberType" minOccurs="0"/>
16892           </xs:sequence>
16893       </xs:complexType>
```



```
16894       <xs:element name="operatingConstraintsPowerRangeData"
16895   type="ns_p:OperatingConstraintsPowerRangeDataType"/>
16896       <xs:complexType name="OperatingConstraintsPowerRangeDataElementsType">
16897           <xs:sequence>
16898               <xs:element minOccurs="0" name="sequenceId" type="ns_p:ElementTagType"/>
16899               <xs:element name="powerMin" minOccurs="0" type="ns_p:ScaledNumberElementsType"/>
16900               <xs:element name="powerMax" minOccurs="0" type="ns_p:ScaledNumberElementsType"/>
16901               <xs:element name="energyMin" minOccurs="0" type="ns_p:ScaledNumberElementsType"/>
16902               <xs:element name="energyMax" minOccurs="0" type="ns_p:ScaledNumberElementsType"/>
16903           </xs:sequence>
16904       </xs:complexType>
16905       <xs:element name="operatingConstraintsPowerRangeDataElements"
16906   type="ns_p:OperatingConstraintsPowerRangeDataElementsType"/>
16907       <xs:complexType name="OperatingConstraintsPowerRangeListDataType">
16908           <xs:sequence>
16909               <xs:element maxOccurs="unbounded" minOccurs="0"
16910   ref="ns_p:operatingConstraintsPowerRangeData"/>
16911           </xs:sequence>
16912       </xs:complexType>
16913       <xs:element name="operatingConstraintsPowerRangeListData"
16914   type="ns_p:OperatingConstraintsPowerRangeListDataType"/>
16915       <xs:complexType name="OperatingConstraintsPowerRangeListDataSelectorsType">
16916           <xs:sequence>
16917               <xs:element minOccurs="0" name="sequenceId" type="ns_p:PowerSequenceIdType"/>
16918           </xs:sequence>
16919       </xs:complexType>
16920       <xs:element name="operatingConstraintsPowerRangeListDataSelectors"
16921   type="ns_p:OperatingConstraintsPowerRangeListDataSelectorsType"/>
16922       <xs:complexType name="OperatingConstraintsPowerLevelDataType">
16923           <xs:sequence>
16924               <xs:element minOccurs="0" name="sequenceId" type="ns_p:PowerSequenceIdType"/>
16925               <xs:element name="power" type="ns_p:ScaledNumberType" minOccurs="0"
16926   maxOccurs="unbounded"/>
16927           </xs:sequence>
16928       </xs:complexType>
16929       <xs:element name="operatingConstraintsPowerLevelData"
16930   type="ns_p:OperatingConstraintsPowerLevelDataType"/>
16931       <xs:complexType name="OperatingConstraintsPowerLevelDataElementsType">
16932           <xs:sequence>
16933               <xs:element minOccurs="0" name="sequenceId" type="ns_p:ElementTagType"/>
16934               <xs:element name="power" minOccurs="0" type="ns_p:ScaledNumberElementsType"/>
16935           </xs:sequence>
16936       </xs:complexType>
16937       <xs:element name="operatingConstraintsPowerLevelDataElements"
16938   type="ns_p:OperatingConstraintsPowerLevelDataElementsType"/>
16939       <xs:complexType name="OperatingConstraintsPowerLevelListDataType">
16940           <xs:sequence>
16941               <xs:element maxOccurs="unbounded" minOccurs="0"
16942   ref="ns_p:operatingConstraintsPowerLevelData"/>
16943           </xs:sequence>
16944       </xs:complexType>
16945       <xs:element name="operatingConstraintsPowerLevelListData"
16946   type="ns_p:OperatingConstraintsPowerLevelListDataType"/>
16947       <xs:complexType name="OperatingConstraintsPowerLevelListDataSelectorsType">
16948           <xs:sequence>
16949               <xs:element minOccurs="0" name="sequenceId" type="ns_p:PowerSequenceIdType"/>
16950           </xs:sequence>
16951       </xs:complexType>
16952       <xs:element name="operatingConstraintsPowerLevelListDataSelectors"
16953   type="ns_p:OperatingConstraintsPowerLevelListDataSelectorsType"/>
16954       <xs:complexType name="OperatingConstraintsResumeImplicationDataType">
16955           <xs:sequence>
16956               <xs:element minOccurs="0" name="sequenceId" type="ns_p:PowerSequenceIdType"/>
16957               <xs:element name="resumeEnergyEstimated" type="ns_p:ScaledNumberType"
16958   minOccurs="0"/>
16959               <xs:element name="energyUnit" minOccurs="0" type="ns_p:UnitOfMeasurementType"/>
16960               <xs:element name="resumeCostEstimated" type="ns_p:ScaledNumberType"
16961   minOccurs="0"/>
16962               <xs:element minOccurs="0" name="currency" type="ns_p:CurrencyType"/>
16963           </xs:sequence>
16964       </xs:complexType>
16965       <xs:element name="operatingConstraintsResumeImplicationData"
16966   type="ns_p:OperatingConstraintsResumeImplicationDataType"/>
16967       <xs:complexType name="OperatingConstraintsResumeImplicationDataElementsType">
16968           <xs:sequence>
16969               <xs:element minOccurs="0" name="sequenceId" type="ns_p:ElementTagType"/>
16970               <xs:element name="resumeEnergyEstimated" minOccurs="0"
16971   type="ns_p:ScaledNumberElementsType"/>
```



```
            <xs:element name="energyUnit" minOccurs="0" type="ns_p:ElementTagType"/>
            <xs:element name="resumeCostEstimated" minOccurs="0"
type="ns_p:ScaledNumberElementsType"/>
            <xs:element minOccurs="0" name="currency" type="ns_p:ElementTagType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="operatingConstraintsResumeImplicationDataElements"
type="ns_p:OperatingConstraintsResumeImplicationDataElementsType"/>
    <xs:complexType name="OperatingConstraintsResumeImplicationListDataType">
        <xs:sequence>
            <xs:element maxOccurs="unbounded" minOccurs="0"
ref="ns_p:operatingConstraintsResumeImplicationData"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="operatingConstraintsResumeImplicationListData"
type="ns_p:OperatingConstraintsResumeImplicationListDataType"/>
    <xs:complexType name="OperatingConstraintsResumeImplicationListDataSelectorsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="sequenceId" type="ns_p:PowerSequenceIdType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="operatingConstraintsResumeImplicationListDataSelectors"
type="ns_p:OperatingConstraintsResumeImplicationListDataSelectorsType"/>
</xs:schema>
```

A.2.19 PowerSequences

File name: EEBus_SPINE_TS_PowerSequences.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
    Smart Premises Interoperable Neutral-Message Exchange (SPINE)
    Version 1.1.1
    2018-12-21
    Copyright (c) 2018 EEBus Initiative e.V.  All Rights Reserved.
    Source: https://www.eebus.org/en/specifications/
-->
<xs:schema xmlns:ns_p="http://docs.eebus.org/spine/xsd/v1"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://docs.eebus.org/spine/xsd/v1" version="1.1.1"
    blockDefault="#all" elementFormDefault="qualified">
    <xs:annotation>
        <xs:documentation>EEBus SPINE Specification schema. Copyright 2018 EEBus Initiative
e.V.  All rights reserved.</xs:documentation>
    </xs:annotation>
    <xs:include schemaLocation="EEBus_SPINE_TS_CommonDataTypes.xsd"/>
    <xs:include schemaLocation="EEBus_SPINE_TS_Measurement.xsd"/>
    <xs:simpleType name="AlternativesIdType">
        <xs:restriction base="xs:unsignedInt"/>
    </xs:simpleType>
    <xs:simpleType name="PowerSequenceIdType">
        <xs:restriction base="xs:unsignedInt"/>
    </xs:simpleType>
    <xs:simpleType name="PowerTimeSlotNumberType">
        <xs:restriction base="xs:unsignedInt"/>
    </xs:simpleType>
    <xs:simpleType name="PowerTimeSlotValueTypeType">
        <xs:union memberTypes="ns_p:PowerTimeSlotValueTypeEnumType ns_p:EnumExtendType"/>
    </xs:simpleType>
    <xs:simpleType name="PowerTimeSlotValueTypeEnumType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="power"/>
            <xs:enumeration value="powerMin"/>
            <xs:enumeration value="powerMax"/>
            <xs:enumeration value="powerExpectedValue"/>
            <xs:enumeration value="powerStandardDeviation"/>
            <xs:enumeration value="powerSkewness"/>
            <xs:enumeration value="energy"/>
            <xs:enumeration value="energyMin"/>
            <xs:enumeration value="energyMax"/>
            <xs:enumeration value="energyExpectedValue"/>
            <xs:enumeration value="energyStandardDeviation"/>
```

```
            <xs:enumeration value="energySkewness"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="PowerSequenceScopeType">
        <xs:union memberTypes="ns_p:PowerSequenceScopeEnumType ns_p:EnumExtendType"/>
    </xs:simpleType>
    <xs:simpleType name="PowerSequenceScopeEnumType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="forecast"/>
            <xs:enumeration value="measurement"/>
            <xs:enumeration value="recommendation"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="PowerSequenceStateType">
        <xs:union memberTypes="ns_p:PowerSequenceStateEnumType ns_p:EnumExtendType"/>
    </xs:simpleType>
    <xs:simpleType name="PowerSequenceStateEnumType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="running"/>
            <xs:enumeration value="paused"/>
            <xs:enumeration value="scheduled"/>
            <xs:enumeration value="scheduledPaused"/>
            <xs:enumeration value="pending"/>
            <xs:enumeration value="inactive"/>
            <xs:enumeration value="completed"/>
            <xs:enumeration value="invalid"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:complexType name="PowerTimeSlotScheduleDataType">
        <xs:sequence>
            <xs:element minOccurs="0" name="sequenceId" type="ns_p:PowerSequenceIdType"/>
            <xs:element minOccurs="0" name="slotNumber" type="ns_p:PowerTimeSlotNumberType"/>
            <xs:element minOccurs="0" name="timePeriod" type="ns_p:TimePeriodType"/>
            <xs:element minOccurs="0" name="defaultDuration" type="xs:duration"/>
            <xs:element minOccurs="0" name="durationUncertainty" type="xs:duration"/>
            <xs:element minOccurs="0" name="slotActivated" type="xs:boolean"/>
            <xs:element minOccurs="0" name="description" type="ns_p:DescriptionType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="powerTimeSlotScheduleData" type="ns_p:PowerTimeSlotScheduleDataType"/>
    <xs:complexType name="PowerTimeSlotScheduleDataElementsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="sequenceId" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="slotNumber" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="timePeriod" type="ns_p:TimePeriodElementsType"/>
            <xs:element minOccurs="0" name="defaultDuration" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="durationUncertainty" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="slotActivated" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="description" type="ns_p:ElementTagType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="powerTimeSlotScheduleDataElements"
type="ns_p:PowerTimeSlotScheduleDataElementsType"/>
    <xs:complexType name="PowerTimeSlotScheduleListDataType">
        <xs:sequence>
            <xs:element maxOccurs="unbounded" minOccurs="0"
ref="ns_p:powerTimeSlotScheduleData"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="powerTimeSlotScheduleListData"
type="ns_p:PowerTimeSlotScheduleListDataType"/>
    <xs:complexType name="PowerTimeSlotScheduleListDataSelectorsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="sequenceId" type="ns_p:PowerSequenceIdType"/>
            <xs:element minOccurs="0" name="slotNumber" type="ns_p:PowerTimeSlotNumberType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="powerTimeSlotScheduleListDataSelectors"
type="ns_p:PowerTimeSlotScheduleListDataSelectorsType"/>
    <xs:complexType name="PowerTimeSlotValueDataType">
        <xs:sequence>
            <xs:element minOccurs="0" name="sequenceId" type="ns_p:PowerSequenceIdType"/>
            <xs:element minOccurs="0" name="slotNumber" type="ns_p:PowerTimeSlotNumberType"/>
            <xs:element minOccurs="0" name="valueType"
type="ns_p:PowerTimeSlotValueTypeType"/>
            <xs:element minOccurs="0" name="value" type="ns_p:ScaledNumberType"/>
        </xs:sequence>
    </xs:complexType>
```



```
17123       <xs:element name="powerTimeSlotValueData" type="ns_p:PowerTimeSlotValueDataType"/>
17124       <xs:complexType name="PowerTimeSlotValueDataElementsType">
17125           <xs:sequence>
17126               <xs:element minOccurs="0" name="sequenceId" type="ns_p:ElementTagType"/>
17127               <xs:element minOccurs="0" name="slotNumber" type="ns_p:ElementTagType"/>
17128               <xs:element minOccurs="0" name="valueType" type="ns_p:ElementTagType"/>
17129               <xs:element minOccurs="0" name="value" type="ns_p:ScaledNumberElementsType"/>
17130           </xs:sequence>
17131       </xs:complexType>
17132       <xs:element name="powerTimeSlotValueDataElements"
17133   type="ns_p:PowerTimeSlotValueDataElementsType"/>
17134       <xs:complexType name="PowerTimeSlotValueListDataType">
17135           <xs:sequence>
17136               <xs:element maxOccurs="unbounded" minOccurs="0"
17137   ref="ns_p:powerTimeSlotValueData"/>
17138           </xs:sequence>
17139       </xs:complexType>
17140       <xs:element name="powerTimeSlotValueListData" type="ns_p:PowerTimeSlotValueListDataType"/>
17141       <xs:complexType name="PowerTimeSlotValueListDataSelectorsType">
17142           <xs:sequence>
17143               <xs:element minOccurs="0" name="sequenceId" type="ns_p:PowerSequenceIdType"/>
17144               <xs:element minOccurs="0" name="slotNumber" type="ns_p:PowerTimeSlotNumberType"/>
17145               <xs:element minOccurs="0" name="valueType"
17146   type="ns_p:PowerTimeSlotValueTypeType"/>
17147           </xs:sequence>
17148       </xs:complexType>
17149       <xs:element name="powerTimeSlotValueListDataSelectors"
17150   type="ns_p:PowerTimeSlotValueListDataSelectorsType"/>
17151       <xs:complexType name="PowerTimeSlotScheduleConstraintsDataType">
17152           <xs:sequence>
17153               <xs:element minOccurs="0" name="sequenceId" type="ns_p:PowerSequenceIdType"/>
17154               <xs:element minOccurs="0" name="slotNumber" type="ns_p:PowerTimeSlotNumberType"/>
17155               <xs:element minOccurs="0" name="earliestStartTime"
17156   type="ns_p:AbsoluteOrRelativeTimeType"/>
17157               <xs:element minOccurs="0" name="latestEndTime"
17158   type="ns_p:AbsoluteOrRelativeTimeType"/>
17159               <xs:element minOccurs="0" name="minDuration" type="xs:duration"/>
17160               <xs:element minOccurs="0" name="maxDuration" type="xs:duration"/>
17161               <xs:element minOccurs="0" name="optionalSlot" type="xs:boolean"/>
17162           </xs:sequence>
17163       </xs:complexType>
17164       <xs:element name="powerTimeSlotScheduleConstraintsData"
17165   type="ns_p:PowerTimeSlotScheduleConstraintsDataType"/>
17166       <xs:complexType name="PowerTimeSlotScheduleConstraintsDataElementsType">
17167           <xs:sequence>
17168               <xs:element minOccurs="0" name="sequenceId" type="ns_p:ElementTagType"/>
17169               <xs:element minOccurs="0" name="slotNumber" type="ns_p:ElementTagType"/>
17170               <xs:element minOccurs="0" name="earliestStartTime" type="ns_p:ElementTagType"/>
17171               <xs:element minOccurs="0" name="latestEndTime" type="ns_p:ElementTagType"/>
17172               <xs:element minOccurs="0" name="minDuration" type="ns_p:ElementTagType"/>
17173               <xs:element minOccurs="0" name="maxDuration" type="ns_p:ElementTagType"/>
17174               <xs:element minOccurs="0" name="optionalSlot" type="ns_p:ElementTagType"/>
17175           </xs:sequence>
17176       </xs:complexType>
17177       <xs:element name="powerTimeSlotScheduleConstraintsDataElements"
17178   type="ns_p:PowerTimeSlotScheduleConstraintsDataElementsType"/>
17179       <xs:complexType name="PowerTimeSlotScheduleConstraintsListDataType">
17180           <xs:sequence>
17181               <xs:element maxOccurs="unbounded" minOccurs="0"
17182   ref="ns_p:powerTimeSlotScheduleConstraintsData"/>
17183           </xs:sequence>
17184       </xs:complexType>
17185       <xs:element name="powerTimeSlotScheduleConstraintsListData"
17186   type="ns_p:PowerTimeSlotScheduleConstraintsListDataType"/>
17187       <xs:complexType name="PowerTimeSlotScheduleConstraintsListDataSelectorsType">
17188           <xs:sequence>
17189               <xs:element minOccurs="0" name="sequenceId" type="ns_p:PowerSequenceIdType"/>
17190               <xs:element minOccurs="0" name="slotNumber" type="ns_p:PowerTimeSlotNumberType"/>
17191           </xs:sequence>
17192       </xs:complexType>
17193       <xs:element name="powerTimeSlotScheduleConstraintsListDataSelectors"
17194   type="ns_p:PowerTimeSlotScheduleConstraintsListDataSelectorsType"/>
17195       <xs:complexType name="PowerSequenceAlternativesRelationDataType">
17196           <xs:sequence>
17197               <xs:element minOccurs="0" name="alternativesId" type="ns_p:AlternativesIdType"/>
17198               <xs:element name="sequenceId" type="ns_p:PowerSequenceIdType" minOccurs="0"
17199   maxOccurs="unbounded"/>
17200           </xs:sequence>
```



```
17201       </xs:complexType>
17202       <xs:element name="powerSequenceAlternativesRelationData"
17203   type="ns_p:PowerSequenceAlternativesRelationDataType"/>
17204       <xs:complexType name="PowerSequenceAlternativesRelationDataElementsType">
17205           <xs:sequence>
17206               <xs:element minOccurs="0" name="alternativesId" type="ns_p:ElementTagType"/>
17207               <xs:element name="sequenceId" minOccurs="0" type="ns_p:ElementTagType"/>
17208           </xs:sequence>
17209       </xs:complexType>
17210       <xs:element name="powerSequenceAlternativesRelationDataElements"
17211   type="ns_p:PowerSequenceAlternativesRelationDataElementsType"/>
17212       <xs:complexType name="PowerSequenceAlternativesRelationListDataType">
17213           <xs:sequence>
17214               <xs:element maxOccurs="unbounded" minOccurs="0"
17215   ref="ns_p:powerSequenceAlternativesRelationData"/>
17216           </xs:sequence>
17217       </xs:complexType>
17218       <xs:element name="powerSequenceAlternativesRelationListData"
17219   type="ns_p:PowerSequenceAlternativesRelationListDataType"/>
17220       <xs:complexType name="PowerSequenceAlternativesRelationListDataSelectorsType">
17221           <xs:sequence>
17222               <xs:element minOccurs="0" name="alternativesId" type="ns_p:AlternativesIdType"/>
17223               <xs:element name="sequenceId" type="ns_p:PowerSequenceIdType" minOccurs="0"/>
17224           </xs:sequence>
17225       </xs:complexType>
17226       <xs:element name="powerSequenceAlternativesRelationListDataSelectors"
17227   type="ns_p:PowerSequenceAlternativesRelationListDataSelectorsType"/>
17228       <xs:complexType name="PowerSequenceDescriptionDataType">
17229           <xs:sequence>
17230               <xs:element name="sequenceId" type="ns_p:PowerSequenceIdType" minOccurs="0"/>
17231               <xs:element minOccurs="0" name="description" type="ns_p:DescriptionType"/>
17232               <xs:element minOccurs="0" name="positiveEnergyDirection"
17233   type="ns_p:EnergyDirectionType"/>
17234               <xs:element minOccurs="0" name="powerUnit" type="ns_p:UnitOfMeasurementType"/>
17235               <xs:element minOccurs="0" name="energyUnit" type="ns_p:UnitOfMeasurementType"/>
17236               <xs:element minOccurs="0" name="valueSource"
17237   type="ns_p:MeasurementValueSourceType"/>
17238               <xs:element minOccurs="0" name="scope" type="ns_p:PowerSequenceScopeType"/>
17239               <xs:element minOccurs="0" name="taskIdentifier" type="xs:unsignedInt"/>
17240               <xs:element minOccurs="0" name="repetitionsTotal" type="xs:unsignedInt"/>
17241           </xs:sequence>
17242       </xs:complexType>
17243       <xs:element name="powerSequenceDescriptionData"
17244   type="ns_p:PowerSequenceDescriptionDataType"/>
17245       <xs:complexType name="PowerSequenceDescriptionDataElementsType">
17246           <xs:sequence>
17247               <xs:element name="sequenceId" minOccurs="0" type="ns_p:ElementTagType"/>
17248               <xs:element minOccurs="0" name="description" type="ns_p:ElementTagType"/>
17249               <xs:element minOccurs="0" name="positiveEnergyDirection"
17250   type="ns_p:ElementTagType"/>
17251               <xs:element minOccurs="0" name="powerUnit" type="ns_p:ElementTagType"/>
17252               <xs:element minOccurs="0" name="energyUnit" type="ns_p:ElementTagType"/>
17253               <xs:element minOccurs="0" name="valueSource" type="ns_p:ElementTagType"/>
17254               <xs:element minOccurs="0" name="scope" type="ns_p:ElementTagType"/>
17255               <xs:element minOccurs="0" name="taskIdentifier" type="ns_p:ElementTagType"/>
17256               <xs:element minOccurs="0" name="repetitionsTotal" type="ns_p:ElementTagType"/>
17257           </xs:sequence>
17258       </xs:complexType>
17259       <xs:element name="powerSequenceDescriptionDataElements"
17260   type="ns_p:PowerSequenceDescriptionDataElementsType"/>
17261       <xs:complexType name="PowerSequenceDescriptionListDataType">
17262           <xs:sequence>
17263               <xs:element maxOccurs="unbounded" minOccurs="0"
17264   ref="ns_p:powerSequenceDescriptionData"/>
17265           </xs:sequence>
17266       </xs:complexType>
17267       <xs:element name="powerSequenceDescriptionListData"
17268   type="ns_p:PowerSequenceDescriptionListDataType"/>
17269       <xs:complexType name="PowerSequenceDescriptionListDataSelectorsType">
17270           <xs:sequence>
17271               <xs:element minOccurs="0" name="sequenceId" type="ns_p:PowerSequenceIdType"/>
17272           </xs:sequence>
17273       </xs:complexType>
17274       <xs:element name="powerSequenceDescriptionListDataSelectors"
17275   type="ns_p:PowerSequenceDescriptionListDataSelectorsType"/>
17276       <xs:complexType name="PowerSequenceStateDataType">
17277           <xs:sequence>
17278               <xs:element minOccurs="0" name="sequenceId" type="ns_p:PowerSequenceIdType"/>
```



```
            <xs:element minOccurs="0" name="state" type="ns_p:PowerSequenceStateType"/>
            <xs:element minOccurs="0" name="activeSlotNumber"
type="ns_p:PowerTimeSlotNumberType"/>
            <xs:element minOccurs="0" name="elapsedSlotTime" type="xs:duration"/>
            <xs:element minOccurs="0" name="remainingSlotTime" type="xs:duration"/>
            <xs:element minOccurs="0" name="sequenceRemoteControllable" type="xs:boolean"/>
            <xs:element minOccurs="0" name="activeRepetitionNumber" type="xs:unsignedInt"/>
            <xs:element minOccurs="0" name="remainingPauseTime" type="xs:duration"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="powerSequenceStateData" type="ns_p:PowerSequenceStateDataType"/>
    <xs:complexType name="PowerSequenceStateDataElementsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="sequenceId" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="state" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="activeSlotNumber" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="elapsedSlotTime" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="remainingSlotTime" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="sequenceRemoteControllable"
type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="activeRepetitionNumber"
type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="remainingPauseTime" type="ns_p:ElementTagType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="powerSequenceStateDataElements"
type="ns_p:PowerSequenceStateDataElementsType"/>
    <xs:complexType name="PowerSequenceStateListDataType">
        <xs:sequence>
            <xs:element maxOccurs="unbounded" minOccurs="0"
ref="ns_p:powerSequenceStateData"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="powerSequenceStateListData" type="ns_p:PowerSequenceStateListDataType"/>
    <xs:complexType name="PowerSequenceStateListDataSelectorsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="sequenceId" type="ns_p:PowerSequenceIdType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="powerSequenceStateListDataSelectors"
type="ns_p:PowerSequenceStateListDataSelectorsType"/>
    <xs:complexType name="PowerSequenceScheduleDataType">
        <xs:sequence>
            <xs:element minOccurs="0" name="sequenceId" type="ns_p:PowerSequenceIdType"/>
            <xs:element minOccurs="0" name="startTime"
type="ns_p:AbsoluteOrRelativeTimeType"/>
            <xs:element minOccurs="0" name="endTime" type="ns_p:AbsoluteOrRelativeTimeType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="powerSequenceScheduleData" type="ns_p:PowerSequenceScheduleDataType"/>
    <xs:complexType name="PowerSequenceScheduleDataElementsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="sequenceId" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="startTime" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="endTime" type="ns_p:ElementTagType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="powerSequenceScheduleDataElements"
type="ns_p:PowerSequenceScheduleDataElementsType"/>
    <xs:complexType name="PowerSequenceScheduleListDataType">
        <xs:sequence>
            <xs:element maxOccurs="unbounded" minOccurs="0"
ref="ns_p:powerSequenceScheduleData"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="powerSequenceScheduleListData"
type="ns_p:PowerSequenceScheduleListDataType"/>
    <xs:complexType name="PowerSequenceScheduleListDataSelectorsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="sequenceId" type="ns_p:PowerSequenceIdType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="powerSequenceScheduleListDataSelectors"
type="ns_p:PowerSequenceScheduleListDataSelectorsType"/>
    <xs:complexType name="PowerSequenceScheduleConstraintsDataType">
        <xs:sequence>
            <xs:element minOccurs="0" name="sequenceId" type="ns_p:PowerSequenceIdType"/>
```



```
            <xs:element minOccurs="0" name="earliestStartTime"
type="ns_p:AbsoluteOrRelativeTimeType"/>
            <xs:element minOccurs="0" name="latestStartTime"
type="ns_p:AbsoluteOrRelativeTimeType"/>
            <xs:element minOccurs="0" name="earliestEndTime"
type="ns_p:AbsoluteOrRelativeTimeType"/>
            <xs:element minOccurs="0" name="latestEndTime"
type="ns_p:AbsoluteOrRelativeTimeType"/>
            <xs:element minOccurs="0" name="optionalSequence" type="xs:boolean"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="powerSequenceScheduleConstraintsData"
type="ns_p:PowerSequenceScheduleConstraintsDataType"/>
    <xs:complexType name="PowerSequenceScheduleConstraintsDataElementsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="sequenceId" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="earliestStartTime" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="latestStartTime" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="earliestEndTime" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="latestEndTime" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="optionalSequence" type="ns_p:ElementTagType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="powerSequenceScheduleConstraintsDataElements"
type="ns_p:PowerSequenceScheduleConstraintsDataElementsType"/>
    <xs:complexType name="PowerSequenceScheduleConstraintsListDataType">
        <xs:sequence>
            <xs:element maxOccurs="unbounded" minOccurs="0"
ref="ns_p:powerSequenceScheduleConstraintsData"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="powerSequenceScheduleConstraintsListData"
type="ns_p:PowerSequenceScheduleConstraintsListDataType"/>
    <xs:complexType name="PowerSequenceScheduleConstraintsListDataSelectorsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="sequenceId" type="ns_p:PowerSequenceIdType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="powerSequenceScheduleConstraintsListDataSelectors"
type="ns_p:PowerSequenceScheduleConstraintsListDataSelectorsType"/>
    <xs:complexType name="PowerSequencePriceDataType">
        <xs:sequence>
            <xs:element minOccurs="0" name="sequenceId" type="ns_p:PowerSequenceIdType"/>
            <xs:element minOccurs="0" name="potentialStartTime"
type="ns_p:AbsoluteOrRelativeTimeType"/>
            <xs:element minOccurs="0" name="price" type="ns_p:ScaledNumberType"/>
            <xs:element minOccurs="0" name="currency" type="ns_p:CurrencyType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="powerSequencePriceData" type="ns_p:PowerSequencePriceDataType"/>
    <xs:complexType name="PowerSequencePriceDataElementsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="sequenceId" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="potentialStartTime" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="price" type="ns_p:ScaledNumberElementsType"/>
            <xs:element minOccurs="0" name="currency" type="ns_p:ElementTagType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="powerSequencePriceDataElements"
type="ns_p:PowerSequencePriceDataElementsType"/>
    <xs:complexType name="PowerSequencePriceListDataType">
        <xs:sequence>
            <xs:element maxOccurs="unbounded" minOccurs="0"
ref="ns_p:powerSequencePriceData"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="powerSequencePriceListData" type="ns_p:PowerSequencePriceListDataType"/>
    <xs:complexType name="PowerSequencePriceListDataSelectorsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="sequenceId" type="ns_p:PowerSequenceIdType"/>
            <xs:element minOccurs="0" name="potentialStartTimeInterval"
type="ns_p:TimestampIntervalType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="powerSequencePriceListDataSelectors"
type="ns_p:PowerSequencePriceListDataSelectorsType"/>
    <xs:complexType name="PowerSequenceSchedulePreferenceDataType">
        <xs:sequence>
```



```
            <xs:element minOccurs="0" name="sequenceId" type="ns_p:PowerSequenceIdType"/>
            <xs:element minOccurs="0" name="greenest" type="xs:boolean"/>
            <xs:element minOccurs="0" name="cheapest" type="xs:boolean"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="powerSequenceSchedulePreferenceData"
type="ns_p:PowerSequenceSchedulePreferenceDataType"/>
    <xs:complexType name="PowerSequenceSchedulePreferenceDataElementsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="sequenceId" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="greenest" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="cheapest" type="ns_p:ElementTagType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="powerSequenceSchedulePreferenceDataElements"
type="ns_p:PowerSequenceSchedulePreferenceDataElementsType"/>
    <xs:complexType name="PowerSequenceSchedulePreferenceListDataType">
        <xs:sequence>
            <xs:element maxOccurs="unbounded" minOccurs="0"
ref="ns_p:powerSequenceSchedulePreferenceData"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="powerSequenceSchedulePreferenceListData"
type="ns_p:PowerSequenceSchedulePreferenceListDataType"/>
    <xs:complexType name="PowerSequenceSchedulePreferenceListDataSelectorsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="sequenceId" type="ns_p:PowerSequenceIdType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="powerSequenceSchedulePreferenceListDataSelectors"
type="ns_p:PowerSequenceSchedulePreferenceListDataSelectorsType"/>
    <xs:complexType name="PowerSequenceNodeScheduleInformationDataType">
        <xs:sequence>
            <xs:element minOccurs="0" name="nodeRemoteControllable" type="xs:boolean"/>
            <xs:element minOccurs="0" name="supportsSingleSlotSchedulingOnly"
type="xs:boolean"/>
            <xs:element minOccurs="0" name="alternativesCount" type="xs:unsignedInt"/>
            <xs:element minOccurs="0" name="totalSequencesCountMax" type="xs:unsignedInt"/>
            <xs:element minOccurs="0" name="supportsReselection" type="xs:boolean"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="powerSequenceNodeScheduleInformationData"
type="ns_p:PowerSequenceNodeScheduleInformationDataType"/>
    <xs:complexType name="PowerSequenceNodeScheduleInformationDataElementsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="nodeRemoteControllable"
type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="supportsSingleSlotSchedulingOnly"
type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="alternativesCount" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="totalSequencesCountMax"
type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="supportsReselection" type="ns_p:ElementTagType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="powerSequenceNodeScheduleInformationDataElements"
type="ns_p:PowerSequenceNodeScheduleInformationDataElementsType"/>
    <xs:complexType name="PowerSequenceScheduleConfigurationRequestCallType">
        <xs:sequence>
            <xs:element minOccurs="0" name="sequenceId" type="ns_p:PowerSequenceIdType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="powerSequenceScheduleConfigurationRequestCall"
type="ns_p:PowerSequenceScheduleConfigurationRequestCallType"/>
    <xs:complexType name="PowerSequenceScheduleConfigurationRequestCallElementsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="sequenceId" type="ns_p:ElementTagType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="powerSequenceScheduleConfigurationRequestCallElements"
type="ns_p:PowerSequenceScheduleConfigurationRequestCallElementsType"/>
    <xs:complexType name="PowerSequencePriceCalculationRequestCallType">
        <xs:sequence>
            <xs:element minOccurs="0" name="sequenceId" type="ns_p:PowerSequenceIdType"/>
            <xs:element minOccurs="0" name="potentialStartTime"
type="ns_p:AbsoluteOrRelativeTimeType"/>
        </xs:sequence>
    </xs:complexType>
```



```
    <xs:element name="powerSequencePriceCalculationRequestCall"
type="ns_p:PowerSequencePriceCalculationRequestCallType"/>
    <xs:complexType name="PowerSequencePriceCalculationRequestCallElementsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="sequenceId" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="potentialStartTime" type="ns_p:ElementTagType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="powerSequencePriceCalculationRequestCallElements"
type="ns_p:PowerSequencePriceCalculationRequestCallElementsType"/>
</xs:schema>
```

A.2.20 Sensing

File name: EEBus_SPINE_TS_Sensing.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
    Smart Premises Interoperable Neutral-Message Exchange (SPINE)
    Version 1.1.1
    2018-12-21
    Copyright (c) 2018 EEBus Initiative e.V.  All Rights Reserved.
    Source: https://www.eebus.org/en/specifications/
-->
<xs:schema xmlns:ns_p="http://docs.eebus.org/spine/xsd/v1"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://docs.eebus.org/spine/xsd/v1" version="1.1.1"
    blockDefault="#all" elementFormDefault="qualified">
    <xs:annotation>
        <xs:documentation>EEBus SPINE Specification schema. Copyright 2018 EEBus Initiative
e.V.  All rights reserved.</xs:documentation>
    </xs:annotation>
    <xs:include schemaLocation="EEBus_SPINE_TS_CommonDataTypes.xsd"/>
    <xs:simpleType name="SensingStateType">
        <xs:union memberTypes="ns_p:SensingStateEnumType ns_p:EnumExtendType"/>
    </xs:simpleType>
    <xs:simpleType name="SensingStateEnumType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="on"/>
            <xs:enumeration value="off"/>
            <xs:enumeration value="toggle"/>
            <xs:enumeration value="level"/>
            <xs:enumeration value="levelUp"/>
            <xs:enumeration value="levelDown"/>
            <xs:enumeration value="levelStart"/>
            <xs:enumeration value="levelStop"/>
            <xs:enumeration value="levelAbsolute"/>
            <xs:enumeration value="levelRelative"/>
            <xs:enumeration value="levelPercentageAbsolute"/>
            <xs:enumeration value="levelPercentageRelative"/>
            <xs:enumeration value="pressed"/>
            <xs:enumeration value="longPressed"/>
            <xs:enumeration value="released"/>
            <xs:enumeration value="changed"/>
            <xs:enumeration value="started"/>
            <xs:enumeration value="stopped"/>
            <xs:enumeration value="paused"/>
            <xs:enumeration value="middle"/>
            <xs:enumeration value="up"/>
            <xs:enumeration value="down"/>
            <xs:enumeration value="forward"/>
            <xs:enumeration value="backwards"/>
            <xs:enumeration value="open"/>
            <xs:enumeration value="closed"/>
            <xs:enumeration value="opening"/>
            <xs:enumeration value="closing"/>
            <xs:enumeration value="high"/>
            <xs:enumeration value="low"/>
            <xs:enumeration value="day"/>
            <xs:enumeration value="night"/>
            <xs:enumeration value="detected"/>
            <xs:enumeration value="notDetected"/>
```

```
17585             <xs:enumeration value="alarmed"/>
17586             <xs:enumeration value="notAlarmed"/>
17587         </xs:restriction>
17588     </xs:simpleType>
17589     <xs:simpleType name="SensingTypeType">
17590         <xs:union memberTypes="ns_p:SensingTypeEnumType ns_p:EnumExtendType"/>
17591     </xs:simpleType>
17592     <xs:simpleType name="SensingTypeEnumType">
17593         <xs:restriction base="xs:string">
17594             <xs:enumeration value="switch"/>
17595             <xs:enumeration value="button"/>
17596             <xs:enumeration value="level"/>
17597             <xs:enumeration value="levelSwitch"/>
17598             <xs:enumeration value="windowHandle"/>
17599             <xs:enumeration value="contactSensor"/>
17600             <xs:enumeration value="occupancySensor"/>
17601             <xs:enumeration value="motionDetector"/>
17602             <xs:enumeration value="fireDetector"/>
17603             <xs:enumeration value="smokeDetector"/>
17604             <xs:enumeration value="heatDetector"/>
17605             <xs:enumeration value="waterDetector"/>
17606             <xs:enumeration value="gasDetector"/>
17607             <xs:enumeration value="alarmSensor"/>
17608             <xs:enumeration value="powerAlarmSensor"/>
17609             <xs:enumeration value="dayNightIndicator"/>
17610         </xs:restriction>
17611     </xs:simpleType>
17612     <xs:complexType name="SensingDataType">
17613         <xs:sequence>
17614             <xs:element minOccurs="0" name="timestamp"
17615 type="ns_p:AbsoluteOrRelativeTimeType"/>
17616             <xs:element minOccurs="0" name="state" type="ns_p:SensingStateType"/>
17617             <xs:element minOccurs="0" name="value" type="ns_p:ScaledNumberType"/>
17618         </xs:sequence>
17619     </xs:complexType>
17620     <xs:element name="sensingData" type="ns_p:SensingDataType"/>
17621     <xs:complexType name="SensingDataElementsType">
17622         <xs:sequence>
17623             <xs:element minOccurs="0" name="timestamp" type="ns_p:ElementTagType"/>
17624             <xs:element minOccurs="0" name="state" type="ns_p:ElementTagType"/>
17625             <xs:element minOccurs="0" name="value" type="ns_p:ScaledNumberElementsType"/>
17626         </xs:sequence>
17627     </xs:complexType>
17628     <xs:element name="sensingDataElements" type="ns_p:SensingDataElementsType"/>
17629     <xs:complexType name="SensingListDataType">
17630         <xs:sequence>
17631             <xs:element maxOccurs="unbounded" minOccurs="0" ref="ns_p:sensingData"/>
17632         </xs:sequence>
17633     </xs:complexType>
17634     <xs:element name="sensingListData" type="ns_p:SensingListDataType"/>
17635     <xs:complexType name="SensingListDataSelectorsType">
17636         <xs:sequence>
17637             <xs:element minOccurs="0" name="timestampInterval"
17638 type="ns_p:TimestampIntervalType"/>
17639         </xs:sequence>
17640     </xs:complexType>
17641     <xs:element name="sensingListDataSelectors" type="ns_p:SensingListDataSelectorsType"/>
17642     <xs:complexType name="SensingDescriptionDataType">
17643         <xs:sequence>
17644             <xs:element minOccurs="0" name="sensingType" type="ns_p:SensingTypeType"/>
17645             <xs:element minOccurs="0" name="unit" type="ns_p:UnitOfMeasurementType"/>
17646             <xs:element minOccurs="0" name="scopeType" type="ns_p:ScopeTypeType"/>
17647             <xs:element minOccurs="0" name="label" type="ns_p:LabelType"/>
17648             <xs:element minOccurs="0" name="description" type="ns_p:DescriptionType"/>
17649         </xs:sequence>
17650     </xs:complexType>
17651     <xs:element name="sensingDescriptionData" type="ns_p:SensingDescriptionDataType"/>
17652     <xs:complexType name="SensingDescriptionDataElementsType">
17653         <xs:sequence>
17654             <xs:element minOccurs="0" name="sensingType" type="ns_p:ElementTagType"/>
17655             <xs:element minOccurs="0" name="unit" type="ns_p:ElementTagType"/>
17656             <xs:element minOccurs="0" name="scopeType" type="ns_p:ElementTagType"/>
17657             <xs:element minOccurs="0" name="label" type="ns_p:ElementTagType"/>
17658             <xs:element minOccurs="0" name="description" type="ns_p:ElementTagType"/>
17659         </xs:sequence>
17660     </xs:complexType>
17661     <xs:element name="sensingDescriptionDataElements"
17662 type="ns_p:SensingDescriptionDataElementsType"/>
```



```
</xs:schema>
```

A.2.21 Setpoint

File name: EEBus_SPINE_TS_Setpoint.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
    Smart Premises Interoperable Neutral-Message Exchange (SPINE)
    Version 1.1.1
    2018-12-21
    Copyright (c) 2018 EEBus Initiative e.V.  All Rights Reserved.
    Source: https://www.eebus.org/en/specifications/
-->
<xs:schema xmlns:ns_p="http://docs.eebus.org/spine/xsd/v1"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://docs.eebus.org/spine/xsd/v1" version="1.1.1"
    blockDefault="#all" elementFormDefault="qualified">
    <xs:annotation>
        <xs:documentation>EEBus SPINE Specification schema. Copyright 2018 EEBus Initiative
e.V.  All rights reserved.</xs:documentation>
    </xs:annotation>
    <xs:include schemaLocation="EEBus_SPINE_TS_CommonDataTypes.xsd"/>
    <xs:include schemaLocation="EEBus_SPINE_TS_Measurement.xsd"/>
    <xs:include schemaLocation="EEBus_SPINE_TS_TimeTable.xsd"/>
    <xs:simpleType name="SetpointIdType">
        <xs:restriction base="xs:unsignedInt"/>
    </xs:simpleType>
    <xs:simpleType name="SetpointTypeType">
        <xs:union memberTypes="ns_p:SetpointTypeEnumType ns_p:EnumExtendType"/>
    </xs:simpleType>
    <xs:simpleType name="SetpointTypeEnumType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="valueAbsolute"/>
            <xs:enumeration value="valueRelative"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:complexType name="SetpointDataType">
        <xs:sequence>
            <xs:element minOccurs="0" name="setpointId" type="ns_p:SetpointIdType"/>
            <xs:element minOccurs="0" name="value" type="ns_p:ScaledNumberType"/>
            <xs:element minOccurs="0" name="valueMin" type="ns_p:ScaledNumberType"/>
            <xs:element minOccurs="0" name="valueMax" type="ns_p:ScaledNumberType"/>
            <xs:element minOccurs="0" name="valueToleranceAbsolute"
type="ns_p:ScaledNumberType"/>
            <xs:element minOccurs="0" name="valueTolerancePercentage"
type="ns_p:ScaledNumberType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="setpointData" type="ns_p:SetpointDataType"/>
    <xs:complexType name="SetpointDataElementsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="setpointId" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="value" type="ns_p:ScaledNumberElementsType"/>
            <xs:element minOccurs="0" name="valueMin" type="ns_p:ScaledNumberElementsType"/>
            <xs:element minOccurs="0" name="valueMax" type="ns_p:ScaledNumberElementsType"/>
            <xs:element minOccurs="0" name="valueToleranceAbsolute"
type="ns_p:ScaledNumberElementsType"/>
            <xs:element minOccurs="0" name="valueTolerancePercentage"
type="ns_p:ScaledNumberElementsType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="setpointDataElements" type="ns_p:SetpointDataElementsType"/>
    <xs:complexType name="SetpointListDataType">
        <xs:sequence>
            <xs:element maxOccurs="unbounded" minOccurs="0" ref="ns_p:setpointData"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="setpointListData" type="ns_p:SetpointListDataType"/>
    <xs:complexType name="SetpointListDataSelectorsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="setpointId" type="ns_p:SetpointIdType"/>
```



```
        </xs:sequence>
    </xs:complexType>
    <xs:element name="setpointListDataSelectors" type="ns_p:SetpointListDataSelectorsType"/>
    <xs:complexType name="SetpointConstraintsDataType">
        <xs:sequence>
            <xs:element minOccurs="0" name="setpointId" type="ns_p:SetpointIdType"/>
            <xs:element minOccurs="0" name="setpointRangeMin" type="ns_p:ScaledNumberType"/>
            <xs:element minOccurs="0" name="setpointRangeMax" type="ns_p:ScaledNumberType"/>
            <xs:element minOccurs="0" name="setpointStepSize" type="ns_p:ScaledNumberType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="setpointConstraintsData" type="ns_p:SetpointConstraintsDataType"/>
    <xs:complexType name="SetpointConstraintsDataElementsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="setpointId" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="setpointRangeMin"
type="ns_p:ScaledNumberElementsType"/>
            <xs:element minOccurs="0" name="setpointRangeMax"
type="ns_p:ScaledNumberElementsType"/>
            <xs:element minOccurs="0" name="setpointStepSize"
type="ns_p:ScaledNumberElementsType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="setpointConstraintsDataElements"
type="ns_p:SetpointConstraintsDataElementsType"/>
    <xs:complexType name="SetpointConstraintsListDataType">
        <xs:sequence>
            <xs:element maxOccurs="unbounded" minOccurs="0"
ref="ns_p:setpointConstraintsData"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="setpointConstraintsListData"
type="ns_p:SetpointConstraintsListDataType"/>
    <xs:complexType name="SetpointConstraintsListDataSelectorsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="setpointId" type="ns_p:SetpointIdType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="setpointConstraintsListDataSelectors"
type="ns_p:SetpointConstraintsListDataSelectorsType"/>
    <xs:complexType name="SetpointDescriptionDataType">
        <xs:sequence>
            <xs:element minOccurs="0" name="setpointId" type="ns_p:SetpointIdType"/>
            <xs:element minOccurs="0" name="measurementId" type="ns_p:MeasurementIdType"/>
            <xs:element minOccurs="0" name="timeTableId" type="ns_p:TimeTableIdType"/>
            <xs:element minOccurs="0" name="setpointType" type="ns_p:SetpointTypeType"/>
            <xs:element minOccurs="0" name="unit" type="ns_p:UnitOfMeasurementType"/>
            <xs:element minOccurs="0" name="scopeType" type="ns_p:ScopeTypeType"/>
            <xs:element minOccurs="0" name="label" type="ns_p:LabelType"/>
            <xs:element minOccurs="0" name="description" type="ns_p:DescriptionType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="setpointDescriptionData" type="ns_p:SetpointDescriptionDataType"/>
    <xs:complexType name="SetpointDescriptionDataElementsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="setpointId" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="measurementId" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="timeTableId" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="setpointType" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="unit" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="scopeType" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="label" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="description" type="ns_p:ElementTagType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="setpointDescriptionDataElements"
type="ns_p:SetpointDescriptionDataElementsType"/>
    <xs:complexType name="SetpointDescriptionListDataType">
        <xs:sequence>
            <xs:element maxOccurs="unbounded" minOccurs="0"
ref="ns_p:setpointDescriptionData"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="setpointDescriptionListData"
type="ns_p:SetpointDescriptionListDataType"/>
    <xs:complexType name="SetpointDescriptionListDataSelectorsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="setpointId" type="ns_p:SetpointIdType"/>
```



```
            <xs:element minOccurs="0" name="measurementId" type="ns_p:MeasurementIdType"/>
            <xs:element minOccurs="0" name="timeTableId" type="ns_p:TimeTableIdType"/>
            <xs:element minOccurs="0" name="setpointType" type="ns_p:SetpointTypeType"/>
            <xs:element minOccurs="0" name="scopeType" type="ns_p:ScopeTypeType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="setpointDescriptionListDataSelectors"
type="ns_p:SetpointDescriptionListDataSelectorsType"/>
</xs:schema>
```

A.2.22 SubscriptionManagement

File name: EEBus_SPINE_TS_SubscriptionManagement.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
    Smart Premises Interoperable Neutral-Message Exchange (SPINE)
    Version 1.1.1
    2018-12-21
    Copyright (c) 2018 EEBus Initiative e.V.  All Rights Reserved.
    Source: https://www.eebus.org/en/specifications/
-->
<xs:schema xmlns:ns_p="http://docs.eebus.org/spine/xsd/v1"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://docs.eebus.org/spine/xsd/v1" version="1.1.1"
    blockDefault="#all" elementFormDefault="qualified">
    <xs:annotation>
        <xs:documentation>EEBus SPINE Specification schema. Copyright 2018 EEBus Initiative
e.V.  All rights reserved.</xs:documentation>
    </xs:annotation>
    <xs:include schemaLocation="EEBus_SPINE_TS_CommonDataTypes.xsd"/>
    <xs:simpleType name="SubscriptionIdType">
        <xs:restriction base="xs:unsignedInt"/>
    </xs:simpleType>
    <xs:complexType name="SubscriptionManagementEntryDataType">
        <xs:sequence>
            <xs:element minOccurs="0" name="subscriptionId" type="ns_p:SubscriptionIdType"/>
            <xs:element minOccurs="0" name="clientAddress" type="ns_p:FeatureAddressType"/>
            <xs:element minOccurs="0" name="serverAddress" type="ns_p:FeatureAddressType"/>
            <xs:element minOccurs="0" name="label" type="ns_p:LabelType"/>
            <xs:element minOccurs="0" name="description" type="ns_p:DescriptionType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="subscriptionManagementEntryData"
type="ns_p:SubscriptionManagementEntryDataType"/>
    <xs:complexType name="SubscriptionManagementEntryDataElementsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="subscriptionId" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="clientAddress"
type="ns_p:FeatureAddressElementsType"/>
            <xs:element minOccurs="0" name="serverAddress"
type="ns_p:FeatureAddressElementsType"/>
            <xs:element minOccurs="0" name="label" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="description" type="ns_p:ElementTagType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="subscriptionManagementEntryDataElements"
type="ns_p:SubscriptionManagementEntryDataElementsType"/>
    <xs:complexType name="SubscriptionManagementEntryListDataType">
        <xs:sequence>
            <xs:element maxOccurs="unbounded" minOccurs="0"
ref="ns_p:subscriptionManagementEntryData"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="subscriptionManagementEntryListData"
type="ns_p:SubscriptionManagementEntryListDataType"/>
    <xs:complexType name="SubscriptionManagementEntryListDataSelectorsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="subscriptionId" type="ns_p:SubscriptionIdType"/>
            <xs:element minOccurs="0" name="clientAddress" type="ns_p:FeatureAddressType"/>
            <xs:element minOccurs="0" name="serverAddress" type="ns_p:FeatureAddressType"/>
        </xs:sequence>
```

```
    </xs:complexType>
    <xs:element name="subscriptionManagementEntryListDataSelectors"
type="ns_p:SubscriptionManagementEntryListDataSelectorsType"/>
    <xs:complexType name="SubscriptionManagementRequestCallType">
        <xs:sequence>
            <xs:element minOccurs="0" name="clientAddress" type="ns_p:FeatureAddressType"/>
            <xs:element minOccurs="0" name="serverAddress" type="ns_p:FeatureAddressType"/>
            <xs:element minOccurs="0" name="serverFeatureType" type="ns_p:FeatureTypeType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="subscriptionManagementRequestCall"
type="ns_p:SubscriptionManagementRequestCallType"/>
    <xs:complexType name="SubscriptionManagementRequestCallElementsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="clientAddress"
type="ns_p:FeatureAddressElementsType"/>
            <xs:element minOccurs="0" name="serverAddress"
type="ns_p:FeatureAddressElementsType"/>
            <xs:element minOccurs="0" name="serverFeatureType" type="ns_p:ElementTagType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="subscriptionManagementRequestCallElements"
type="ns_p:SubscriptionManagementRequestCallElementsType"/>
    <xs:complexType name="SubscriptionManagementDeleteCallType">
        <xs:sequence>
            <xs:element minOccurs="0" name="subscriptionId" type="ns_p:SubscriptionIdType"/>
            <xs:element minOccurs="0" name="clientAddress" type="ns_p:FeatureAddressType"/>
            <xs:element minOccurs="0" name="serverAddress" type="ns_p:FeatureAddressType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="subscriptionManagementDeleteCall"
type="ns_p:SubscriptionManagementDeleteCallType"/>
    <xs:complexType name="SubscriptionManagementDeleteCallElementsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="subscriptionId" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="clientAddress"
type="ns_p:FeatureAddressElementsType"/>
            <xs:element minOccurs="0" name="serverAddress"
type="ns_p:FeatureAddressElementsType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="subscriptionManagementDeleteCallElements"
type="ns_p:SubscriptionManagementDeleteCallElementsType"/>
</xs:schema>
```

A.2.23 SupplyCondition

File name: EEBus_SPINE_TS_SupplyCondition.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
    Smart Premises Interoperable Neutral-Message Exchange (SPINE)
    Version 1.1.1
    2018-12-21
    Copyright (c) 2018 EEBus Initiative e.V.  All Rights Reserved.
    Source: https://www.eebus.org/en/specifications/
-->
<xs:schema xmlns:ns_p="http://docs.eebus.org/spine/xsd/v1"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://docs.eebus.org/spine/xsd/v1" version="1.1.1"
    blockDefault="#all" elementFormDefault="qualified">
    <xs:annotation>
        <xs:documentation>EEBus SPINE Specification schema. Copyright 2018 EEBus Initiative
e.V.  All rights reserved.</xs:documentation>
    </xs:annotation>
    <xs:include schemaLocation="EEBus_SPINE_TS_CommonDataTypes.xsd"/>
    <xs:include schemaLocation="EEBus_SPINE_TS_Threshold.xsd"/>
    <xs:simpleType name="ConditionIdType">
        <xs:restriction base="xs:unsignedInt"/>
    </xs:simpleType>
    <xs:simpleType name="SupplyConditionEventTypeType">
        <xs:union memberTypes="ns_p:SupplyConditionEventTypeEnumType ns_p:EnumExtendType"/>
```



```
17960       </xs:simpleType>
17961       <xs:simpleType name="SupplyConditionEventTypeEnumType">
17962           <xs:restriction base="xs:string">
17963               <xs:enumeration value="thesholdExceeded"/>
17964               <xs:enumeration value="fallenBelowThreshold"/>
17965               <xs:enumeration value="supplyInterrupt"/>
17966               <xs:enumeration value="releaseOfLimitations"/>
17967               <xs:enumeration value="otherProblem"/>
17968               <xs:enumeration value="gridConditionUpdate"/>
17969           </xs:restriction>
17970       </xs:simpleType>
17971       <xs:simpleType name="SupplyConditionOriginatorType">
17972           <xs:union memberTypes="ns_p:SupplyConditionOriginatorEnumType ns_p:EnumExtendType"/>
17973       </xs:simpleType>
17974       <xs:simpleType name="SupplyConditionOriginatorEnumType">
17975           <xs:restriction base="xs:string">
17976               <xs:enumeration value="externDSO"/>
17977               <xs:enumeration value="externSupplier"/>
17978               <xs:enumeration value="internalLimit"/>
17979               <xs:enumeration value="internalService"/>
17980               <xs:enumeration value="internalUser"/>
17981           </xs:restriction>
17982       </xs:simpleType>
17983       <xs:simpleType name="GridConditionType">
17984           <xs:union memberTypes="ns_p:GridConditionEnumType ns_p:EnumExtendType"/>
17985       </xs:simpleType>
17986       <xs:simpleType name="GridConditionEnumType">
17987           <xs:restriction base="xs:string">
17988               <xs:enumeration value="consumptionRed"/>
17989               <xs:enumeration value="consumptionYellow"/>
17990               <xs:enumeration value="good"/>
17991               <xs:enumeration value="productionYellow"/>
17992               <xs:enumeration value="productionRed"/>
17993           </xs:restriction>
17994       </xs:simpleType>
17995       <xs:complexType name="SupplyConditionDataType">
17996           <xs:sequence>
17997               <xs:element minOccurs="0" name="conditionId" type="ns_p:ConditionIdType"/>
17998               <xs:element minOccurs="0" name="timestamp"
17999   type="ns_p:AbsoluteOrRelativeTimeType"/>
18000               <xs:element minOccurs="0" name="eventType"
18001   type="ns_p:SupplyConditionEventTypeType"/>
18002               <xs:element minOccurs="0" name="originator"
18003   type="ns_p:SupplyConditionOriginatorType"/>
18004               <xs:element minOccurs="0" name="thresholdId" type="ns_p:ThresholdIdType"/>
18005               <xs:element minOccurs="0" name="thresholdPercentage"
18006   type="ns_p:ScaledNumberType"/>
18007               <xs:element minOccurs="0" name="relevantPeriod" type="ns_p:TimePeriodType"/>
18008               <xs:element minOccurs="0" name="description" type="ns_p:DescriptionType"/>
18009               <xs:element minOccurs="0" name="gridCondition" type="ns_p:GridConditionType"/>
18010           </xs:sequence>
18011       </xs:complexType>
18012       <xs:element name="supplyConditionData" type="ns_p:SupplyConditionDataType"/>
18013       <xs:complexType name="SupplyConditionDataElementsType">
18014           <xs:sequence>
18015               <xs:element minOccurs="0" name="conditionId" type="ns_p:ElementTagType"/>
18016               <xs:element minOccurs="0" name="timestamp" type="ns_p:ElementTagType"/>
18017               <xs:element minOccurs="0" name="eventType" type="ns_p:ElementTagType"/>
18018               <xs:element minOccurs="0" name="originator" type="ns_p:ElementTagType"/>
18019               <xs:element minOccurs="0" name="thresholdId" type="ns_p:ElementTagType"/>
18020               <xs:element minOccurs="0" name="thresholdPercentage"
18021   type="ns_p:ScaledNumberElementsType"/>
18022               <xs:element minOccurs="0" name="relevantPeriod"
18023   type="ns_p:TimePeriodElementsType"/>
18024               <xs:element minOccurs="0" name="description" type="ns_p:ElementTagType"/>
18025               <xs:element minOccurs="0" name="gridCondition" type="ns_p:ElementTagType"/>
18026           </xs:sequence>
18027       </xs:complexType>
18028       <xs:element name="supplyConditionDataElements"
18029   type="ns_p:SupplyConditionDataElementsType"/>
18030       <xs:complexType name="SupplyConditionListDataType">
18031           <xs:sequence>
18032               <xs:element maxOccurs="unbounded" minOccurs="0" ref="ns_p:supplyConditionData"/>
18033           </xs:sequence>
18034       </xs:complexType>
18035       <xs:element name="supplyConditionListData" type="ns_p:SupplyConditionListDataType"/>
18036       <xs:complexType name="SupplyConditionListDataSelectorsType">
18037           <xs:sequence>
```



```
            <xs:element minOccurs="0" name="conditionId" type="ns_p:ConditionIdType"/>
            <xs:element minOccurs="0" name="timestampInterval"
type="ns_p:TimestampIntervalType"/>
            <xs:element minOccurs="0" name="eventType"
type="ns_p:SupplyConditionEventTypeType"/>
            <xs:element minOccurs="0" name="originator"
type="ns_p:SupplyConditionOriginatorType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="supplyConditionListDataSelectors"
type="ns_p:SupplyConditionListDataSelectorsType"/>
    <xs:complexType name="SupplyConditionDescriptionDataType">
        <xs:sequence>
            <xs:element minOccurs="0" name="conditionId" type="ns_p:ConditionIdType"/>
            <xs:element minOccurs="0" name="commodityType" type="ns_p:CommodityTypeType"/>
            <xs:element minOccurs="0" name="positiveEnergyDirection"
type="ns_p:EnergyDirectionType"/>
            <xs:element minOccurs="0" name="label" type="ns_p:LabelType"/>
            <xs:element minOccurs="0" name="description" type="ns_p:DescriptionType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="supplyConditionDescriptionData"
type="ns_p:SupplyConditionDescriptionDataType"/>
    <xs:complexType name="SupplyConditionDescriptionDataElementsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="conditionId" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="commodityType" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="positiveEnergyDirection"
type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="label" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="description" type="ns_p:ElementTagType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="supplyConditionDescriptionDataElements"
type="ns_p:SupplyConditionDescriptionDataElementsType"/>
    <xs:complexType name="SupplyConditionDescriptionListDataType">
        <xs:sequence>
            <xs:element maxOccurs="unbounded" minOccurs="0"
ref="ns_p:supplyConditionDescriptionData"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="supplyConditionDescriptionListData"
type="ns_p:SupplyConditionDescriptionListDataType"/>
    <xs:complexType name="SupplyConditionDescriptionListDataSelectorsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="conditionId" type="ns_p:ConditionIdType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="supplyConditionDescriptionListDataSelectors"
type="ns_p:SupplyConditionDescriptionListDataSelectorsType"/>
    <xs:complexType name="SupplyConditionThresholdRelationDataType">
        <xs:sequence>
            <xs:element minOccurs="0" name="conditionId" type="ns_p:ConditionIdType"/>
            <xs:element maxOccurs="unbounded" minOccurs="0" name="thresholdId"
type="ns_p:ThresholdIdType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="supplyConditionThresholdRelationData"
type="ns_p:SupplyConditionThresholdRelationDataType"/>
    <xs:complexType name="SupplyConditionThresholdRelationDataElementsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="conditionId" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="thresholdId" type="ns_p:ElementTagType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="supplyConditionThresholdRelationDataElements"
type="ns_p:SupplyConditionThresholdRelationDataElementsType"/>
    <xs:complexType name="SupplyConditionThresholdRelationListDataType">
        <xs:sequence>
            <xs:element maxOccurs="unbounded" minOccurs="0"
ref="ns_p:supplyConditionThresholdRelationData"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="supplyConditionThresholdRelationListData"
type="ns_p:SupplyConditionThresholdRelationListDataType"/>
    <xs:complexType name="SupplyConditionThresholdRelationListDataSelectorsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="conditionId" type="ns_p:ConditionIdType"/>
```



```
            <xs:element minOccurs="0" name="thresholdId" type="ns_p:ThresholdIdType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="supplyConditionThresholdRelationListDataSelectors"
type="ns_p:SupplyConditionThresholdRelationListDataSelectorsType"/>
</xs:schema>
```

A.2.24 TariffInformation

File name: EEBus_SPINE_TS_TariffInformation.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
    Smart Premises Interoperable Neutral-Message Exchange (SPINE)
    Version 1.1.1
    2018-12-21
    Copyright (c) 2018 EEBus Initiative e.V.  All Rights Reserved.
    Source: https://www.eebus.org/en/specifications/
-->
<xs:schema xmlns:ns_p="http://docs.eebus.org/spine/xsd/v1"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://docs.eebus.org/spine/xsd/v1" version="1.1.1"
    blockDefault="#all" elementFormDefault="qualified">
    <xs:annotation>
        <xs:documentation>EEBus SPINE Specification schema. Copyright 2018 EEBus Initiative
e.V.  All rights reserved.</xs:documentation>
    </xs:annotation>
    <xs:include schemaLocation="EEBus_SPINE_TS_CommonDataTypes.xsd"/>
    <xs:include schemaLocation="EEBus_SPINE_TS_Measurement.xsd"/>
    <xs:include schemaLocation="EEBus_SPINE_TS_TimeTable.xsd"/>
    <xs:simpleType name="TariffIdType">
        <xs:restriction base="xs:unsignedInt"/>
    </xs:simpleType>
    <xs:simpleType name="TariffCountType">
        <xs:restriction base="ns_p:TariffIdType"/>
    </xs:simpleType>
    <xs:simpleType name="TierBoundaryIdType">
        <xs:restriction base="xs:unsignedInt"/>
    </xs:simpleType>
    <xs:simpleType name="TierBoundaryCountType">
        <xs:restriction base="ns_p:TierBoundaryIdType"/>
    </xs:simpleType>
    <xs:simpleType name="TierBoundaryTypeType">
        <xs:union memberTypes="ns_p:TierBoundaryTypeEnumType ns_p:EnumExtendType"/>
    </xs:simpleType>
    <xs:simpleType name="TierBoundaryTypeEnumType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="powerBoundary"/>
            <xs:enumeration value="energyBoundary"/>
            <xs:enumeration value="countBoundary"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="CommodityIdType">
        <xs:restriction base="xs:unsignedInt"/>
    </xs:simpleType>
    <xs:simpleType name="TierIdType">
        <xs:restriction base="xs:unsignedInt"/>
    </xs:simpleType>
    <xs:simpleType name="TierCountType">
        <xs:restriction base="ns_p:TierIdType"/>
    </xs:simpleType>
    <xs:simpleType name="TierTypeType">
        <xs:union memberTypes="ns_p:TierTypeEnumType ns_p:EnumExtendType"/>
    </xs:simpleType>
    <xs:simpleType name="TierTypeEnumType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="fixedCost"/>
            <xs:enumeration value="dynamicCost"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="IncentiveIdType">
        <xs:restriction base="xs:unsignedInt"/>
```

```
18189       </xs:simpleType>
18190       <xs:simpleType name="IncentiveCountType">
18191           <xs:restriction base="ns_p:IncentiveIdType"/>
18192       </xs:simpleType>
18193       <xs:simpleType name="IncentiveTypeType">
18194           <xs:union memberTypes="ns_p:IncentiveTypeEnumType ns_p:EnumExtendType"/>
18195       </xs:simpleType>
18196       <xs:simpleType name="IncentiveTypeEnumType">
18197           <xs:restriction base="xs:string">
18198               <xs:enumeration value="absoluteCost"/>
18199               <xs:enumeration value="relativeCost"/>
18200               <xs:enumeration value="renewableEnergyPercentage"/>
18201               <xs:enumeration value="co2Emission"/>
18202           </xs:restriction>
18203       </xs:simpleType>
18204       <xs:simpleType name="IncentivePriorityType">
18205           <xs:restriction base="xs:unsignedInt"/>
18206       </xs:simpleType>
18207       <xs:simpleType name="IncentiveValueTypeType">
18208           <xs:union memberTypes="ns_p:IncentiveValueTypeEnumType ns_p:EnumExtendType"/>
18209       </xs:simpleType>
18210       <xs:simpleType name="IncentiveValueTypeEnumType">
18211           <xs:restriction base="xs:string">
18212               <xs:enumeration value="value"/>
18213               <xs:enumeration value="averageValue"/>
18214               <xs:enumeration value="minValue"/>
18215               <xs:enumeration value="maxValue"/>
18216           </xs:restriction>
18217       </xs:simpleType>
18218       <xs:complexType name="TariffOverallConstraintsDataType">
18219           <xs:sequence>
18220               <xs:element minOccurs="0" name="maxTariffCount" type="ns_p:TariffCountType"/>
18221               <xs:element minOccurs="0" name="maxBoundaryCount"
18222   type="ns_p:TierBoundaryCountType"/>
18223               <xs:element minOccurs="0" name="maxTierCount" type="ns_p:TierCountType"/>
18224               <xs:element minOccurs="0" name="maxIncentiveCount"
18225   type="ns_p:IncentiveCountType"/>
18226               <xs:element minOccurs="0" name="maxBoundariesPerTariff"
18227   type="ns_p:TierBoundaryCountType"/>
18228               <xs:element minOccurs="0" name="maxTiersPerTariff" type="ns_p:TierCountType"/>
18229               <xs:element minOccurs="0" name="maxBoundariesPerTier"
18230   type="ns_p:TierBoundaryCountType"/>
18231               <xs:element minOccurs="0" name="maxIncentivesPerTier"
18232   type="ns_p:IncentiveCountType"/>
18233           </xs:sequence>
18234       </xs:complexType>
18235       <xs:element name="tariffOverallConstraintsData"
18236   type="ns_p:TariffOverallConstraintsDataType"/>
18237       <xs:complexType name="TariffOverallConstraintsDataElementsType">
18238           <xs:sequence>
18239               <xs:element minOccurs="0" name="maxTariffCount" type="ns_p:ElementTagType"/>
18240               <xs:element minOccurs="0" name="maxBoundaryCount" type="ns_p:ElementTagType"/>
18241               <xs:element minOccurs="0" name="maxTierCount" type="ns_p:ElementTagType"/>
18242               <xs:element minOccurs="0" name="maxIncentiveCount" type="ns_p:ElementTagType"/>
18243               <xs:element minOccurs="0" name="maxBoundariesPerTariff"
18244   type="ns_p:ElementTagType"/>
18245               <xs:element minOccurs="0" name="maxTiersPerTariff" type="ns_p:ElementTagType"/>
18246               <xs:element minOccurs="0" name="maxBoundariesPerTier" type="ns_p:ElementTagType"/>
18247               <xs:element minOccurs="0" name="maxIncentivesPerTier" type="ns_p:ElementTagType"/>
18248           </xs:sequence>
18249       </xs:complexType>
18250       <xs:element name="tariffOverallConstraintsDataElements"
18251   type="ns_p:TariffOverallConstraintsDataElementsType"/>
18252       <xs:complexType name="TariffDataType">
18253           <xs:sequence>
18254               <xs:element name="tariffId" type="ns_p:TariffIdType" minOccurs="0"/>
18255               <xs:element name="activeTierId" type="ns_p:TierIdType" minOccurs="0"
18256   maxOccurs="unbounded"/>
18257           </xs:sequence>
18258       </xs:complexType>
18259       <xs:element name="tariffData" type="ns_p:TariffDataType"/>
18260       <xs:complexType name="TariffDataElementsType">
18261           <xs:sequence>
18262               <xs:element name="tariffId" type="ns_p:ElementTagType" minOccurs="0"/>
18263               <xs:element name="activeTierId" type="ns_p:ElementTagType" minOccurs="0"/>
18264           </xs:sequence>
18265       </xs:complexType>
18266       <xs:element name="tariffDataElements" type="ns_p:TariffDataElementsType"/>
```



```
18267       <xs:complexType name="TariffListDataType">
18268           <xs:sequence>
18269               <xs:element maxOccurs="unbounded" minOccurs="0" ref="ns_p:tariffData"/>
18270           </xs:sequence>
18271       </xs:complexType>
18272       <xs:element name="tariffListData" type="ns_p:TariffListDataType"/>
18273       <xs:complexType name="TariffListDataSelectorsType">
18274           <xs:sequence>
18275               <xs:element name="tariffId" type="ns_p:TariffIdType" minOccurs="0"/>
18276               <xs:element name="activeTierId" type="ns_p:TierIdType" minOccurs="0"/>
18277           </xs:sequence>
18278       </xs:complexType>
18279       <xs:element name="tariffListDataSelectors" type="ns_p:TariffListDataSelectorsType"/>
18280       <xs:complexType name="TariffTierRelationDataType">
18281           <xs:sequence>
18282               <xs:element name="tariffId" type="ns_p:TariffIdType" minOccurs="0"/>
18283               <xs:element name="tierId" maxOccurs="unbounded" type="ns_p:TierIdType"
18284   minOccurs="0"/>
18285           </xs:sequence>
18286       </xs:complexType>
18287       <xs:element name="tariffTierRelationData" type="ns_p:TariffTierRelationDataType"/>
18288       <xs:complexType name="TariffTierRelationDataElementsType">
18289           <xs:sequence>
18290               <xs:element name="tariffId" type="ns_p:ElementTagType" minOccurs="0"/>
18291               <xs:element name="tierId" type="ns_p:ElementTagType" minOccurs="0"/>
18292           </xs:sequence>
18293       </xs:complexType>
18294       <xs:element name="tariffTierRelationDataElements"
18295   type="ns_p:TariffTierRelationDataElementsType"/>
18296       <xs:complexType name="TariffTierRelationListDataType">
18297           <xs:sequence>
18298               <xs:element maxOccurs="unbounded" minOccurs="0"
18299   ref="ns_p:tariffTierRelationData"/>
18300           </xs:sequence>
18301       </xs:complexType>
18302       <xs:element name="tariffTierRelationListData" type="ns_p:TariffTierRelationListDataType"/>
18303       <xs:complexType name="TariffTierRelationListDataSelectorsType">
18304           <xs:sequence>
18305               <xs:element name="tariffId" type="ns_p:TariffIdType" minOccurs="0"/>
18306               <xs:element name="tierId" type="ns_p:TierIdType" minOccurs="0"/>
18307           </xs:sequence>
18308       </xs:complexType>
18309       <xs:element name="tariffTierRelationListDataSelectors"
18310   type="ns_p:TariffTierRelationListDataSelectorsType"/>
18311       <xs:complexType name="TariffBoundaryRelationDataType">
18312           <xs:sequence>
18313               <xs:element name="tariffId" type="ns_p:TariffIdType" minOccurs="0"/>
18314               <xs:element minOccurs="0" name="boundaryId" type="ns_p:TierBoundaryIdType"
18315   maxOccurs="unbounded"/>
18316           </xs:sequence>
18317       </xs:complexType>
18318       <xs:element name="tariffBoundaryRelationData" type="ns_p:TariffBoundaryRelationDataType"/>
18319       <xs:complexType name="TariffBoundaryRelationDataElementsType">
18320           <xs:sequence>
18321               <xs:element name="tariffId" type="ns_p:ElementTagType" minOccurs="0"/>
18322               <xs:element name="boundaryId" type="ns_p:ElementTagType" minOccurs="0"/>
18323           </xs:sequence>
18324       </xs:complexType>
18325       <xs:element name="tariffBoundaryRelationDataElements"
18326   type="ns_p:TariffBoundaryRelationDataElementsType"/>
18327       <xs:complexType name="TariffBoundaryRelationListDataType">
18328           <xs:sequence>
18329               <xs:element maxOccurs="unbounded" minOccurs="0"
18330   ref="ns_p:tariffBoundaryRelationData"/>
18331           </xs:sequence>
18332       </xs:complexType>
18333       <xs:element name="tariffBoundaryRelationListData"
18334   type="ns_p:TariffBoundaryRelationListDataType"/>
18335       <xs:complexType name="TariffBoundaryRelationListDataSelectorsType">
18336           <xs:sequence>
18337               <xs:element name="tariffId" type="ns_p:TariffIdType" minOccurs="0"/>
18338               <xs:element minOccurs="0" name="boundaryId" type="ns_p:TierBoundaryIdType"/>
18339           </xs:sequence>
18340       </xs:complexType>
18341       <xs:element name="tariffBoundaryRelationListDataSelectors"
18342   type="ns_p:TariffBoundaryRelationListDataSelectorsType"/>
18343       <xs:complexType name="TariffDescriptionDataType">
18344           <xs:sequence>
```



```
            <xs:element name="tariffId" type="ns_p:TariffIdType" minOccurs="0"/>
            <xs:element name="commodityId" type="ns_p:CommodityIdType" minOccurs="0"/>
            <xs:element name="measurementId" type="ns_p:MeasurementIdType" minOccurs="0"/>
            <xs:element name="tariffWriteable" type="xs:boolean" minOccurs="0"/>
            <xs:element name="updateRequired" type="xs:boolean" minOccurs="0"/>
            <xs:element minOccurs="0" name="scopeType" type="ns_p:ScopeTypeType"/>
            <xs:element name="label" type="ns_p:LabelType" minOccurs="0"/>
            <xs:element name="description" type="ns_p:DescriptionType" minOccurs="0"/>
            <xs:element name="slotIdSupport" type="xs:boolean" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="tariffDescriptionData" type="ns_p:TariffDescriptionDataType"/>
    <xs:complexType name="TariffDescriptionDataElementsType">
        <xs:sequence>
            <xs:element name="tariffId" type="ns_p:ElementTagType" minOccurs="0"/>
            <xs:element name="commodityId" type="ns_p:ElementTagType" minOccurs="0"/>
            <xs:element name="measurementId" type="ns_p:ElementTagType" minOccurs="0"/>
            <xs:element name="tariffWriteable" type="ns_p:ElementTagType" minOccurs="0"/>
            <xs:element name="updateRequired" type="ns_p:ElementTagType" minOccurs="0"/>
            <xs:element minOccurs="0" name="scopeType" type="ns_p:ElementTagType"/>
            <xs:element name="label" type="ns_p:ElementTagType" minOccurs="0"/>
            <xs:element name="description" type="ns_p:ElementTagType" minOccurs="0"/>
            <xs:element name="slotIdSupport" type="ns_p:ElementTagType" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="tariffDescriptionDataElements"
type="ns_p:TariffDescriptionDataElementsType"/>
    <xs:complexType name="TariffDescriptionListDataType">
        <xs:sequence>
            <xs:element maxOccurs="unbounded" minOccurs="0" ref="ns_p:tariffDescriptionData"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="tariffDescriptionListData" type="ns_p:TariffDescriptionListDataType"/>
    <xs:complexType name="TariffDescriptionListDataSelectorsType">
        <xs:sequence>
            <xs:element name="tariffId" type="ns_p:TariffIdType" minOccurs="0"/>
            <xs:element name="commodityId" type="ns_p:CommodityIdType" minOccurs="0"/>
            <xs:element name="measurementId" type="ns_p:MeasurementIdType" minOccurs="0"/>
            <xs:element minOccurs="0" name="scopeType" type="ns_p:ScopeTypeType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="tariffDescriptionListDataSelectors"
type="ns_p:TariffDescriptionListDataSelectorsType"/>
    <xs:complexType name="TierBoundaryDataType">
        <xs:sequence>
            <xs:element minOccurs="0" name="boundaryId" type="ns_p:TierBoundaryIdType"/>
            <xs:element minOccurs="0" name="timePeriod" type="ns_p:TimePeriodType"/>
            <xs:element minOccurs="0" name="timeTableId" type="ns_p:TimeTableIdType"/>
            <xs:element minOccurs="0" name="lowerBoundaryValue" type="ns_p:ScaledNumberType"/>
            <xs:element minOccurs="0" name="upperBoundaryValue" type="ns_p:ScaledNumberType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="tierBoundaryData" type="ns_p:TierBoundaryDataType"/>
    <xs:complexType name="TierBoundaryDataElementsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="boundaryId" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="timePeriod" type="ns_p:TimePeriodElementsType"/>
            <xs:element minOccurs="0" name="timeTableId" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="lowerBoundaryValue"
type="ns_p:ScaledNumberElementsType"/>
            <xs:element minOccurs="0" name="upperBoundaryValue"
type="ns_p:ScaledNumberElementsType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="tierBoundaryDataElements" type="ns_p:TierBoundaryDataElementsType"/>
    <xs:complexType name="TierBoundaryListDataType">
        <xs:sequence>
            <xs:element maxOccurs="unbounded" minOccurs="0" ref="ns_p:tierBoundaryData"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="tierBoundaryListData" type="ns_p:TierBoundaryListDataType"/>
    <xs:complexType name="TierBoundaryListDataSelectorsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="boundaryId" type="ns_p:TierBoundaryIdType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="tierBoundaryListDataSelectors"
type="ns_p:TierBoundaryListDataSelectorsType"/>
```



```
18423       <xs:complexType name="TierBoundaryDescriptionDataType">
18424           <xs:sequence>
18425               <xs:element minOccurs="0" name="boundaryId" type="ns_p:TierBoundaryIdType"/>
18426               <xs:element minOccurs="0" name="boundaryType" type="ns_p:TierBoundaryTypeType"/>
18427               <xs:element minOccurs="0" name="validForTierId" type="ns_p:TierIdType"/>
18428               <xs:element minOccurs="0" name="switchToTierIdWhenLower" type="ns_p:TierIdType"/>
18429               <xs:element minOccurs="0" name="switchToTierIdWhenHigher" type="ns_p:TierIdType"/>
18430               <xs:element minOccurs="0" name="boundaryUnit" type="ns_p:UnitOfMeasurementType"/>
18431               <xs:element minOccurs="0" name="label" type="ns_p:LabelType"/>
18432               <xs:element minOccurs="0" name="description" type="ns_p:DescriptionType"/>
18433           </xs:sequence>
18434       </xs:complexType>
18435       <xs:element name="tierBoundaryDescriptionData"
18436   type="ns_p:TierBoundaryDescriptionDataType"/>
18437       <xs:complexType name="TierBoundaryDescriptionDataElementsType">
18438           <xs:sequence>
18439               <xs:element minOccurs="0" name="boundaryId" type="ns_p:ElementTagType"/>
18440               <xs:element minOccurs="0" name="boundaryType" type="ns_p:ElementTagType"/>
18441               <xs:element minOccurs="0" name="validForTierId" type="ns_p:ElementTagType"/>
18442               <xs:element minOccurs="0" name="switchToTierIdWhenLower"
18443   type="ns_p:ElementTagType"/>
18444               <xs:element minOccurs="0" name="switchToTierIdWhenHigher"
18445   type="ns_p:ElementTagType"/>
18446               <xs:element minOccurs="0" name="boundaryUnit" type="ns_p:ElementTagType"/>
18447               <xs:element minOccurs="0" name="label" type="ns_p:ElementTagType"/>
18448               <xs:element minOccurs="0" name="description" type="ns_p:ElementTagType"/>
18449           </xs:sequence>
18450       </xs:complexType>
18451       <xs:element name="tierBoundaryDescriptionDataElements"
18452   type="ns_p:TierBoundaryDescriptionDataElementsType"/>
18453       <xs:complexType name="TierBoundaryDescriptionListDataType">
18454           <xs:sequence>
18455               <xs:element maxOccurs="unbounded" minOccurs="0"
18456   ref="ns_p:tierBoundaryDescriptionData"/>
18457           </xs:sequence>
18458       </xs:complexType>
18459       <xs:element name="tierBoundaryDescriptionListData"
18460   type="ns_p:TierBoundaryDescriptionListDataType"/>
18461       <xs:complexType name="TierBoundaryDescriptionListDataSelectorsType">
18462           <xs:sequence>
18463               <xs:element minOccurs="0" name="boundaryId" type="ns_p:TierBoundaryIdType"/>
18464               <xs:element minOccurs="0" name="boundaryType" type="ns_p:TierBoundaryTypeType"/>
18465           </xs:sequence>
18466       </xs:complexType>
18467       <xs:element name="tierBoundaryDescriptionListDataSelectors"
18468   type="ns_p:TierBoundaryDescriptionListDataSelectorsType"/>
18469       <xs:complexType name="CommodityDataType">
18470           <xs:sequence>
18471               <xs:element name="commodityId" type="ns_p:CommodityIdType" minOccurs="0"/>
18472               <xs:element name="commodityType" type="ns_p:CommodityTypeType" minOccurs="0"/>
18473               <xs:element name="positiveEnergyDirection" type="ns_p:EnergyDirectionType"
18474   minOccurs="0"/>
18475               <xs:element name="label" type="ns_p:LabelType" minOccurs="0"/>
18476               <xs:element name="description" type="ns_p:DescriptionType" minOccurs="0"/>
18477           </xs:sequence>
18478       </xs:complexType>
18479       <xs:element name="commodityData" type="ns_p:CommodityDataType"/>
18480       <xs:complexType name="CommodityDataElementsType">
18481           <xs:sequence>
18482               <xs:element name="commodityId" type="ns_p:ElementTagType" minOccurs="0"/>
18483               <xs:element name="commodityType" type="ns_p:ElementTagType" minOccurs="0"/>
18484               <xs:element name="positiveEnergyDirection" type="ns_p:ElementTagType"
18485   minOccurs="0"/>
18486               <xs:element name="label" type="ns_p:ElementTagType" minOccurs="0"/>
18487               <xs:element name="description" type="ns_p:ElementTagType" minOccurs="0"/>
18488           </xs:sequence>
18489       </xs:complexType>
18490       <xs:element name="commodityDataElements" type="ns_p:CommodityDataElementsType"/>
18491       <xs:complexType name="CommodityListDataType">
18492           <xs:sequence>
18493               <xs:element maxOccurs="unbounded" minOccurs="0" ref="ns_p:commodityData"/>
18494           </xs:sequence>
18495       </xs:complexType>
18496       <xs:element name="commodityListData" type="ns_p:CommodityListDataType"/>
18497       <xs:complexType name="CommodityListDataSelectorsType">
18498           <xs:sequence>
18499               <xs:element name="commodityId" type="ns_p:CommodityIdType" minOccurs="0"/>
18500               <xs:element name="commodityType" type="ns_p:CommodityTypeType" minOccurs="0"/>
```



```
18501           </xs:sequence>
18502       </xs:complexType>
18503       <xs:element name="commodityListDataSelectors" type="ns_p:CommodityListDataSelectorsType"/>
18504       <xs:complexType name="TierDataType">
18505           <xs:sequence>
18506               <xs:element name="tierId" type="ns_p:TierIdType" minOccurs="0"/>
18507               <xs:element name="timePeriod" type="ns_p:TimePeriodType" minOccurs="0"/>
18508               <xs:element minOccurs="0" name="timeTableId" type="ns_p:TimeTableIdType"/>
18509               <xs:element name="activeIncentiveId" type="ns_p:IncentiveIdType" minOccurs="0"
18510   maxOccurs="unbounded"/>
18511           </xs:sequence>
18512       </xs:complexType>
18513       <xs:element name="tierData" type="ns_p:TierDataType"/>
18514       <xs:complexType name="TierDataElementsType">
18515           <xs:sequence>
18516               <xs:element name="tierId" type="ns_p:ElementTagType" minOccurs="0"/>
18517               <xs:element name="timePeriod" type="ns_p:TimePeriodElementsType" minOccurs="0"/>
18518               <xs:element minOccurs="0" name="timeTableId" type="ns_p:ElementTagType"/>
18519               <xs:element name="activeIncentiveId" type="ns_p:ElementTagType" minOccurs="0"/>
18520           </xs:sequence>
18521       </xs:complexType>
18522       <xs:element name="tierDataElements" type="ns_p:TierDataElementsType"/>
18523       <xs:complexType name="TierListDataType">
18524           <xs:sequence>
18525               <xs:element maxOccurs="unbounded" minOccurs="0" ref="ns_p:tierData"/>
18526           </xs:sequence>
18527       </xs:complexType>
18528       <xs:element name="tierListData" type="ns_p:TierListDataType"/>
18529       <xs:complexType name="TierListDataSelectorsType">
18530           <xs:sequence>
18531               <xs:element name="tierId" type="ns_p:TierIdType" minOccurs="0"/>
18532               <xs:element name="activeIncentiveId" type="ns_p:IncentiveIdType" minOccurs="0"/>
18533           </xs:sequence>
18534       </xs:complexType>
18535       <xs:element name="tierListDataSelectors" type="ns_p:TierListDataSelectorsType"/>
18536       <xs:complexType name="TierIncentiveRelationDataType">
18537           <xs:sequence>
18538               <xs:element name="tierId" type="ns_p:TierIdType" minOccurs="0"/>
18539               <xs:element name="incentiveId" maxOccurs="unbounded" type="ns_p:IncentiveIdType"
18540   minOccurs="0"/>
18541           </xs:sequence>
18542       </xs:complexType>
18543       <xs:element name="tierIncentiveRelationData" type="ns_p:TierIncentiveRelationDataType"/>
18544       <xs:complexType name="TierIncentiveRelationDataElementsType">
18545           <xs:sequence>
18546               <xs:element name="tierId" type="ns_p:ElementTagType" minOccurs="0"/>
18547               <xs:element name="incentiveId" type="ns_p:ElementTagType" minOccurs="0"/>
18548           </xs:sequence>
18549       </xs:complexType>
18550       <xs:element name="tierIncentiveRelationDataElements"
18551   type="ns_p:TierIncentiveRelationDataElementsType"/>
18552       <xs:complexType name="TierIncentiveRelationListDataType">
18553           <xs:sequence>
18554               <xs:element maxOccurs="unbounded" minOccurs="0"
18555   ref="ns_p:tierIncentiveRelationData"/>
18556           </xs:sequence>
18557       </xs:complexType>
18558       <xs:element name="tierIncentiveRelationListData"
18559   type="ns_p:TierIncentiveRelationListDataType"/>
18560       <xs:complexType name="TierIncentiveRelationListDataSelectorsType">
18561           <xs:sequence>
18562               <xs:element name="tierId" type="ns_p:TierIdType" minOccurs="0"/>
18563               <xs:element name="incentiveId" type="ns_p:IncentiveIdType" minOccurs="0"/>
18564           </xs:sequence>
18565       </xs:complexType>
18566       <xs:element name="tierIncentiveRelationListDataSelectors"
18567   type="ns_p:TierIncentiveRelationListDataSelectorsType"/>
18568       <xs:complexType name="TierDescriptionDataType">
18569           <xs:sequence>
18570               <xs:element name="tierId" type="ns_p:TierIdType" minOccurs="0"/>
18571               <xs:element name="tierType" minOccurs="0" type="ns_p:TierTypeType"/>
18572               <xs:element name="label" type="ns_p:LabelType" minOccurs="0"/>
18573               <xs:element name="description" type="ns_p:DescriptionType" minOccurs="0"/>
18574           </xs:sequence>
18575       </xs:complexType>
18576       <xs:element name="tierDescriptionData" type="ns_p:TierDescriptionDataType"/>
18577       <xs:complexType name="TierDescriptionDataElementsType">
18578           <xs:sequence>
```



```
            <xs:element name="tierId" type="ns_p:ElementTagType" minOccurs="0"/>
            <xs:element name="tierType" minOccurs="0" type="ns_p:ElementTagType"/>
            <xs:element name="label" type="ns_p:ElementTagType" minOccurs="0"/>
            <xs:element name="description" type="ns_p:ElementTagType" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="tierDescriptionDataElements"
type="ns_p:TierDescriptionDataElementsType"/>
    <xs:complexType name="TierDescriptionListDataType">
        <xs:sequence>
            <xs:element maxOccurs="unbounded" minOccurs="0" ref="ns_p:tierDescriptionData"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="tierDescriptionListData" type="ns_p:TierDescriptionListDataType"/>
    <xs:complexType name="TierDescriptionListDataSelectorsType">
        <xs:sequence>
            <xs:element name="tierId" type="ns_p:TierIdType" minOccurs="0"/>
            <xs:element name="tierType" minOccurs="0" type="ns_p:TierTypeType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="tierDescriptionListDataSelectors"
type="ns_p:TierDescriptionListDataSelectorsType"/>
    <xs:complexType name="IncentiveDataType">
        <xs:sequence>
            <xs:element name="incentiveId" type="ns_p:IncentiveIdType" minOccurs="0"/>
            <xs:element name="valueType" type="ns_p:IncentiveValueTypeType" minOccurs="0"/>
            <xs:element name="timestamp" type="ns_p:AbsoluteOrRelativeTimeType"
minOccurs="0"/>
            <xs:element name="timePeriod" type="ns_p:TimePeriodType" minOccurs="0"/>
            <xs:element minOccurs="0" name="timeTableId" type="ns_p:TimeTableIdType"/>
            <xs:element minOccurs="0" name="value" type="ns_p:ScaledNumberType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="incentiveData" type="ns_p:IncentiveDataType"/>
    <xs:complexType name="IncentiveDataElementsType">
        <xs:sequence>
            <xs:element name="incentiveId" type="ns_p:ElementTagType" minOccurs="0"/>
            <xs:element name="valueType" type="ns_p:ElementTagType" minOccurs="0"/>
            <xs:element name="timestamp" type="ns_p:ElementTagType" minOccurs="0"/>
            <xs:element name="timePeriod" type="ns_p:TimePeriodElementsType" minOccurs="0"/>
            <xs:element minOccurs="0" name="timeTableId" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="value" type="ns_p:ElementTagType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="incentiveDataElements" type="ns_p:IncentiveDataElementsType"/>
    <xs:complexType name="IncentiveListDataType">
        <xs:sequence>
            <xs:element maxOccurs="unbounded" minOccurs="0" ref="ns_p:incentiveData"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="incentiveListData" type="ns_p:IncentiveListDataType"/>
    <xs:complexType name="IncentiveListDataSelectorsType">
        <xs:sequence>
            <xs:element name="incentiveId" type="ns_p:IncentiveIdType" minOccurs="0"/>
            <xs:element name="valueType" type="ns_p:IncentiveValueTypeType" minOccurs="0"/>
            <xs:element minOccurs="0" name="timestampInterval"
type="ns_p:TimestampIntervalType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="incentiveListDataSelectors" type="ns_p:IncentiveListDataSelectorsType"/>
    <xs:complexType name="IncentiveDescriptionDataType">
        <xs:sequence>
            <xs:element name="incentiveId" type="ns_p:IncentiveIdType" minOccurs="0"/>
            <xs:element name="incentiveType" type="ns_p:IncentiveTypeType" minOccurs="0"/>
            <xs:element name="incentivePriority" type="ns_p:IncentivePriorityType"
minOccurs="0"/>
            <xs:element name="currency" type="ns_p:CurrencyType" minOccurs="0"/>
            <xs:element name="unit" minOccurs="0" type="ns_p:UnitOfMeasurementType"/>
            <xs:element name="label" type="ns_p:LabelType" minOccurs="0"/>
            <xs:element name="description" type="ns_p:DescriptionType" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="incentiveDescriptionData" type="ns_p:IncentiveDescriptionDataType"/>
    <xs:complexType name="IncentiveDescriptionDataElementsType">
        <xs:sequence>
            <xs:element name="incentiveId" type="ns_p:ElementTagType" minOccurs="0"/>
            <xs:element name="incentiveType" type="ns_p:ElementTagType" minOccurs="0"/>
            <xs:element name="incentivePriority" type="ns_p:ElementTagType" minOccurs="0"/>
```



```
            <xs:element name="currency" type="ns_p:ElementTagType" minOccurs="0"/>
            <xs:element name="unit" minOccurs="0" type="ns_p:ElementTagType"/>
            <xs:element name="label" type="ns_p:ElementTagType" minOccurs="0"/>
            <xs:element name="description" type="ns_p:ElementTagType" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="incentiveDescriptionDataElements"
type="ns_p:IncentiveDescriptionDataElementsType"/>
    <xs:complexType name="IncentiveDescriptionListDataType">
        <xs:sequence>
            <xs:element maxOccurs="unbounded" minOccurs="0"
ref="ns_p:incentiveDescriptionData"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="incentiveDescriptionListData"
type="ns_p:IncentiveDescriptionListDataType"/>
    <xs:complexType name="IncentiveDescriptionListDataSelectorsType">
        <xs:sequence>
            <xs:element name="incentiveId" type="ns_p:IncentiveIdType" minOccurs="0"/>
            <xs:element name="incentiveType" type="ns_p:IncentiveTypeType" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="incentiveDescriptionListDataSelectors"
type="ns_p:IncentiveDescriptionListDataSelectorsType"/>
</xs:schema>
```

A.2.25 TaskManagement

File name: EEBus_SPINE_TS_TaskManagement.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
    Smart Premises Interoperable Neutral-Message Exchange (SPINE)
    Version 1.1.1
    2018-12-21
    Copyright (c) 2018 EEBus Initiative e.V.  All Rights Reserved.
    Source: https://www.eebus.org/en/specifications/
-->
<xs:schema xmlns:ns_p="http://docs.eebus.org/spine/xsd/v1"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://docs.eebus.org/spine/xsd/v1" version="1.1.1"
    blockDefault="#all" elementFormDefault="qualified">
    <xs:annotation>
        <xs:documentation>EEBus SPINE Specification schema. Copyright 2018 EEBus Initiative
e.V.  All rights reserved.</xs:documentation>
    </xs:annotation>
    <xs:include schemaLocation="EEBus_SPINE_TS_CommonDataTypes.xsd"/>
    <xs:include schemaLocation="EEBus_SPINE_TS_DirectControl.xsd"/>
    <xs:include schemaLocation="EEBus_SPINE_TS_HVAC.xsd"/>
    <xs:include schemaLocation="EEBus_SPINE_TS_LoadControl.xsd"/>
    <xs:include schemaLocation="EEBus_SPINE_TS_PowerSequences.xsd"/>
    <xs:simpleType name="TaskManagementJobIdType">
        <xs:restriction base="xs:unsignedInt"/>
    </xs:simpleType>
    <xs:simpleType name="TaskManagementJobStateType">
        <xs:union memberTypes="ns_p:DirectControlActivityStateEnumType
ns_p:HvacOverrunStatusEnumType ns_p:LoadControlEventStateEnumType
ns_p:PowerSequenceStateEnumType ns_p:EnumExtendType"/>
    </xs:simpleType>
    <xs:simpleType name="TaskManagementJobSourceType">
        <xs:union memberTypes="ns_p:TaskManagementJobSourceEnumType ns_p:EnumExtendType"/>
    </xs:simpleType>
    <xs:simpleType name="TaskManagementJobSourceEnumType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="internalMechanism"/>
            <xs:enumeration value="userInteraction"/>
            <xs:enumeration value="externalConfiguration"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:complexType name="TaskManagementDirectControlRelatedType"/>
    <xs:complexType name="TaskManagementDirectControlRelatedElementsType"/>
    <xs:complexType name="TaskManagementHvacRelatedType">
```



```
            <xs:sequence>
                <xs:element minOccurs="0" name="overrunId" type="ns_p:HvacOverrunIdType"/>
            </xs:sequence>
        </xs:complexType>
        <xs:complexType name="TaskManagementHvacRelatedElementsType">
            <xs:sequence>
                <xs:element minOccurs="0" name="overrunId" type="ns_p:ElementTagType"/>
            </xs:sequence>
        </xs:complexType>
        <xs:complexType name="TaskManagementLoadControlReleatedType">
            <xs:sequence>
                <xs:element minOccurs="0" name="eventId" type="ns_p:LoadControlEventIdType"/>
            </xs:sequence>
        </xs:complexType>
        <xs:complexType name="TaskManagementLoadControlReleatedElementsType">
            <xs:sequence>
                <xs:element minOccurs="0" name="eventId" type="ns_p:ElementTagType"/>
            </xs:sequence>
        </xs:complexType>
        <xs:complexType name="TaskManagementPowerSequencesRelatedType">
            <xs:sequence>
                <xs:element minOccurs="0" name="sequenceId" type="ns_p:PowerSequenceIdType"/>
            </xs:sequence>
        </xs:complexType>
        <xs:complexType name="TaskManagementPowerSequencesRelatedElementsType">
            <xs:sequence>
                <xs:element minOccurs="0" name="sequenceId" type="ns_p:ElementTagType"/>
            </xs:sequence>
        </xs:complexType>
        <xs:complexType name="TaskManagementSmartEnergyManagementPsRelatedType">
            <xs:sequence>
                <xs:element minOccurs="0" name="sequenceId" type="ns_p:PowerSequenceIdType"/>
            </xs:sequence>
        </xs:complexType>
        <xs:complexType name="TaskManagementSmartEnergyManagementPsRelatedElementsType">
            <xs:sequence>
                <xs:element minOccurs="0" name="sequenceId" type="ns_p:ElementTagType"/>
            </xs:sequence>
        </xs:complexType>
        <xs:complexType name="TaskManagementJobDataType">
            <xs:sequence>
                <xs:element minOccurs="0" name="jobId" type="ns_p:TaskManagementJobIdType"/>
                <xs:element name="timestamp" type="ns_p:AbsoluteOrRelativeTimeType"
    minOccurs="0"/>
                <xs:element minOccurs="0" name="jobState" type="ns_p:TaskManagementJobStateType"/>
                <xs:element name="elapsedTime" type="xs:duration" minOccurs="0"/>
                <xs:element name="remainingTime" type="xs:duration" minOccurs="0"/>
            </xs:sequence>
        </xs:complexType>
        <xs:element name="taskManagementJobData" type="ns_p:TaskManagementJobDataType"/>
        <xs:complexType name="TaskManagementJobDataElementsType">
            <xs:sequence>
                <xs:element minOccurs="0" name="jobId" type="ns_p:ElementTagType"/>
                <xs:element name="timestamp" minOccurs="0" type="ns_p:ElementTagType"/>
                <xs:element minOccurs="0" name="jobState" type="ns_p:ElementTagType"/>
                <xs:element name="elapsedTime" minOccurs="0" type="ns_p:ElementTagType"/>
                <xs:element name="remainingTime" minOccurs="0" type="ns_p:ElementTagType"/>
            </xs:sequence>
        </xs:complexType>
        <xs:element name="taskManagementJobDataElements"
    type="ns_p:TaskManagementJobDataElementsType"/>
        <xs:complexType name="TaskManagementJobListDataType">
            <xs:sequence>
                <xs:element maxOccurs="unbounded" minOccurs="0" ref="ns_p:taskManagementJobData"/>
            </xs:sequence>
        </xs:complexType>
        <xs:element name="taskManagementJobListData" type="ns_p:TaskManagementJobListDataType"/>
        <xs:complexType name="TaskManagementJobListDataSelectorsType">
            <xs:sequence>
                <xs:element minOccurs="0" name="jobId" type="ns_p:TaskManagementJobIdType"/>
                <xs:element minOccurs="0" name="jobState" type="ns_p:TaskManagementJobStateType"/>
            </xs:sequence>
        </xs:complexType>
        <xs:element name="taskManagementJobListDataSelectors"
    type="ns_p:TaskManagementJobListDataSelectorsType"/>
        <xs:complexType name="TaskManagementJobRelationDataType">
            <xs:sequence>
                <xs:element minOccurs="0" name="jobId" type="ns_p:TaskManagementJobIdType"/>
```



```
            <xs:element minOccurs="0" name="directControlRelated"
type="ns_p:TaskManagementDirectControlRelatedType"/>
            <xs:element minOccurs="0" name="hvacRelated"
type="ns_p:TaskManagementHvacRelatedType"/>
            <xs:element minOccurs="0" name="loadControlReleated"
type="ns_p:TaskManagementLoadControlReleatedType"/>
            <xs:element minOccurs="0" name="powerSequencesRelated"
type="ns_p:TaskManagementPowerSequencesRelatedType"/>
            <xs:element minOccurs="0" name="smartEnergyManagementPsRelated"
type="ns_p:TaskManagementSmartEnergyManagementPsRelatedType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="taskManagementJobRelationData"
type="ns_p:TaskManagementJobRelationDataType"/>
    <xs:complexType name="TaskManagementJobRelationDataElementsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="jobId" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="directControlRelated"
type="ns_p:TaskManagementDirectControlRelatedElementsType"/>
            <xs:element minOccurs="0" name="hvacRelated"
type="ns_p:TaskManagementHvacRelatedElementsType"/>
            <xs:element minOccurs="0" name="loadControlReleated"
type="ns_p:TaskManagementLoadControlReleatedElementsType"/>
            <xs:element minOccurs="0" name="powerSequencesRelated"
type="ns_p:TaskManagementPowerSequencesRelatedElementsType"/>
            <xs:element minOccurs="0" name="smartEnergyManagementPsRelated"
type="ns_p:TaskManagementSmartEnergyManagementPsRelatedElementsType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="taskManagementJobRelationDataElements"
type="ns_p:TaskManagementJobRelationDataElementsType"/>
    <xs:complexType name="TaskManagementJobRelationListDataType">
        <xs:sequence>
            <xs:element maxOccurs="unbounded" minOccurs="0"
ref="ns_p:taskManagementJobRelationData"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="taskManagementJobRelationListData"
type="ns_p:TaskManagementJobRelationListDataType"/>
    <xs:complexType name="TaskManagementJobRelationListDataSelectorsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="jobId" type="ns_p:TaskManagementJobIdType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="taskManagementJobRelationListDataSelectors"
type="ns_p:TaskManagementJobRelationListDataSelectorsType"/>
    <xs:complexType name="TaskManagementJobDescriptionDataType">
        <xs:sequence>
            <xs:element minOccurs="0" name="jobId" type="ns_p:TaskManagementJobIdType"/>
            <xs:element minOccurs="0" name="jobSource"
type="ns_p:TaskManagementJobSourceType"/>
            <xs:element minOccurs="0" name="label" type="ns_p:LabelType"/>
            <xs:element minOccurs="0" name="description" type="ns_p:DescriptionType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="taskManagementJobDescriptionData"
type="ns_p:TaskManagementJobDescriptionDataType"/>
    <xs:complexType name="TaskManagementJobDescriptionDataElementsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="jobId" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="jobSource" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="label" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="description" type="ns_p:ElementTagType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="taskManagementJobDescriptionDataElements"
type="ns_p:TaskManagementJobDescriptionDataElementsType"/>
    <xs:complexType name="TaskManagementJobDescriptionListDataType">
        <xs:sequence>
            <xs:element maxOccurs="unbounded" minOccurs="0"
ref="ns_p:taskManagementJobDescriptionData"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="taskManagementJobDescriptionListData"
type="ns_p:TaskManagementJobDescriptionListDataType"/>
    <xs:complexType name="TaskManagementJobDescriptionListDataSelectorsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="jobId" type="ns_p:TaskManagementJobIdType"/>
```



```
            <xs:element minOccurs="0" name="jobSource"
type="ns_p:TaskManagementJobSourceType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="taskManagementJobDescriptionListDataSelectors"
type="ns_p:TaskManagementJobDescriptionListDataSelectorsType"/>
    <xs:complexType name="TaskManagementOverviewDataType">
        <xs:sequence>
            <xs:element minOccurs="0" name="remoteControllable" type="xs:boolean"/>
            <xs:element minOccurs="0" name="jobsActive" type="xs:boolean"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="taskManagementOverviewData" type="ns_p:TaskManagementOverviewDataType"/>
    <xs:complexType name="TaskManagementOverviewDataElementsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="remoteControllable" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="jobsActive" type="ns_p:ElementTagType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="taskManagementOverviewDataElements"
type="ns_p:TaskManagementOverviewDataElementsType"/>
</xs:schema>
```

A.2.26 Threshold

File name: EEBus_SPINE_TS_Threshold.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
    Smart Premises Interoperable Neutral-Message Exchange (SPINE)
    Version 1.1.1
    2018-12-21
    Copyright (c) 2018 EEBus Initiative e.V.  All Rights Reserved.
    Source: https://www.eebus.org/en/specifications/
-->
<xs:schema xmlns:ns_p="http://docs.eebus.org/spine/xsd/v1"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://docs.eebus.org/spine/xsd/v1" version="1.1.1"
    blockDefault="#all" elementFormDefault="qualified">
    <xs:annotation>
        <xs:documentation>EEBus SPINE Specification schema. Copyright 2018 EEBus Initiative
e.V.  All rights reserved.</xs:documentation>
    </xs:annotation>
    <xs:include schemaLocation="EEBus_SPINE_TS_CommonDataTypes.xsd"/>
    <xs:simpleType name="ThresholdIdType">
        <xs:restriction base="xs:unsignedInt"/>
    </xs:simpleType>
    <xs:simpleType name="ThresholdTypeType">
        <xs:union memberTypes="ns_p:ThresholdTypeEnumType ns_p:EnumExtendType"/>
    </xs:simpleType>
    <xs:simpleType name="ThresholdTypeEnumType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="goodAbove"/>
            <xs:enumeration value="badAbove"/>
            <xs:enumeration value="goodBelow"/>
            <xs:enumeration value="badBelow"/>
            <xs:enumeration value="minValueThreshold"/>
            <xs:enumeration value="maxValueThreshold"/>
            <xs:enumeration value="minValueThresholdExtreme"/>
            <xs:enumeration value="maxValueThresholdExtreme"/>
            <xs:enumeration value="sagThreshold"/>
            <xs:enumeration value="swellThreshold"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:complexType name="ThresholdDataType">
        <xs:sequence>
            <xs:element name="thresholdId" minOccurs="0" type="ns_p:ThresholdIdType"/>
            <xs:element minOccurs="0" name="thresholdValue" type="ns_p:ScaledNumberType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="thresholdData" type="ns_p:ThresholdDataType"/>
    <xs:complexType name="ThresholdDataElementsType">
```



```
            <xs:sequence>
                <xs:element name="thresholdId" minOccurs="0" type="ns_p:ElementTagType"/>
                <xs:element minOccurs="0" name="thresholdValue"
type="ns_p:ScaledNumberElementsType"/>
            </xs:sequence>
        </xs:complexType>
        <xs:element name="thresholdDataElements" type="ns_p:ThresholdDataElementsType"/>
        <xs:complexType name="ThresholdListDataType">
            <xs:sequence>
                <xs:element maxOccurs="unbounded" minOccurs="0" ref="ns_p:thresholdData"/>
            </xs:sequence>
        </xs:complexType>
        <xs:element name="thresholdListData" type="ns_p:ThresholdListDataType"/>
        <xs:complexType name="ThresholdListDataSelectorsType">
            <xs:sequence>
                <xs:element name="thresholdId" minOccurs="0" type="ns_p:ThresholdIdType"/>
            </xs:sequence>
        </xs:complexType>
        <xs:element name="thresholdListDataSelectors" type="ns_p:ThresholdListDataSelectorsType"/>
        <xs:complexType name="ThresholdConstraintsDataType">
            <xs:sequence>
                <xs:element minOccurs="0" name="thresholdId" type="ns_p:ThresholdIdType"/>
                <xs:element minOccurs="0" name="thresholdRangeMin" type="ns_p:ScaledNumberType"/>
                <xs:element minOccurs="0" name="thresholdRangeMax" type="ns_p:ScaledNumberType"/>
                <xs:element minOccurs="0" name="thresholdStepSize" type="ns_p:ScaledNumberType"/>
            </xs:sequence>
        </xs:complexType>
        <xs:element name="thresholdConstraintsData" type="ns_p:ThresholdConstraintsDataType"/>
        <xs:complexType name="ThresholdConstraintsDataElementsType">
            <xs:sequence>
                <xs:element minOccurs="0" name="thresholdId" type="ns_p:ElementTagType"/>
                <xs:element minOccurs="0" name="thresholdRangeMin"
type="ns_p:ScaledNumberElementsType"/>
                <xs:element minOccurs="0" name="thresholdRangeMax"
type="ns_p:ScaledNumberElementsType"/>
                <xs:element minOccurs="0" name="thresholdStepSize"
type="ns_p:ScaledNumberElementsType"/>
            </xs:sequence>
        </xs:complexType>
        <xs:element name="thresholdConstraintsDataElements"
type="ns_p:ThresholdConstraintsDataElementsType"/>
        <xs:complexType name="ThresholdConstraintsListDataType">
            <xs:sequence>
                <xs:element maxOccurs="unbounded" minOccurs="0"
ref="ns_p:thresholdConstraintsData"/>
            </xs:sequence>
        </xs:complexType>
        <xs:element name="thresholdConstraintsListData"
type="ns_p:ThresholdConstraintsListDataType"/>
        <xs:complexType name="ThresholdConstraintsListDataSelectorsType">
            <xs:sequence>
                <xs:element name="thresholdId" minOccurs="0" type="ns_p:ThresholdIdType"/>
            </xs:sequence>
        </xs:complexType>
        <xs:element name="thresholdConstraintsListDataSelectors"
type="ns_p:ThresholdConstraintsListDataSelectorsType"/>
        <xs:complexType name="ThresholdDescriptionDataType">
            <xs:sequence>
                <xs:element name="thresholdId" minOccurs="0" type="ns_p:ThresholdIdType"/>
                <xs:element name="thresholdType" minOccurs="0" type="ns_p:ThresholdTypeType"/>
                <xs:element minOccurs="0" name="unit" type="ns_p:UnitOfMeasurementType"/>
                <xs:element minOccurs="0" name="scopeType" type="ns_p:ScopeTypeType"/>
                <xs:element minOccurs="0" name="label" type="ns_p:LabelType"/>
                <xs:element minOccurs="0" name="description" type="ns_p:DescriptionType"/>
            </xs:sequence>
        </xs:complexType>
        <xs:element name="thresholdDescriptionData" type="ns_p:ThresholdDescriptionDataType"/>
        <xs:complexType name="ThresholdDescriptionDataElementsType">
            <xs:sequence>
                <xs:element name="thresholdId" minOccurs="0" type="ns_p:ElementTagType"/>
                <xs:element name="thresholdType" minOccurs="0" type="ns_p:ElementTagType"/>
                <xs:element minOccurs="0" name="unit" type="ns_p:ElementTagType"/>
                <xs:element minOccurs="0" name="scopeType" type="ns_p:ElementTagType"/>
                <xs:element minOccurs="0" name="label" type="ns_p:ElementTagType"/>
                <xs:element minOccurs="0" name="description" type="ns_p:ElementTagType"/>
            </xs:sequence>
        </xs:complexType>
```



```
    <xs:element name="thresholdDescriptionDataElements"
type="ns_p:ThresholdDescriptionDataElementsType"/>
    <xs:complexType name="ThresholdDescriptionListDataType">
        <xs:sequence>
            <xs:element maxOccurs="unbounded" minOccurs="0"
ref="ns_p:thresholdDescriptionData"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="thresholdDescriptionListData"
type="ns_p:ThresholdDescriptionListDataType"/>
    <xs:complexType name="ThresholdDescriptionListDataSelectorsType">
        <xs:sequence>
            <xs:element name="thresholdId" minOccurs="0" type="ns_p:ThresholdIdType"/>
            <xs:element minOccurs="0" name="scopeType" type="ns_p:ScopeTypeType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="thresholdDescriptionListDataSelectors"
type="ns_p:ThresholdDescriptionListDataSelectorsType"/>
</xs:schema>
```

A.2.27 TimeInformation

File name: EEBus_SPINE_TS_TimeInformation.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
    Smart Premises Interoperable Neutral-Message Exchange (SPINE)
    Version 1.1.1
    2018-12-21
    Copyright (c) 2018 EEBus Initiative e.V.  All Rights Reserved.
    Source: https://www.eebus.org/en/specifications/
-->
<xs:schema xmlns:ns_p="http://docs.eebus.org/spine/xsd/v1"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://docs.eebus.org/spine/xsd/v1" version="1.1.1" blockDefault="#all"
elementFormDefault="qualified">
    <xs:annotation>
        <xs:documentation>EEBus SPINE Specification schema. Copyright 2018 EEBus Initiative
e.V.  All rights reserved.</xs:documentation>
    </xs:annotation>
    <xs:include schemaLocation="EEBus_SPINE_TS_CommonDataTypes.xsd"/>
    <xs:complexType name="TimeInformationDataType">
        <xs:sequence>
            <xs:element minOccurs="0" name="utc" type="xs:dateTime"/>
            <xs:element minOccurs="0" name="utcOffset" type="xs:duration"/>
            <xs:element minOccurs="0" name="dayOfWeek" type="ns_p:DayOfWeekType"/>
            <xs:element minOccurs="0" name="calendarWeek" type="ns_p:CalendarWeekType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="timeInformationData" type="ns_p:TimeInformationDataType"/>
    <xs:complexType name="TimeInformationDataElementsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="utc" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="utcOffset" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="dayOfWeek" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="calendarWeek" type="ns_p:ElementTagType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="timeInformationDataElements"
type="ns_p:TimeInformationDataElementsType"/>
    <xs:complexType name="TimeDistributorDataType">
        <xs:sequence>
            <xs:element minOccurs="0" name="isTimeDistributor" type="xs:boolean"/>
            <xs:element minOccurs="0" name="distributorPriority" type="xs:unsignedInt"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="timeDistributorData" type="ns_p:TimeDistributorDataType"/>
    <xs:complexType name="TimeDistributorDataElementsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="isTimeDistributor" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="distributorPriority" type="ns_p:ElementTagType"/>
        </xs:sequence>
```



```
    </xs:complexType>
    <xs:element name="timeDistributorDataElements"
type="ns_p:TimeDistributorDataElementsType"/>
    <xs:complexType name="TimePrecisionDataType">
        <xs:sequence>
            <xs:element minOccurs="0" name="isSynchronised" type="xs:boolean"/>
            <xs:element minOccurs="0" name="lastSyncAt" type="xs:dateTime"/>
            <xs:element minOccurs="0" name="clockDrift" type="xs:integer"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="timePrecisionData" type="ns_p:TimePrecisionDataType"/>
    <xs:complexType name="TimePrecisionDataElementsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="isSynchronised" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="lastSyncAt" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="clockDrift" type="ns_p:ElementTagType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="timePrecisionDataElements" type="ns_p:TimePrecisionDataElementsType"/>
    <xs:complexType name="TimeDistributorEnquiryCallType"/>
    <xs:element name="timeDistributorEnquiryCall" type="ns_p:TimeDistributorEnquiryCallType"/>
    <xs:complexType name="TimeDistributorEnquiryCallElementsType"/>
    <xs:element name="timeDistributorEnquiryCallElements"
type="ns_p:TimeDistributorEnquiryCallElementsType"/>
</xs:schema>
```

A.2.28 TimeSeries

File name: EEBus_SPINE_TS_TimeSeries.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
    Smart Premises Interoperable Neutral-Message Exchange (SPINE)
    Version 1.1.1
    2018-12-21
    Copyright (c) 2018 EEBus Initiative e.V.  All Rights Reserved.
    Source: https://www.eebus.org/en/specifications/
-->
<xs:schema xmlns:ns_p="http://docs.eebus.org/spine/xsd/v1"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://docs.eebus.org/spine/xsd/v1" version="1.1.1"
    blockDefault="#all" elementFormDefault="qualified">
    <xs:annotation>
        <xs:documentation>EEBus SPINE Specification schema. Copyright 2018 EEBus Initiative
e.V.  All rights reserved.</xs:documentation>
    </xs:annotation>
    <xs:include schemaLocation="EEBus_SPINE_TS_CommonDataTypes.xsd"/>
    <xs:include schemaLocation="EEBus_SPINE_TS_Measurement.xsd"/>
    <xs:simpleType name="TimeSeriesIdType">
        <xs:restriction base="xs:unsignedInt"/>
    </xs:simpleType>
    <xs:simpleType name="TimeSeriesSlotIdType">
        <xs:restriction base="xs:unsignedInt"/>
    </xs:simpleType>
    <xs:simpleType name="TimeSeriesSlotCountType">
        <xs:restriction base="ns_p:TimeSeriesSlotIdType"/>
    </xs:simpleType>
    <xs:simpleType name="TimeSeriesTypeType">
        <xs:union memberTypes="ns_p:TimeSeriesTypeEnumType ns_p:EnumExtendType"/>
    </xs:simpleType>
    <xs:simpleType name="TimeSeriesTypeEnumType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="plan"/>
            <xs:enumeration value="singleDemand"/>
            <xs:enumeration value="constraints"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:complexType name="TimeSeriesSlotType">
        <xs:sequence>
            <xs:element minOccurs="0" name="timeSeriesSlotId"
type="ns_p:TimeSeriesSlotIdType"/>
            <xs:element minOccurs="0" name="timePeriod" type="ns_p:TimePeriodType"/>
```

```
            <xs:element minOccurs="0" name="duration" type="xs:duration"/>
            <xs:element minOccurs="0" name="recurrenceInformation"
type="ns_p:AbsoluteOrRecurringTimeType"/>
            <xs:element minOccurs="0" name="value" type="ns_p:ScaledNumberType"/>
            <xs:element minOccurs="0" name="minValue" type="ns_p:ScaledNumberType"/>
            <xs:element minOccurs="0" name="maxValue" type="ns_p:ScaledNumberType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="TimeSeriesSlotElementsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="timeSeriesSlotId" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="timePeriod" type="ns_p:TimePeriodElementsType"/>
            <xs:element minOccurs="0" name="duration" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="recurrenceInformation"
type="ns_p:AbsoluteOrRecurringTimeElementsType"/>
            <xs:element minOccurs="0" name="value" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="minValue" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="maxValue" type="ns_p:ElementTagType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="TimeSeriesDataType">
        <xs:sequence>
            <xs:element minOccurs="0" name="timeSeriesId" type="ns_p:TimeSeriesIdType"/>
            <xs:element minOccurs="0" name="timePeriod" type="ns_p:TimePeriodType"/>
            <xs:element maxOccurs="unbounded" minOccurs="0" name="timeSeriesSlot"
type="ns_p:TimeSeriesSlotType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="timeSeriesData" type="ns_p:TimeSeriesDataType"/>
    <xs:complexType name="TimeSeriesDataElementsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="timeSeriesId" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="timePeriod" type="ns_p:TimePeriodElementsType"/>
            <xs:element minOccurs="0" name="timeSeriesSlot"
type="ns_p:TimeSeriesSlotElementsType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="timeSeriesDataElements" type="ns_p:TimeSeriesDataElementsType"/>
    <xs:complexType name="TimeSeriesListDataType">
        <xs:sequence>
            <xs:element maxOccurs="unbounded" minOccurs="0" ref="ns_p:timeSeriesData"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="timeSeriesListData" type="ns_p:TimeSeriesListDataType"/>
    <xs:complexType name="TimeSeriesListDataSelectorsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="timeSeriesId" type="ns_p:TimeSeriesIdType"/>
            <xs:element minOccurs="0" name="timeSeriesSlotId"
type="ns_p:TimeSeriesSlotIdType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="timeSeriesListDataSelectors"
type="ns_p:TimeSeriesListDataSelectorsType"/>
    <xs:complexType name="TimeSeriesDescriptionDataType">
        <xs:sequence>
            <xs:element minOccurs="0" name="timeSeriesId" type="ns_p:TimeSeriesIdType"/>
            <xs:element minOccurs="0" name="timeSeriesType" type="ns_p:TimeSeriesTypeType"/>
            <xs:element minOccurs="0" name="timeSeriesWriteable" type="xs:boolean"/>
            <xs:element minOccurs="0" name="updateRequired" type="xs:boolean"/>
            <xs:element minOccurs="0" name="measurementId" type="ns_p:MeasurementIdType"/>
            <xs:element minOccurs="0" name="currency" type="ns_p:CurrencyType"/>
            <xs:element minOccurs="0" name="unit" type="ns_p:UnitOfMeasurementType"/>
            <xs:element minOccurs="0" name="label" type="ns_p:LabelType"/>
            <xs:element minOccurs="0" name="description" type="ns_p:DescriptionType"/>
            <xs:element minOccurs="0" name="scopeType" type="ns_p:ScopeTypeType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="timeSeriesDescriptionData" type="ns_p:TimeSeriesDescriptionDataType"/>
    <xs:complexType name="TimeSeriesDescriptionDataElementsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="timeSeriesId" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="timeSeriesType" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="timeSeriesWriteable" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="updateRequired" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="measurementId" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="currency" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="unit" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="label" type="ns_p:ElementTagType"/>
```



```
            <xs:element minOccurs="0" name="description" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="scopeType" type="ns_p:ElementTagType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="timeSeriesDescriptionDataElements"
type="ns_p:TimeSeriesDescriptionDataElementsType"/>
    <xs:complexType name="TimeSeriesDescriptionListDataType">
        <xs:sequence>
            <xs:element maxOccurs="unbounded" minOccurs="0"
ref="ns_p:timeSeriesDescriptionData"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="timeSeriesDescriptionListData"
type="ns_p:TimeSeriesDescriptionListDataType"/>
    <xs:complexType name="TimeSeriesDescriptionListDataSelectorsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="timeSeriesId" type="ns_p:TimeSeriesIdType"/>
            <xs:element minOccurs="0" name="timeSeriesType" type="ns_p:TimeSeriesTypeType"/>
            <xs:element minOccurs="0" name="measurementId" type="ns_p:MeasurementIdType"/>
            <xs:element minOccurs="0" name="scopeType" type="ns_p:ScopeTypeType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="timeSeriesDescriptionListDataSelectors"
type="ns_p:TimeSeriesDescriptionListDataSelectorsType"/>
    <xs:complexType name="TimeSeriesConstraintsDataType">
        <xs:sequence>
            <xs:element minOccurs="0" name="timeSeriesId" type="ns_p:TimeSeriesIdType"/>
            <xs:element minOccurs="0" name="slotCountMin"
type="ns_p:TimeSeriesSlotCountType"/>
            <xs:element minOccurs="0" name="slotCountMax"
type="ns_p:TimeSeriesSlotCountType"/>
            <xs:element minOccurs="0" name="slotDurationMin" type="xs:duration"/>
            <xs:element minOccurs="0" name="slotDurationMax" type="xs:duration"/>
            <xs:element minOccurs="0" name="slotDurationStepSize" type="xs:duration"/>
            <xs:element minOccurs="0" name="earliestTimeSeriesStartTime"
type="ns_p:AbsoluteOrRelativeTimeType"/>
            <xs:element minOccurs="0" name="latestTimeSeriesEndTime"
type="ns_p:AbsoluteOrRelativeTimeType"/>
            <xs:element minOccurs="0" name="slotValueMin" type="ns_p:ScaledNumberType"/>
            <xs:element minOccurs="0" name="slotValueMax" type="ns_p:ScaledNumberType"/>
            <xs:element minOccurs="0" name="slotValueStepSize" type="ns_p:ScaledNumberType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="timeSeriesConstraintsData" type="ns_p:TimeSeriesConstraintsDataType"/>
    <xs:complexType name="TimeSeriesConstraintsDataElementsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="timeSeriesId" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="slotCountMin" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="slotCountMax" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="slotDurationMin" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="slotDurationMax" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="slotDurationStepSize" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="earliestTimeSeriesStartTime"
type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="latestTimeSeriesEndTime"
type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="slotValueMin"
type="ns_p:ScaledNumberElementsType"/>
            <xs:element minOccurs="0" name="slotValueMax"
type="ns_p:ScaledNumberElementsType"/>
            <xs:element minOccurs="0" name="slotValueStepSize"
type="ns_p:ScaledNumberElementsType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="timeSeriesConstraintsDataElements"
type="ns_p:TimeSeriesConstraintsDataElementsType"/>
    <xs:complexType name="TimeSeriesConstraintsListDataType">
        <xs:sequence>
            <xs:element maxOccurs="unbounded" minOccurs="0"
ref="ns_p:timeSeriesConstraintsData"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="timeSeriesConstraintsListData"
type="ns_p:TimeSeriesConstraintsListDataType"/>
    <xs:complexType name="TimeSeriesConstraintsListDataSelectorsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="timeSeriesId" type="ns_p:TimeSeriesIdType"/>
        </xs:sequence>
```



```
    </xs:complexType>
    <xs:element name="timeSeriesConstraintsListDataSelectors"
type="ns_p:TimeSeriesConstraintsListDataSelectorsType"/>
</xs:schema>
```

A.2.29 TimeTable

File name: EEBus_SPINE_TS_TimeTable.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
    Smart Premises Interoperable Neutral-Message Exchange (SPINE)
    Version 1.1.1
    2018-12-21
    Copyright (c) 2018 EEBus Initiative e.V.  All Rights Reserved.
    Source: https://www.eebus.org/en/specifications/
-->
<xs:schema xmlns:ns_p="http://docs.eebus.org/spine/xsd/v1"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://docs.eebus.org/spine/xsd/v1" version="1.1.1"
    blockDefault="#all" elementFormDefault="qualified">
    <xs:annotation>
        <xs:documentation>EEBus SPINE Specification schema. Copyright 2018 EEBus Initiative
e.V.  All rights reserved.</xs:documentation>
    </xs:annotation>
    <xs:include schemaLocation="EEBus_SPINE_TS_CommonDataTypes.xsd"/>
    <xs:simpleType name="TimeTableIdType">
        <xs:restriction base="xs:unsignedInt"/>
    </xs:simpleType>
    <xs:simpleType name="TimeSlotIdType">
        <xs:restriction base="xs:unsignedInt"/>
    </xs:simpleType>
    <xs:simpleType name="TimeSlotCountType">
        <xs:restriction base="ns_p:TimeSlotIdType"/>
    </xs:simpleType>
    <xs:simpleType name="TimeSlotTimeModeType">
        <xs:union memberTypes="ns_p:TimeSlotTimeModeEnumType ns_p:EnumExtendType"/>
    </xs:simpleType>
    <xs:simpleType name="TimeSlotTimeModeEnumType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="absolute"/>
            <xs:enumeration value="recurring"/>
            <xs:enumeration value="both"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:complexType name="TimeTableDataType">
        <xs:sequence>
            <xs:element minOccurs="0" name="timeTableId" type="ns_p:TimeTableIdType"/>
            <xs:element minOccurs="0" name="timeSlotId" type="ns_p:TimeSlotIdType"/>
            <xs:element name="recurrenceInformation" minOccurs="0"
type="ns_p:RecurrenceInformationType"/>
            <xs:element minOccurs="0" name="startTime"
type="ns_p:AbsoluteOrRecurringTimeType"/>
            <xs:element minOccurs="0" name="endTime" type="ns_p:AbsoluteOrRecurringTimeType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="timeTableData" type="ns_p:TimeTableDataType"/>
    <xs:complexType name="TimeTableDataElementsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="timeTableId" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="timeSlotId" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="recurrenceInformation"
type="ns_p:RecurrenceInformationElementsType"/>
            <xs:element minOccurs="0" name="startTime"
type="ns_p:AbsoluteOrRecurringTimeElementsType"/>
            <xs:element minOccurs="0" name="endTime"
type="ns_p:AbsoluteOrRecurringTimeElementsType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="timeTableDataElements" type="ns_p:TimeTableDataElementsType"/>
    <xs:complexType name="TimeTableListDataType">
        <xs:sequence>
```



```
            <xs:element maxOccurs="unbounded" minOccurs="0" ref="ns_p:timeTableData"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="timeTableListData" type="ns_p:TimeTableListDataType"/>
    <xs:complexType name="TimeTableListDataSelectorsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="timeTableId" type="ns_p:TimeTableIdType"/>
            <xs:element minOccurs="0" name="timeSlotId" type="ns_p:TimeSlotIdType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="timeTableListDataSelectors" type="ns_p:TimeTableListDataSelectorsType"/>
    <xs:complexType name="TimeTableConstraintsDataType">
        <xs:sequence>
            <xs:element minOccurs="0" name="timeTableId" type="xs:unsignedInt"/>
            <xs:element minOccurs="0" name="slotCountMin" type="ns_p:TimeSlotCountType"/>
            <xs:element minOccurs="0" name="slotCountMax" type="ns_p:TimeSlotCountType"/>
            <xs:element minOccurs="0" name="slotDurationMin" type="xs:duration"/>
            <xs:element minOccurs="0" name="slotDurationMax" type="xs:duration"/>
            <xs:element minOccurs="0" name="slotDurationStepSize" type="xs:duration"/>
            <xs:element minOccurs="0" name="slotShiftStepSize" type="xs:duration"/>
            <xs:element minOccurs="0" name="firstSlotBeginsAt" type="xs:time"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="timeTableConstraintsData" type="ns_p:TimeTableConstraintsDataType"/>
    <xs:complexType name="TimeTableConstraintsDataElementsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="timeTableId" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="slotCountMin" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="slotCountMax" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="slotDurationMin" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="slotDurationMax" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="slotDurationStepSize" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="slotShiftStepSize" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="firstSlotBeginsAt" type="ns_p:ElementTagType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="timeTableConstraintsDataElements"
type="ns_p:TimeTableConstraintsDataElementsType"/>
    <xs:complexType name="TimeTableConstraintsListDataType">
        <xs:sequence>
            <xs:element maxOccurs="unbounded" minOccurs="0"
ref="ns_p:timeTableConstraintsData"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="timeTableConstraintsListData"
type="ns_p:TimeTableConstraintsListDataType"/>
    <xs:complexType name="TimeTableConstraintsListDataSelectorsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="timeTableId" type="ns_p:TimeTableIdType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="timeTableConstraintsListDataSelectors"
type="ns_p:TimeTableConstraintsListDataSelectorsType"/>
    <xs:complexType name="TimeTableDescriptionDataType">
        <xs:sequence>
            <xs:element minOccurs="0" name="timeTableId" type="xs:unsignedInt"/>
            <xs:element minOccurs="0" name="timeSlotCountChangeable" type="xs:boolean"/>
            <xs:element minOccurs="0" name="timeSlotTimesChangeable" type="xs:boolean"/>
            <xs:element minOccurs="0" name="timeSlotTimeMode"
type="ns_p:TimeSlotTimeModeType"/>
            <xs:element minOccurs="0" name="label" type="ns_p:LabelType"/>
            <xs:element minOccurs="0" name="description" type="ns_p:DescriptionType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="timeTableDescriptionData" type="ns_p:TimeTableDescriptionDataType"/>
    <xs:complexType name="TimeTableDescriptionDataElementsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="timeTableId" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="timeSlotCountChangeable"
type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="timeSlotTimesChangeable"
type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="timeSlotTimeMode" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="label" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="description" type="ns_p:ElementTagType"/>
        </xs:sequence>
    </xs:complexType>
```



```
    <xs:element name="timeTableDescriptionDataElements"
type="ns_p:TimeTableDescriptionDataElementsType"/>
    <xs:complexType name="TimeTableDescriptionListDataType">
        <xs:sequence>
            <xs:element maxOccurs="unbounded" minOccurs="0"
ref="ns_p:timeTableDescriptionData"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="timeTableDescriptionListData"
type="ns_p:TimeTableDescriptionListDataType"/>
    <xs:complexType name="TimeTableDescriptionListDataSelectorsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="timeTableId" type="ns_p:TimeTableIdType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="timeTableDescriptionListDataSelectors"
type="ns_p:TimeTableDescriptionListDataSelectorsType"/>
</xs:schema>
```

A.2.30 UseCaseInformation

File name: EEBus_SPINE_TS_UseCaseInformation.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
    Smart Premises Interoperable Neutral-Message Exchange (SPINE)
    Version 1.1.1
    2018-12-21
    Copyright (c) 2018 EEBus Initiative e.V.  All Rights Reserved.
    Source: https://www.eebus.org/en/specifications/
-->
<xs:schema xmlns:ns_p="http://docs.eebus.org/spine/xsd/v1"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://docs.eebus.org/spine/xsd/v1" version="1.1.1"
    blockDefault="#all" elementFormDefault="qualified">
    <xs:annotation>
        <xs:documentation>EEBus SPINE Specification schema. Copyright 2018 EEBus Initiative
e.V.  All rights reserved.</xs:documentation>
    </xs:annotation>
    <xs:include schemaLocation="EEBus_SPINE_TS_CommonDataTypes.xsd"/>
    <xs:include schemaLocation="EEBus_SPINE_TS_Version.xsd"/>
    <xs:simpleType name="UseCaseActorType">
        <xs:union memberTypes="ns_p:UseCaseActorEnumType ns_p:EnumExtendType"/>
    </xs:simpleType>
    <xs:simpleType name="UseCaseActorEnumType">
        <xs:restriction base="xs:string"/>
    </xs:simpleType>
    <xs:simpleType name="UseCaseNameType">
        <xs:union memberTypes="ns_p:UseCaseNameEnumType ns_p:EnumExtendType"/>
    </xs:simpleType>
    <xs:simpleType name="UseCaseNameEnumType">
        <xs:restriction base="xs:string"/>
    </xs:simpleType>
    <xs:simpleType name="UseCaseScenarioSupportType">
        <xs:restriction base="xs:unsignedInt"/>
    </xs:simpleType>
    <xs:complexType name="UseCaseSupportType">
        <xs:sequence>
            <xs:element minOccurs="0" name="useCaseName" type="ns_p:UseCaseNameType"/>
            <xs:element minOccurs="0" name="useCaseVersion"
type="ns_p:SpecificationVersionType"/>
            <xs:element minOccurs="0" name="useCaseAvailable" type="xs:boolean"/>
            <xs:element maxOccurs="unbounded" minOccurs="0" name="scenarioSupport"
type="ns_p:UseCaseScenarioSupportType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="UseCaseSupportElementsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="useCaseName" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="useCaseVersion" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="useCaseAvailable" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="scenarioSupport" type="ns_p:ElementTagType"/>
```

```
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="UseCaseSupportSelectorsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="useCaseName" type="ns_p:UseCaseNameType"/>
            <xs:element minOccurs="0" name="useCaseVersion"
type="ns_p:SpecificationVersionType"/>
            <xs:element minOccurs="0" name="scenarioSupport"
type="ns_p:UseCaseScenarioSupportType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="UseCaseInformationDataType">
        <xs:sequence>
            <xs:element minOccurs="0" name="address" type="ns_p:FeatureAddressType"/>
            <xs:element minOccurs="0" name="actor" type="ns_p:UseCaseActorType"/>
            <xs:element maxOccurs="unbounded" minOccurs="0" name="useCaseSupport"
type="ns_p:UseCaseSupportType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="useCaseInformationData" type="ns_p:UseCaseInformationDataType"/>
    <xs:complexType name="UseCaseInformationDataElementsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="address" type="ns_p:FeatureAddressElementsType"/>
            <xs:element minOccurs="0" name="actor" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="useCaseSupport"
type="ns_p:UseCaseSupportElementsType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="useCaseInformationDataElements"
type="ns_p:UseCaseInformationDataElementsType"/>
    <xs:complexType name="UseCaseInformationListDataType">
        <xs:sequence>
            <xs:element maxOccurs="unbounded" minOccurs="0"
ref="ns_p:useCaseInformationData"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="useCaseInformationListData" type="ns_p:UseCaseInformationListDataType"/>
    <xs:complexType name="UseCaseInformationListDataSelectorsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="address" type="ns_p:FeatureAddressType"/>
            <xs:element minOccurs="0" name="actor" type="ns_p:UseCaseActorType"/>
            <xs:element minOccurs="0" name="useCaseSupport"
type="ns_p:UseCaseSupportSelectorsType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="useCaseInformationListDataSelectors"
type="ns_p:UseCaseInformationListDataSelectorsType"/>
</xs:schema>
```

A.2.31 Version

File name: EEBus_SPINE_TS_Version.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
    Smart Premises Interoperable Neutral-Message Exchange (SPINE)
    Version 1.1.1
    2018-12-21
    Copyright (c) 2018 EEBus Initiative e.V.  All Rights Reserved.
    Source: https://www.eebus.org/en/specifications/
-->
<xs:schema xmlns:ns_p="http://docs.eebus.org/spine/xsd/v1"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://docs.eebus.org/spine/xsd/v1" version="1.1.1" blockDefault="#all"
elementFormDefault="qualified">
    <xs:annotation>
        <xs:documentation>EEBus SPINE Specification schema. Copyright 2018 EEBus Initiative
e.V.  All rights reserved.</xs:documentation>
    </xs:annotation>
    <xs:include schemaLocation="EEBus_SPINE_TS_CommonDataTypes.xsd"/>
    <xs:complexType name="SpecificationVersionDataType">
        <xs:simpleContent>
```

```
            <xs:extension base="ns_p:SpecificationVersionType"/>
        </xs:simpleContent>
    </xs:complexType>
    <xs:element name="specificationVersionData" type="ns_p:SpecificationVersionDataType"/>
    <xs:complexType name="SpecificationVersionDataElementsType"/>
    <xs:element name="specificationVersionDataElements"
type="ns_p:SpecificationVersionDataElementsType"/>
    <xs:complexType name="SpecificationVersionListDataType">
        <xs:sequence>
            <xs:element maxOccurs="unbounded" minOccurs="0"
ref="ns_p:specificationVersionData"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="specificationVersionListData"
type="ns_p:SpecificationVersionListDataType"/>
    <xs:complexType name="SpecificationVersionListDataSelectorsType"/>
    <xs:element name="specificationVersionListDataSelectors"
type="ns_p:SpecificationVersionListDataSelectorsType"/>
</xs:schema>
```

A.3 Further XSDs

A.3.1 CommandCommonDefinitions

File name: EEBus_SPINE_TS_CommandCommonDefinitions.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
    Smart Premises Interoperable Neutral-Message Exchange (SPINE)
    Version 1.1.1
    2018-12-21
    Copyright (c) 2018 EEBus Initiative e.V.  All Rights Reserved.
    Source: https://www.eebus.org/en/specifications/
-->
<xs:schema xmlns:ns_p="http://docs.eebus.org/spine/xsd/v1"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://docs.eebus.org/spine/xsd/v1" version="1.1.1"
    blockDefault="#all" elementFormDefault="qualified">
    <xs:annotation>
        <xs:documentation>EEBus SPINE Specification schema. Copyright 2018 EEBus Initiative
e.V.  All rights reserved.</xs:documentation>
    </xs:annotation>
    <xs:include schemaLocation="EEBus_SPINE_TS_ActuatorLevel.xsd"/>
    <xs:include schemaLocation="EEBus_SPINE_TS_ActuatorSwitch.xsd"/>
    <xs:include schemaLocation="EEBus_SPINE_TS_Alarm.xsd"/>
    <xs:include schemaLocation="EEBus_SPINE_TS_Bill.xsd"/>
    <xs:include schemaLocation="EEBus_SPINE_TS_BindingManagement.xsd"/>
    <xs:include schemaLocation="EEBus_SPINE_TS_DataTunneling.xsd"/>
    <xs:include schemaLocation="EEBus_SPINE_TS_DeviceClassification.xsd"/>
    <xs:include schemaLocation="EEBus_SPINE_TS_DeviceConfiguration.xsd"/>
    <xs:include schemaLocation="EEBus_SPINE_TS_DeviceDiagnosis.xsd"/>
    <xs:include schemaLocation="EEBus_SPINE_TS_DirectControl.xsd"/>
    <xs:include schemaLocation="EEBus_SPINE_TS_ElectricalConnection.xsd"/>
    <xs:include schemaLocation="EEBus_SPINE_TS_HVAC.xsd"/>
    <xs:include schemaLocation="EEBus_SPINE_TS_Identification.xsd"/>
    <xs:include schemaLocation="EEBus_SPINE_TS_IncentiveTable.xsd"/>
    <xs:include schemaLocation="EEBus_SPINE_TS_LoadControl.xsd"/>
    <xs:include schemaLocation="EEBus_SPINE_TS_Measurement.xsd"/>
    <xs:include schemaLocation="EEBus_SPINE_TS_Messaging.xsd"/>
    <xs:include schemaLocation="EEBus_SPINE_TS_NetworkManagement.xsd"/>
    <xs:include schemaLocation="EEBus_SPINE_TS_NodeManagement.xsd"/>
    <xs:include schemaLocation="EEBus_SPINE_TS_OperatingConstraints.xsd"/>
    <xs:include schemaLocation="EEBus_SPINE_TS_PowerSequences.xsd"/>
    <xs:include schemaLocation="EEBus_SPINE_TS_Result.xsd"/>
    <xs:include schemaLocation="EEBus_SPINE_TS_Sensing.xsd"/>
    <xs:include schemaLocation="EEBus_SPINE_TS_Setpoint.xsd"/>
    <xs:include schemaLocation="EEBus_SPINE_TS_SmartEnergyManagementPs.xsd"/>
    <xs:include schemaLocation="EEBus_SPINE_TS_SubscriptionManagement.xsd"/>
    <xs:include schemaLocation="EEBus_SPINE_TS_SupplyCondition.xsd"/>
    <xs:include schemaLocation="EEBus_SPINE_TS_TariffInformation.xsd"/>
    <xs:include schemaLocation="EEBus_SPINE_TS_TaskManagement.xsd"/>
```



```
    <xs:include schemaLocation="EEBus_SPINE_TS_Threshold.xsd"/>
    <xs:include schemaLocation="EEBus_SPINE_TS_TimeInformation.xsd"/>
    <xs:include schemaLocation="EEBus_SPINE_TS_TimeSeries.xsd"/>
    <xs:include schemaLocation="EEBus_SPINE_TS_TimeTable.xsd"/>
    <xs:include schemaLocation="EEBus_SPINE_TS_UseCaseInformation.xsd"/>
    <xs:include schemaLocation="EEBus_SPINE_TS_Version.xsd"/>
    <xs:group name="DataChoiceGroup">
        <xs:choice>
            <xs:element ref="ns_p:actuatorLevelData"/>
            <xs:element ref="ns_p:actuatorLevelDescriptionData"/>
            <xs:element ref="ns_p:actuatorSwitchData"/>
            <xs:element ref="ns_p:actuatorSwitchDescriptionData"/>
            <xs:element ref="ns_p:alarmListData"/>
            <xs:element ref="ns_p:billConstraintsListData"/>
            <xs:element ref="ns_p:billDescriptionListData"/>
            <xs:element ref="ns_p:billListData"/>
            <xs:element ref="ns_p:bindingManagementDeleteCall"/>
            <xs:element ref="ns_p:bindingManagementEntryListData"/>
            <xs:element ref="ns_p:bindingManagementRequestCall"/>
            <xs:element ref="ns_p:commodityListData"/>
            <xs:element ref="ns_p:dataTunnelingCall"/>
            <xs:element ref="ns_p:deviceClassificationManufacturerData"/>
            <xs:element ref="ns_p:deviceClassificationUserData"/>
            <xs:element ref="ns_p:deviceConfigurationKeyValueConstraintsListData"/>
            <xs:element ref="ns_p:deviceConfigurationKeyValueDescriptionListData"/>
            <xs:element ref="ns_p:deviceConfigurationKeyValueListData"/>
            <xs:element ref="ns_p:deviceDiagnosisHeartbeatData"/>
            <xs:element ref="ns_p:deviceDiagnosisServiceData"/>
            <xs:element ref="ns_p:deviceDiagnosisStateData"/>
            <xs:element ref="ns_p:directControlActivityListData"/>
            <xs:element ref="ns_p:directControlDescriptionData"/>
            <xs:element ref="ns_p:electricalConnectionDescriptionListData"/>
            <xs:element ref="ns_p:electricalConnectionParameterDescriptionListData"/>
            <xs:element ref="ns_p:electricalConnectionPermittedValueSetListData"/>
            <xs:element ref="ns_p:electricalConnectionStateListData"/>
            <xs:element ref="ns_p:hvacOperationModeDescriptionListData"/>
            <xs:element ref="ns_p:hvacOverrunDescriptionListData"/>
            <xs:element ref="ns_p:hvacOverrunListData"/>
            <xs:element ref="ns_p:hvacSystemFunctionDescriptionListData"/>
            <xs:element ref="ns_p:hvacSystemFunctionListData"/>
            <xs:element ref="ns_p:hvacSystemFunctionOperationModeRelationListData"/>
            <xs:element ref="ns_p:hvacSystemFunctionPowerSequenceRelationListData"/>
            <xs:element ref="ns_p:hvacSystemFunctionSetpointRelationListData"/>
            <xs:element ref="ns_p:identificationListData"/>
            <xs:element ref="ns_p:incentiveDescriptionListData"/>
            <xs:element ref="ns_p:incentiveListData"/>
            <xs:element ref="ns_p:incentiveTableConstraintsData"/>
            <xs:element ref="ns_p:incentiveTableData"/>
            <xs:element ref="ns_p:incentiveTableDescriptionData"/>
            <xs:element ref="ns_p:loadControlEventListData"/>
            <xs:element ref="ns_p:loadControlLimitConstraintsListData"/>
            <xs:element ref="ns_p:loadControlLimitDescriptionListData"/>
            <xs:element ref="ns_p:loadControlLimitListData"/>
            <xs:element ref="ns_p:loadControlNodeData"/>
            <xs:element ref="ns_p:loadControlStateListData"/>
            <xs:element ref="ns_p:measurementConstraintsListData"/>
            <xs:element ref="ns_p:measurementDescriptionListData"/>
            <xs:element ref="ns_p:measurementListData"/>
            <xs:element ref="ns_p:measurementThresholdRelationListData"/>
            <xs:element ref="ns_p:messagingListData"/>
            <xs:element ref="ns_p:networkManagementAbortCall"/>
            <xs:element ref="ns_p:networkManagementAddNodeCall"/>
            <xs:element ref="ns_p:networkManagementDeviceDescriptionListData"/>
            <xs:element ref="ns_p:networkManagementDiscoverCall"/>
            <xs:element ref="ns_p:networkManagementEntityDescriptionListData"/>
            <xs:element ref="ns_p:networkManagementFeatureDescriptionListData"/>
            <xs:element ref="ns_p:networkManagementJoiningModeData"/>
            <xs:element ref="ns_p:networkManagementModifyNodeCall"/>
            <xs:element ref="ns_p:networkManagementProcessStateData"/>
            <xs:element ref="ns_p:networkManagementRemoveNodeCall"/>
            <xs:element ref="ns_p:networkManagementReportCandidateData"/>
            <xs:element ref="ns_p:networkManagementScanNetworkCall"/>
            <xs:element ref="ns_p:nodeManagementBindingData"/>
            <xs:element ref="ns_p:nodeManagementBindingDeleteCall"/>
            <xs:element ref="ns_p:nodeManagementBindingRequestCall"/>
            <xs:element ref="ns_p:nodeManagementDestinationListData"/>
            <xs:element ref="ns_p:nodeManagementDetailedDiscoveryData"/>
            <xs:element ref="ns_p:nodeManagementSubscriptionData"/>
```



```
            <xs:element ref="ns_p:nodeManagementSubscriptionDeleteCall"/>
            <xs:element ref="ns_p:nodeManagementSubscriptionRequestCall"/>
            <xs:element ref="ns_p:nodeManagementUseCaseData"/>
            <xs:element ref="ns_p:operatingConstraintsDurationListData"/>
            <xs:element ref="ns_p:operatingConstraintsInterruptListData"/>
            <xs:element ref="ns_p:operatingConstraintsPowerDescriptionListData"/>
            <xs:element ref="ns_p:operatingConstraintsPowerLevelListData"/>
            <xs:element ref="ns_p:operatingConstraintsPowerRangeListData"/>
            <xs:element ref="ns_p:operatingConstraintsResumeImplicationListData"/>
            <xs:element ref="ns_p:powerSequenceAlternativesRelationListData"/>
            <xs:element ref="ns_p:powerSequenceDescriptionListData"/>
            <xs:element ref="ns_p:powerSequenceNodeScheduleInformationData"/>
            <xs:element ref="ns_p:powerSequencePriceCalculationRequestCall"/>
            <xs:element ref="ns_p:powerSequencePriceListData"/>
            <xs:element ref="ns_p:powerSequenceScheduleConfigurationRequestCall"/>
            <xs:element ref="ns_p:powerSequenceScheduleConstraintsListData"/>
            <xs:element ref="ns_p:powerSequenceScheduleListData"/>
            <xs:element ref="ns_p:powerSequenceSchedulePreferenceListData"/>
            <xs:element ref="ns_p:powerSequenceStateListData"/>
            <xs:element ref="ns_p:powerTimeSlotScheduleConstraintsListData"/>
            <xs:element ref="ns_p:powerTimeSlotScheduleListData"/>
            <xs:element ref="ns_p:powerTimeSlotValueListData"/>
            <xs:element ref="ns_p:resultData"/>
            <xs:element ref="ns_p:sensingDescriptionData"/>
            <xs:element ref="ns_p:sensingListData"/>
            <xs:element ref="ns_p:setpointConstraintsListData"/>
            <xs:element ref="ns_p:setpointDescriptionListData"/>
            <xs:element ref="ns_p:setpointListData"/>
            <xs:element ref="ns_p:smartEnergyManagementPsConfigurationRequestCall"/>
            <xs:element ref="ns_p:smartEnergyManagementPsData"/>
            <xs:element ref="ns_p:smartEnergyManagementPsPriceCalculationRequestCall"/>
            <xs:element ref="ns_p:smartEnergyManagementPsPriceData"/>
            <xs:element ref="ns_p:specificationVersionListData"/>
            <xs:element ref="ns_p:subscriptionManagementDeleteCall"/>
            <xs:element ref="ns_p:subscriptionManagementEntryListData"/>
            <xs:element ref="ns_p:subscriptionManagementRequestCall"/>
            <xs:element ref="ns_p:supplyConditionDescriptionListData"/>
            <xs:element ref="ns_p:supplyConditionListData"/>
            <xs:element ref="ns_p:supplyConditionThresholdRelationListData"/>
            <xs:element ref="ns_p:tariffBoundaryRelationListData"/>
            <xs:element ref="ns_p:tariffDescriptionListData"/>
            <xs:element ref="ns_p:tariffListData"/>
            <xs:element ref="ns_p:tariffOverallConstraintsData"/>
            <xs:element ref="ns_p:tariffTierRelationListData"/>
            <xs:element ref="ns_p:taskManagementJobDescriptionListData"/>
            <xs:element ref="ns_p:taskManagementJobListData"/>
            <xs:element ref="ns_p:taskManagementJobRelationListData"/>
            <xs:element ref="ns_p:taskManagementOverviewData"/>
            <xs:element ref="ns_p:thresholdConstraintsListData"/>
            <xs:element ref="ns_p:thresholdDescriptionListData"/>
            <xs:element ref="ns_p:thresholdListData"/>
            <xs:element ref="ns_p:tierBoundaryDescriptionListData"/>
            <xs:element ref="ns_p:tierBoundaryListData"/>
            <xs:element ref="ns_p:tierDescriptionListData"/>
            <xs:element ref="ns_p:tierIncentiveRelationListData"/>
            <xs:element ref="ns_p:tierListData"/>
            <xs:element ref="ns_p:timeDistributorData"/>
            <xs:element ref="ns_p:timeDistributorEnquiryCall"/>
            <xs:element ref="ns_p:timeInformationData"/>
            <xs:element ref="ns_p:timePrecisionData"/>
            <xs:element ref="ns_p:timeSeriesConstraintsListData"/>
            <xs:element ref="ns_p:timeSeriesDescriptionListData"/>
            <xs:element ref="ns_p:timeSeriesListData"/>
            <xs:element ref="ns_p:timeTableConstraintsListData"/>
            <xs:element ref="ns_p:timeTableDescriptionListData"/>
            <xs:element ref="ns_p:timeTableListData"/>
            <xs:element ref="ns_p:useCaseInformationListData"/>
        </xs:choice>
    </xs:group>
    <xs:group name="DataElementsChoiceGroup">
        <xs:choice>
            <xs:element ref="ns_p:actuatorLevelDataElements"/>
            <xs:element ref="ns_p:actuatorLevelDescriptionDataElements"/>
            <xs:element ref="ns_p:actuatorSwitchDataElements"/>
            <xs:element ref="ns_p:actuatorSwitchDescriptionDataElements"/>
            <xs:element ref="ns_p:alarmDataElements"/>
            <xs:element ref="ns_p:billConstraintsDataElements"/>
            <xs:element ref="ns_p:billDataElements"/>
```



```
            <xs:element ref="ns_p:billDescriptionDataElements"/>
            <xs:element ref="ns_p:bindingManagementDeleteCallElements"/>
            <xs:element ref="ns_p:bindingManagementEntryDataElements"/>
            <xs:element ref="ns_p:bindingManagementRequestCallElements"/>
            <xs:element ref="ns_p:commodityDataElements"/>
            <xs:element ref="ns_p:dataTunnelingCallElements"/>
            <xs:element ref="ns_p:deviceClassificationManufacturerDataElements"/>
            <xs:element ref="ns_p:deviceClassificationUserDataElements"/>
            <xs:element ref="ns_p:deviceConfigurationKeyValueConstraintsDataElements"/>
            <xs:element ref="ns_p:deviceConfigurationKeyValueDataElements"/>
            <xs:element ref="ns_p:deviceConfigurationKeyValueDescriptionDataElements"/>
            <xs:element ref="ns_p:deviceDiagnosisHeartbeatDataElements"/>
            <xs:element ref="ns_p:deviceDiagnosisServiceDataElements"/>
            <xs:element ref="ns_p:deviceDiagnosisStateDataElements"/>
            <xs:element ref="ns_p:directControlActivityDataElements"/>
            <xs:element ref="ns_p:directControlDescriptionDataElements"/>
            <xs:element ref="ns_p:electricalConnectionDescriptionDataElements"/>
            <xs:element ref="ns_p:electricalConnectionParameterDescriptionDataElements"/>
            <xs:element ref="ns_p:electricalConnectionPermittedValueSetDataElements"/>
            <xs:element ref="ns_p:electricalConnectionStateDataElements"/>
            <xs:element ref="ns_p:hvacOperationModeDescriptionDataElements"/>
            <xs:element ref="ns_p:hvacOverrunDataElements"/>
            <xs:element ref="ns_p:hvacOverrunDescriptionDataElements"/>
            <xs:element ref="ns_p:hvacSystemFunctionDataElements"/>
            <xs:element ref="ns_p:hvacSystemFunctionDescriptionDataElements"/>
            <xs:element ref="ns_p:hvacSystemFunctionOperationModeRelationDataElements"/>
            <xs:element ref="ns_p:hvacSystemFunctionPowerSequenceRelationDataElements"/>
            <xs:element ref="ns_p:hvacSystemFunctionSetpointRelationDataElements"/>
            <xs:element ref="ns_p:identificationDataElements"/>
            <xs:element ref="ns_p:incentiveDataElements"/>
            <xs:element ref="ns_p:incentiveDescriptionDataElements"/>
            <xs:element ref="ns_p:incentiveTableConstraintsDataElements"/>
            <xs:element ref="ns_p:incentiveTableDataElements"/>
            <xs:element ref="ns_p:incentiveTableDescriptionDataElements"/>
            <xs:element ref="ns_p:loadControlEventDataElements"/>
            <xs:element ref="ns_p:loadControlLimitConstraintsDataElements"/>
            <xs:element ref="ns_p:loadControlLimitDataElements"/>
            <xs:element ref="ns_p:loadControlLimitDescriptionDataElements"/>
            <xs:element ref="ns_p:loadControlNodeDataElements"/>
            <xs:element ref="ns_p:loadControlStateDataElements"/>
            <xs:element ref="ns_p:measurementConstraintsDataElements"/>
            <xs:element ref="ns_p:measurementDataElements"/>
            <xs:element ref="ns_p:measurementDescriptionDataElements"/>
            <xs:element ref="ns_p:measurementThresholdRelationDataElements"/>
            <xs:element ref="ns_p:messagingDataElements"/>
            <xs:element ref="ns_p:networkManagementAbortCallElements"/>
            <xs:element ref="ns_p:networkManagementAddNodeCallElements"/>
            <xs:element ref="ns_p:networkManagementDeviceDescriptionDataElements"/>
            <xs:element ref="ns_p:networkManagementDiscoverCallElements"/>
            <xs:element ref="ns_p:networkManagementEntityDescriptionDataElements"/>
            <xs:element ref="ns_p:networkManagementFeatureDescriptionDataElements"/>
            <xs:element ref="ns_p:networkManagementJoiningModeDataElements"/>
            <xs:element ref="ns_p:networkManagementModifyNodeCallElements"/>
            <xs:element ref="ns_p:networkManagementProcessStateDataElements"/>
            <xs:element ref="ns_p:networkManagementRemoveNodeCallElements"/>
            <xs:element ref="ns_p:networkManagementReportCandidateDataElements"/>
            <xs:element ref="ns_p:networkManagementScanNetworkCallElements"/>
            <xs:element ref="ns_p:nodeManagementBindingDataElements"/>
            <xs:element ref="ns_p:nodeManagementBindingDeleteCallElements"/>
            <xs:element ref="ns_p:nodeManagementBindingRequestCallElements"/>
            <xs:element ref="ns_p:nodeManagementDestinationDataElements"/>
            <xs:element ref="ns_p:nodeManagementDetailedDiscoveryDataElements"/>
            <xs:element ref="ns_p:nodeManagementSubscriptionDataElements"/>
            <xs:element ref="ns_p:nodeManagementSubscriptionDeleteCallElements"/>
            <xs:element ref="ns_p:nodeManagementSubscriptionRequestCallElements"/>
            <xs:element ref="ns_p:nodeManagementUseCaseDataElements"/>
            <xs:element ref="ns_p:operatingConstraintsDurationDataElements"/>
            <xs:element ref="ns_p:operatingConstraintsInterruptDataElements"/>
            <xs:element ref="ns_p:operatingConstraintsPowerDescriptionDataElements"/>
            <xs:element ref="ns_p:operatingConstraintsPowerLevelDataElements"/>
            <xs:element ref="ns_p:operatingConstraintsPowerRangeDataElements"/>
            <xs:element ref="ns_p:operatingConstraintsResumeImplicationDataElements"/>
            <xs:element ref="ns_p:powerSequenceAlternativesRelationDataElements"/>
            <xs:element ref="ns_p:powerSequenceDescriptionDataElements"/>
            <xs:element ref="ns_p:powerSequenceNodeScheduleInformationDataElements"/>
            <xs:element ref="ns_p:powerSequencePriceCalculationRequestCallElements"/>
            <xs:element ref="ns_p:powerSequencePriceDataElements"/>
            <xs:element ref="ns_p:powerSequenceScheduleConfigurationRequestCallElements"/>
```



```
            <xs:element ref="ns_p:powerSequenceScheduleConstraintsDataElements"/>
            <xs:element ref="ns_p:powerSequenceScheduleDataElements"/>
            <xs:element ref="ns_p:powerSequenceSchedulePreferenceDataElements"/>
            <xs:element ref="ns_p:powerSequenceStateDataElements"/>
            <xs:element ref="ns_p:powerTimeSlotScheduleConstraintsDataElements"/>
            <xs:element ref="ns_p:powerTimeSlotScheduleDataElements"/>
            <xs:element ref="ns_p:powerTimeSlotValueDataElements"/>
            <xs:element ref="ns_p:sensingDataElements"/>
            <xs:element ref="ns_p:sensingDescriptionDataElements"/>
            <xs:element ref="ns_p:setpointConstraintsDataElements"/>
            <xs:element ref="ns_p:setpointDataElements"/>
            <xs:element ref="ns_p:setpointDescriptionDataElements"/>
            <xs:element ref="ns_p:smartEnergyManagementPsConfigurationRequestCallElements"/>
            <xs:element ref="ns_p:smartEnergyManagementPsDataElements"/>
            <xs:element
ref="ns_p:smartEnergyManagementPsPriceCalculationRequestCallElements"/>
            <xs:element ref="ns_p:smartEnergyManagementPsPriceDataElements"/>
            <xs:element ref="ns_p:specificationVersionDataElements"/>
            <xs:element ref="ns_p:subscriptionManagementDeleteCallElements"/>
            <xs:element ref="ns_p:subscriptionManagementEntryDataElements"/>
            <xs:element ref="ns_p:subscriptionManagementRequestCallElements"/>
            <xs:element ref="ns_p:supplyConditionDataElements"/>
            <xs:element ref="ns_p:supplyConditionDescriptionDataElements"/>
            <xs:element ref="ns_p:supplyConditionThresholdRelationDataElements"/>
            <xs:element ref="ns_p:tariffBoundaryRelationDataElements"/>
            <xs:element ref="ns_p:tariffDataElements"/>
            <xs:element ref="ns_p:tariffDescriptionDataElements"/>
            <xs:element ref="ns_p:tariffOverallConstraintsDataElements"/>
            <xs:element ref="ns_p:tariffTierRelationDataElements"/>
            <xs:element ref="ns_p:taskManagementJobDataElements"/>
            <xs:element ref="ns_p:taskManagementJobDescriptionDataElements"/>
            <xs:element ref="ns_p:taskManagementJobRelationDataElements"/>
            <xs:element ref="ns_p:taskManagementOverviewDataElements"/>
            <xs:element ref="ns_p:thresholdConstraintsDataElements"/>
            <xs:element ref="ns_p:thresholdDataElements"/>
            <xs:element ref="ns_p:thresholdDescriptionDataElements"/>
            <xs:element ref="ns_p:tierBoundaryDataElements"/>
            <xs:element ref="ns_p:tierBoundaryDescriptionDataElements"/>
            <xs:element ref="ns_p:tierDataElements"/>
            <xs:element ref="ns_p:tierDescriptionDataElements"/>
            <xs:element ref="ns_p:tierIncentiveRelationDataElements"/>
            <xs:element ref="ns_p:timeDistributorDataElements"/>
            <xs:element ref="ns_p:timeDistributorEnquiryCallElements"/>
            <xs:element ref="ns_p:timeInformationDataElements"/>
            <xs:element ref="ns_p:timePrecisionDataElements"/>
            <xs:element ref="ns_p:timeSeriesConstraintsDataElements"/>
            <xs:element ref="ns_p:timeSeriesDataElements"/>
            <xs:element ref="ns_p:timeSeriesDescriptionDataElements"/>
            <xs:element ref="ns_p:timeTableConstraintsDataElements"/>
            <xs:element ref="ns_p:timeTableDataElements"/>
            <xs:element ref="ns_p:timeTableDescriptionDataElements"/>
            <xs:element ref="ns_p:useCaseInformationDataElements"/>
        </xs:choice>
    </xs:group>
    <xs:group name="DataSelectorsChoiceGroup">
        <xs:choice>
            <xs:element ref="ns_p:alarmListDataSelectors"/>
            <xs:element ref="ns_p:billConstraintsListDataSelectors"/>
            <xs:element ref="ns_p:billDescriptionListDataSelectors"/>
            <xs:element ref="ns_p:billListDataSelectors"/>
            <xs:element ref="ns_p:bindingManagementEntryListDataSelectors"/>
            <xs:element ref="ns_p:commodityListDataSelectors"/>
            <xs:element ref="ns_p:deviceConfigurationKeyValueConstraintsListDataSelectors"/>
            <xs:element ref="ns_p:deviceConfigurationKeyValueDescriptionListDataSelectors"/>
            <xs:element ref="ns_p:deviceConfigurationKeyValueListDataSelectors"/>
            <xs:element ref="ns_p:directControlActivityListDataSelectors"/>
            <xs:element ref="ns_p:electricalConnectionDescriptionListDataSelectors"/>
            <xs:element ref="ns_p:electricalConnectionParameterDescriptionListDataSelectors"/>
            <xs:element ref="ns_p:electricalConnectionPermittedValueSetListDataSelectors"/>
            <xs:element ref="ns_p:electricalConnectionStateListDataSelectors"/>
            <xs:element ref="ns_p:hvacOperationModeDescriptionListDataSelectors"/>
            <xs:element ref="ns_p:hvacOverrunDescriptionListDataSelectors"/>
            <xs:element ref="ns_p:hvacOverrunListDataSelectors"/>
            <xs:element ref="ns_p:hvacSystemFunctionDescriptionListDataSelectors"/>
            <xs:element ref="ns_p:hvacSystemFunctionListDataSelectors"/>
            <xs:element ref="ns_p:hvacSystemFunctionOperationModeRelationListDataSelectors"/>
            <xs:element ref="ns_p:hvacSystemFunctionPowerSequenceRelationListDataSelectors"/>
            <xs:element ref="ns_p:hvacSystemFunctionSetpointRelationListDataSelectors"/>
```



```
            <xs:element ref="ns_p:identificationListDataSelectors"/>
            <xs:element ref="ns_p:incentiveDescriptionListDataSelectors"/>
            <xs:element ref="ns_p:incentiveListDataSelectors"/>
            <xs:element ref="ns_p:incentiveTableConstraintsDataSelectors"/>
            <xs:element ref="ns_p:incentiveTableDataSelectors"/>
            <xs:element ref="ns_p:incentiveTableDescriptionDataSelectors"/>
            <xs:element ref="ns_p:loadControlEventListDataSelectors"/>
            <xs:element ref="ns_p:loadControlLimitConstraintsListDataSelectors"/>
            <xs:element ref="ns_p:loadControlLimitDescriptionListDataSelectors"/>
            <xs:element ref="ns_p:loadControlLimitListDataSelectors"/>
            <xs:element ref="ns_p:loadControlStateListDataSelectors"/>
            <xs:element ref="ns_p:measurementConstraintsListDataSelectors"/>
            <xs:element ref="ns_p:measurementDescriptionListDataSelectors"/>
            <xs:element ref="ns_p:measurementListDataSelectors"/>
            <xs:element ref="ns_p:measurementThresholdRelationListDataSelectors"/>
            <xs:element ref="ns_p:messagingListDataSelectors"/>
            <xs:element ref="ns_p:networkManagementDeviceDescriptionListDataSelectors"/>
            <xs:element ref="ns_p:networkManagementEntityDescriptionListDataSelectors"/>
            <xs:element ref="ns_p:networkManagementFeatureDescriptionListDataSelectors"/>
            <xs:element ref="ns_p:nodeManagementBindingDataSelectors"/>
            <xs:element ref="ns_p:nodeManagementDestinationListDataSelectors"/>
            <xs:element ref="ns_p:nodeManagementDetailedDiscoveryDataSelectors"/>
            <xs:element ref="ns_p:nodeManagementSubscriptionDataSelectors"/>
            <xs:element ref="ns_p:nodeManagementUseCaseDataSelectors"/>
            <xs:element ref="ns_p:operatingConstraintsDurationListDataSelectors"/>
            <xs:element ref="ns_p:operatingConstraintsInterruptListDataSelectors"/>
            <xs:element ref="ns_p:operatingConstraintsPowerDescriptionListDataSelectors"/>
            <xs:element ref="ns_p:operatingConstraintsPowerLevelListDataSelectors"/>
            <xs:element ref="ns_p:operatingConstraintsPowerRangeListDataSelectors"/>
            <xs:element ref="ns_p:operatingConstraintsResumeImplicationListDataSelectors"/>
            <xs:element ref="ns_p:powerSequenceAlternativesRelationListDataSelectors"/>
            <xs:element ref="ns_p:powerSequenceDescriptionListDataSelectors"/>
            <xs:element ref="ns_p:powerSequencePriceListDataSelectors"/>
            <xs:element ref="ns_p:powerSequenceScheduleConstraintsListDataSelectors"/>
            <xs:element ref="ns_p:powerSequenceScheduleListDataSelectors"/>
            <xs:element ref="ns_p:powerSequenceSchedulePreferenceListDataSelectors"/>
            <xs:element ref="ns_p:powerSequenceStateListDataSelectors"/>
            <xs:element ref="ns_p:powerTimeSlotScheduleConstraintsListDataSelectors"/>
            <xs:element ref="ns_p:powerTimeSlotScheduleListDataSelectors"/>
            <xs:element ref="ns_p:powerTimeSlotValueListDataSelectors"/>
            <xs:element ref="ns_p:sensingListDataSelectors"/>
            <xs:element ref="ns_p:setpointConstraintsListDataSelectors"/>
            <xs:element ref="ns_p:setpointDescriptionListDataSelectors"/>
            <xs:element ref="ns_p:setpointListDataSelectors"/>
            <xs:element ref="ns_p:smartEnergyManagementPsDataSelectors"/>
            <xs:element ref="ns_p:smartEnergyManagementPsPriceDataSelectors"/>
            <xs:element ref="ns_p:specificationVersionListDataSelectors"/>
            <xs:element ref="ns_p:subscriptionManagementEntryListDataSelectors"/>
            <xs:element ref="ns_p:supplyConditionDescriptionListDataSelectors"/>
            <xs:element ref="ns_p:supplyConditionListDataSelectors"/>
            <xs:element ref="ns_p:supplyConditionThresholdRelationListDataSelectors"/>
            <xs:element ref="ns_p:tariffBoundaryRelationListDataSelectors"/>
            <xs:element ref="ns_p:tariffDescriptionListDataSelectors"/>
            <xs:element ref="ns_p:tariffListDataSelectors"/>
            <xs:element ref="ns_p:tariffTierRelationListDataSelectors"/>
            <xs:element ref="ns_p:taskManagementJobDescriptionListDataSelectors"/>
            <xs:element ref="ns_p:taskManagementJobListDataSelectors"/>
            <xs:element ref="ns_p:taskManagementJobRelationListDataSelectors"/>
            <xs:element ref="ns_p:thresholdConstraintsListDataSelectors"/>
            <xs:element ref="ns_p:thresholdDescriptionListDataSelectors"/>
            <xs:element ref="ns_p:thresholdListDataSelectors"/>
            <xs:element ref="ns_p:tierBoundaryDescriptionListDataSelectors"/>
            <xs:element ref="ns_p:tierBoundaryListDataSelectors"/>
            <xs:element ref="ns_p:tierDescriptionListDataSelectors"/>
            <xs:element ref="ns_p:tierIncentiveRelationListDataSelectors"/>
            <xs:element ref="ns_p:tierListDataSelectors"/>
            <xs:element ref="ns_p:timeSeriesConstraintsListDataSelectors"/>
            <xs:element ref="ns_p:timeSeriesDescriptionListDataSelectors"/>
            <xs:element ref="ns_p:timeSeriesListDataSelectors"/>
            <xs:element ref="ns_p:timeTableConstraintsListDataSelectors"/>
            <xs:element ref="ns_p:timeTableDescriptionListDataSelectors"/>
            <xs:element ref="ns_p:timeTableListDataSelectors"/>
            <xs:element ref="ns_p:useCaseInformationListDataSelectors"/>
        </xs:choice>
    </xs:group>
</xs:schema>
```


A.3.2 CommandFrame

File name: EEBus_SPINE_TS_CommandFrame.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
    Smart Premises Interoperable Neutral-Message Exchange (SPINE)
    Version 1.1.1
    2018-12-21
    Copyright (c) 2018 EEBus Initiative e.V.  All Rights Reserved.
    Source: https://www.eebus.org/en/specifications/
-->
<xs:schema xmlns:ns_p="http://docs.eebus.org/spine/xsd/v1"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://docs.eebus.org/spine/xsd/v1" version="1.1.1"
    blockDefault="#all" elementFormDefault="qualified">
    <xs:annotation>
        <xs:documentation>EEBus SPINE Specification schema. Copyright 2018 EEBus Initiative
e.V.  All rights reserved.</xs:documentation>
    </xs:annotation>
    <xs:include schemaLocation="EEBus_SPINE_TS_CommandCommonDefinitions.xsd"/>
    <xs:include schemaLocation="EEBus_SPINE_TS_CommonDataTypes.xsd"/>
    <xs:simpleType name="MsgCounterType">
        <xs:restriction base="xs:unsignedLong"> </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="CmdClassifierType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="read"/>
            <xs:enumeration value="reply"/>
            <xs:enumeration value="notify"/>
            <xs:enumeration value="write"/>
            <xs:enumeration value="call"/>
            <xs:enumeration value="result"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:element name="cmdClassifier" type="ns_p:CmdClassifierType"/>
    <xs:element name="manufacturerSpecificExtension" type="xs:hexBinary"/>
    <xs:element name="lastUpdateAt" type="ns_p:AbsoluteOrRelativeTimeType"/>
    <xs:element name="function" type="ns_p:FunctionType"/>
    <xs:simpleType name="FilterIdType">
        <xs:restriction base="xs:unsignedInt"/>
    </xs:simpleType>
    <xs:complexType name="FilterType">
        <xs:sequence>
            <xs:element minOccurs="0" name="filterId" type="ns_p:FilterIdType"/>
            <xs:element minOccurs="0" ref="ns_p:cmdControl"/>
            <xs:group ref="ns_p:DataSelectorsChoiceGroup" minOccurs="0"
maxOccurs="unbounded"/>
            <xs:group ref="ns_p:DataElementsChoiceGroup" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="filter" type="ns_p:FilterType"/>
    <xs:complexType name="CmdControlType">
        <xs:sequence>
            <xs:element name="delete" minOccurs="0" type="ns_p:ElementTagType"/>
            <xs:element name="partial" minOccurs="0" type="ns_p:ElementTagType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="cmdControl" type="ns_p:CmdControlType"/>
    <xs:group name="CmdOptionGroup">
        <xs:sequence>
            <xs:element minOccurs="0" ref="ns_p:function"/>
            <xs:element minOccurs="0" ref="ns_p:filter" maxOccurs="unbounded"/>
        </xs:sequence>
    </xs:group>
    <xs:group name="DataExtendGroup">
        <xs:sequence>
            <xs:element minOccurs="0" ref="ns_p:manufacturerSpecificExtension"/>
            <xs:element minOccurs="0" ref="ns_p:lastUpdateAt"/>
        </xs:sequence>
    </xs:group>
    <xs:group name="PayloadContributionGroup">
        <xs:sequence>
            <xs:group ref="ns_p:CmdOptionGroup" minOccurs="0"/>
```



```
            <xs:group ref="ns_p:DataChoiceGroup" minOccurs="0"/>
            <xs:group minOccurs="0" ref="ns_p:DataExtendGroup"/>
        </xs:sequence>
    </xs:group>
    <xs:complexType name="CmdType">
        <xs:sequence>
            <xs:group ref="ns_p:PayloadContributionGroup"/>
        </xs:sequence>
    </xs:complexType>
</xs:schema>
```

A.3.3 CommonDataTypes

File name: EEBus_SPINE_TS_CommonDataTypes.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
    Smart Premises Interoperable Neutral-Message Exchange (SPINE)
    Version 1.1.1
    2018-12-21
    Copyright (c) 2018 EEBus Initiative e.V.  All Rights Reserved.
    Source: https://www.eebus.org/en/specifications/
-->
<xs:schema xmlns:ns_p="http://docs.eebus.org/spine/xsd/v1"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://docs.eebus.org/spine/xsd/v1" version="1.1.1"
    blockDefault="#all" elementFormDefault="qualified">
    <xs:annotation>
        <xs:documentation>EEBus SPINE Specification schema. Copyright 2018 EEBus Initiative
e.V.  All rights reserved.</xs:documentation>
    </xs:annotation>
    <xs:complexType name="ElementTagType"/>
    <xs:simpleType name="LabelType">
        <xs:restriction base="xs:string"> </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="DescriptionType">
        <xs:restriction base="xs:string"> </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="SpecificationVersionType">
        <xs:restriction base="xs:string">
            <xs:pattern value="[1-9][0-9]*\.[0-9]+\.[0-9]+"/>
            <xs:maxLength value="128"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:complexType name="TimePeriodType">
        <xs:sequence>
            <xs:element minOccurs="0" name="startTime"
type="ns_p:AbsoluteOrRelativeTimeType"/>
            <xs:element minOccurs="0" name="endTime" type="ns_p:AbsoluteOrRelativeTimeType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="TimePeriodElementsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="startTime" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="endTime" type="ns_p:ElementTagType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="TimestampIntervalType">
        <xs:sequence>
            <xs:element name="startTime" type="ns_p:AbsoluteOrRelativeTimeType"
minOccurs="0"/>
            <xs:element name="endTime" type="ns_p:AbsoluteOrRelativeTimeType" minOccurs="0"/>
        </xs:sequence>
    </xs:complexType>
    <xs:simpleType name="EnumExtendType">
        <xs:restriction base="xs:string">
            <xs:pattern value="_(i:[1-9][0-9]*|n:[a-zA-Z0-9-]+)_[^\p{Cc}\p{Cf}\p{Z}]+"/>
            <xs:maxLength value="256"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="AbsoluteOrRelativeTimeType">
        <xs:union memberTypes="xs:duration xs:dateTime"/>
```

```
    </xs:simpleType>
    <xs:simpleType name="RecurringIntervalType">
        <xs:union memberTypes="ns_p:RecurringIntervalEnumType ns_p:EnumExtendType"/>
    </xs:simpleType>
    <xs:simpleType name="RecurringIntervalEnumType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="yearly"/>
            <xs:enumeration value="monthly"/>
            <xs:enumeration value="weekly"/>
            <xs:enumeration value="daily"/>
            <xs:enumeration value="hourly"/>
            <xs:enumeration value="everyMinute"/>
            <xs:enumeration value="everySecond"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="MonthType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="january"/>
            <xs:enumeration value="february"/>
            <xs:enumeration value="march"/>
            <xs:enumeration value="april"/>
            <xs:enumeration value="may"/>
            <xs:enumeration value="june"/>
            <xs:enumeration value="july"/>
            <xs:enumeration value="august"/>
            <xs:enumeration value="september"/>
            <xs:enumeration value="october"/>
            <xs:enumeration value="november"/>
            <xs:enumeration value="december"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="DayOfMonthType">
        <xs:restriction base="xs:unsignedByte">
            <xs:minInclusive value="1"/>
            <xs:maxInclusive value="31"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="CalendarWeekType">
        <xs:restriction base="xs:unsignedByte">
            <xs:minInclusive value="1"/>
            <xs:maxInclusive value="53"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="DayOfWeekType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="monday"/>
            <xs:enumeration value="tuesday"/>
            <xs:enumeration value="wednesday"/>
            <xs:enumeration value="thursday"/>
            <xs:enumeration value="friday"/>
            <xs:enumeration value="saturday"/>
            <xs:enumeration value="sunday"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:complexType name="DaysOfWeekType">
        <xs:sequence>
            <xs:element name="monday" minOccurs="0" type="ns_p:ElementTagType"/>
            <xs:element name="tuesday" minOccurs="0" type="ns_p:ElementTagType"/>
            <xs:element name="wednesday" minOccurs="0" type="ns_p:ElementTagType"/>
            <xs:element name="thursday" minOccurs="0" type="ns_p:ElementTagType"/>
            <xs:element name="friday" minOccurs="0" type="ns_p:ElementTagType"/>
            <xs:element name="saturday" minOccurs="0" type="ns_p:ElementTagType"/>
            <xs:element name="sunday" minOccurs="0" type="ns_p:ElementTagType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:simpleType name="OccurrenceType">
        <xs:union memberTypes="ns_p:OccurrenceEnumType ns_p:EnumExtendType"/>
    </xs:simpleType>
    <xs:simpleType name="OccurrenceEnumType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="first"/>
            <xs:enumeration value="second"/>
            <xs:enumeration value="third"/>
            <xs:enumeration value="fourth"/>
            <xs:enumeration value="last"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:complexType name="AbsoluteOrRecurringTimeType">
```



```
20320           <xs:sequence>
20321               <xs:element minOccurs="0" name="dateTime" type="xs:dateTime"/>
20322               <xs:element name="month" minOccurs="0" type="ns_p:MonthType"/>
20323               <xs:element name="dayOfMonth" minOccurs="0" type="ns_p:DayOfMonthType"/>
20324               <xs:element minOccurs="0" name="calendarWeek" type="ns_p:CalendarWeekType"/>
20325               <xs:element minOccurs="0" name="dayOfWeekOccurrence" type="ns_p:OccurrenceType"/>
20326               <xs:element minOccurs="0" name="daysOfWeek" type="ns_p:DaysOfWeekType"/>
20327               <xs:element minOccurs="0" name="time" type="xs:time"/>
20328               <xs:element minOccurs="0" name="relative" type="xs:duration"/>
20329           </xs:sequence>
20330       </xs:complexType>
20331       <xs:complexType name="AbsoluteOrRecurringTimeElementsType">
20332           <xs:sequence>
20333               <xs:element minOccurs="0" name="dateTime" type="ns_p:ElementTagType"/>
20334               <xs:element name="month" minOccurs="0" type="ns_p:ElementTagType"/>
20335               <xs:element name="dayOfMonth" minOccurs="0" type="ns_p:ElementTagType"/>
20336               <xs:element minOccurs="0" name="calendarWeek" type="ns_p:ElementTagType"/>
20337               <xs:element minOccurs="0" name="dayOfWeekOccurrence" type="ns_p:ElementTagType"/>
20338               <xs:element minOccurs="0" name="daysOfWeek" type="ns_p:ElementTagType"/>
20339               <xs:element minOccurs="0" name="time" type="ns_p:ElementTagType"/>
20340               <xs:element minOccurs="0" name="relative" type="ns_p:ElementTagType"/>
20341           </xs:sequence>
20342       </xs:complexType>
20343       <xs:complexType name="RecurrenceInformationType">
20344           <xs:sequence>
20345               <xs:element minOccurs="0" name="recurringInterval"
20346   type="ns_p:RecurringIntervalType"/>
20347               <xs:element minOccurs="0" name="recurringIntervalStep" type="xs:unsignedInt"/>
20348               <xs:element minOccurs="0" name="firstExecution" type="xs:dateTime"/>
20349               <xs:element minOccurs="0" name="executionCount" type="xs:unsignedInt"/>
20350               <xs:element minOccurs="0" name="lastExecution" type="xs:dateTime"/>
20351           </xs:sequence>
20352       </xs:complexType>
20353       <xs:complexType name="RecurrenceInformationElementsType">
20354           <xs:sequence>
20355               <xs:element minOccurs="0" name="recurringInterval" type="ns_p:ElementTagType"/>
20356               <xs:element minOccurs="0" name="recurringIntervalStep"
20357   type="ns_p:ElementTagType"/>
20358               <xs:element minOccurs="0" name="firstExecution" type="ns_p:ElementTagType"/>
20359               <xs:element minOccurs="0" name="executionCount" type="ns_p:ElementTagType"/>
20360               <xs:element minOccurs="0" name="lastExecution" type="ns_p:ElementTagType"/>
20361           </xs:sequence>
20362       </xs:complexType>
20363       <xs:complexType name="ScaledNumberRangeType">
20364           <xs:sequence>
20365               <xs:element minOccurs="0" name="min" type="ns_p:ScaledNumberType"/>
20366               <xs:element minOccurs="0" name="max" type="ns_p:ScaledNumberType"/>
20367           </xs:sequence>
20368       </xs:complexType>
20369       <xs:complexType name="ScaledNumberRangeElementsType">
20370           <xs:sequence>
20371               <xs:element minOccurs="0" name="min" type="ns_p:ScaledNumberElementsType"/>
20372               <xs:element minOccurs="0" name="max" type="ns_p:ScaledNumberElementsType"/>
20373           </xs:sequence>
20374       </xs:complexType>
20375       <xs:complexType name="ScaledNumberSetType">
20376           <xs:sequence>
20377               <xs:element maxOccurs="unbounded" minOccurs="0" name="value"
20378   type="ns_p:ScaledNumberType"/>
20379               <xs:element maxOccurs="unbounded" minOccurs="0" name="range"
20380   type="ns_p:ScaledNumberRangeType"/>
20381           </xs:sequence>
20382       </xs:complexType>
20383       <xs:complexType name="ScaledNumberSetElementsType">
20384           <xs:sequence>
20385               <xs:element minOccurs="0" name="value" type="ns_p:ScaledNumberElementsType"/>
20386               <xs:element minOccurs="0" name="range" type="ns_p:ScaledNumberRangeElementsType"/>
20387           </xs:sequence>
20388       </xs:complexType>
20389       <xs:simpleType name="NumberType">
20390           <xs:restriction base="xs:long"/>
20391       </xs:simpleType>
20392       <xs:simpleType name="ScaleType">
20393           <xs:restriction base="xs:short"/>
20394       </xs:simpleType>
20395       <xs:complexType name="ScaledNumberType">
20396           <xs:sequence>
20397               <xs:element minOccurs="0" name="number" type="ns_p:NumberType"/>
```



```
            <xs:element minOccurs="0" name="scale" type="ns_p:ScaleType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="ScaledNumberElementsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="number" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="scale" type="ns_p:ElementTagType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:simpleType name="MaxResponseDelayType">
        <xs:restriction base="xs:duration"/>
    </xs:simpleType>
    <xs:simpleType name="CommodityTypeType">
        <xs:union memberTypes="ns_p:CommodityTypeEnumType ns_p:EnumExtendType"/>
    </xs:simpleType>
    <xs:simpleType name="CommodityTypeEnumType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="electricity"/>
            <xs:enumeration value="gas"/>
            <xs:enumeration value="oil"/>
            <xs:enumeration value="water"/>
            <xs:enumeration value="wasteWater"/>
            <xs:enumeration value="domesticHotWater"/>
            <xs:enumeration value="heatingWater"/>
            <xs:enumeration value="steam"/>
            <xs:enumeration value="heat"/>
            <xs:enumeration value="coolingLoad"/>
            <xs:enumeration value="air"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="EnergyDirectionType">
        <xs:union memberTypes="ns_p:EnergyDirectionEnumType ns_p:EnumExtendType"/>
    </xs:simpleType>
    <xs:simpleType name="EnergyDirectionEnumType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="consume"/>
            <xs:enumeration value="produce"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="EnergyModeType">
        <xs:union memberTypes="ns_p:EnergyModeEnumType ns_p:EnumExtendType"/>
    </xs:simpleType>
    <xs:simpleType name="EnergyModeEnumType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="consume"/>
            <xs:enumeration value="produce"/>
            <xs:enumeration value="idle"/>
            <xs:enumeration value="auto"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="UnitOfMeasurementType">
        <xs:union memberTypes="ns_p:UnitOfMeasurementEnumType ns_p:EnumExtendType"/>
    </xs:simpleType>
    <xs:simpleType name="UnitOfMeasurementEnumType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="unknown"/>
            <xs:enumeration value="1"/>
            <xs:enumeration value="m"/>
            <xs:enumeration value="kg"/>
            <xs:enumeration value="s"/>
            <xs:enumeration value="A"/>
            <xs:enumeration value="K"/>
            <xs:enumeration value="mol"/>
            <xs:enumeration value="cd"/>
            <xs:enumeration value="V"/>
            <xs:enumeration value="W"/>
            <xs:enumeration value="Wh"/>
            <xs:enumeration value="VA"/>
            <xs:enumeration value="VAh"/>
            <xs:enumeration value="var"/>
            <xs:enumeration value="varh"/>
            <xs:enumeration value="degC"/>
            <xs:enumeration value="degF"/>
            <xs:enumeration value="Lm"/>
            <xs:enumeration value="lx"/>
            <xs:enumeration value="Ohm"/>
            <xs:enumeration value="Hz"/>
            <xs:enumeration value="dB"/>
```



```
            <xs:enumeration value="dBm"/>
            <xs:enumeration value="pct"/>
            <xs:enumeration value="ppm"/>
            <xs:enumeration value="l"/>
            <xs:enumeration value="l/s"/>
            <xs:enumeration value="l/h"/>
            <xs:enumeration value="deg"/>
            <xs:enumeration value="rad"/>
            <xs:enumeration value="rad/s"/>
            <xs:enumeration value="sr"/>
            <xs:enumeration value="Gy"/>
            <xs:enumeration value="Bq"/>
            <xs:enumeration value="Bq/m^3"/>
            <xs:enumeration value="Sv"/>
            <xs:enumeration value="Rd"/>
            <xs:enumeration value="C"/>
            <xs:enumeration value="F"/>
            <xs:enumeration value="H"/>
            <xs:enumeration value="J"/>
            <xs:enumeration value="N"/>
            <xs:enumeration value="N_m"/>
            <xs:enumeration value="N_s"/>
            <xs:enumeration value="Wb"/>
            <xs:enumeration value="T"/>
            <xs:enumeration value="Pa"/>
            <xs:enumeration value="bar"/>
            <xs:enumeration value="atm"/>
            <xs:enumeration value="psi"/>
            <xs:enumeration value="mmHg"/>
            <xs:enumeration value="m^2"/>
            <xs:enumeration value="m^3"/>
            <xs:enumeration value="m^3/h"/>
            <xs:enumeration value="m/s"/>
            <xs:enumeration value="m/s^2"/>
            <xs:enumeration value="m^3/s"/>
            <xs:enumeration value="m/m^3"/>
            <xs:enumeration value="kg/m^3"/>
            <xs:enumeration value="kg_m"/>
            <xs:enumeration value="m^2/s"/>
            <xs:enumeration value="W/m_K"/>
            <xs:enumeration value="J/K"/>
            <xs:enumeration value="1/s"/>
            <xs:enumeration value="W/m^2"/>
            <xs:enumeration value="J/m^2"/>
            <xs:enumeration value="S"/>
            <xs:enumeration value="S/m"/>
            <xs:enumeration value="K/s"/>
            <xs:enumeration value="Pa/s"/>
            <xs:enumeration value="J/kg_K"/>
            <xs:enumeration value="Vs"/>
            <xs:enumeration value="V/m"/>
            <xs:enumeration value="V/Hz"/>
            <xs:enumeration value="As"/>
            <xs:enumeration value="A/m"/>
            <xs:enumeration value="Hz/s"/>
            <xs:enumeration value="kg/s"/>
            <xs:enumeration value="kg_m^2"/>
            <xs:enumeration value="J/Wh"/>
            <xs:enumeration value="W/s"/>
            <xs:enumeration value="ft^3"/>
            <xs:enumeration value="ft^3/h"/>
            <xs:enumeration value="ccf"/>
            <xs:enumeration value="ccf/h"/>
            <xs:enumeration value="US.liq.gal"/>
            <xs:enumeration value="US.liq.gal/h"/>
            <xs:enumeration value="Imp.gal"/>
            <xs:enumeration value="Imp.gal/h"/>
            <xs:enumeration value="Btu"/>
            <xs:enumeration value="Btu/h"/>
            <xs:enumeration value="Ah"/>
            <xs:enumeration value="kg/Wh"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="CurrencyType">
        <xs:union memberTypes="ns_p:CurrencyEnumType ns_p:EnumExtendType"/>
    </xs:simpleType>
    <xs:simpleType name="CurrencyEnumType">
        <xs:restriction base="xs:string">
```



```
20554               <xs:enumeration value="AED"/>
20555               <xs:enumeration value="AFN"/>
20556               <xs:enumeration value="ALL"/>
20557               <xs:enumeration value="AMD"/>
20558               <xs:enumeration value="ANG"/>
20559               <xs:enumeration value="AOA"/>
20560               <xs:enumeration value="ARS"/>
20561               <xs:enumeration value="AUD"/>
20562               <xs:enumeration value="AWG"/>
20563               <xs:enumeration value="AZN"/>
20564               <xs:enumeration value="BAM"/>
20565               <xs:enumeration value="BBD"/>
20566               <xs:enumeration value="BDT"/>
20567               <xs:enumeration value="BGN"/>
20568               <xs:enumeration value="BHD"/>
20569               <xs:enumeration value="BIF"/>
20570               <xs:enumeration value="BMD"/>
20571               <xs:enumeration value="BND"/>
20572               <xs:enumeration value="BOB"/>
20573               <xs:enumeration value="BOV"/>
20574               <xs:enumeration value="BRL"/>
20575               <xs:enumeration value="BSD"/>
20576               <xs:enumeration value="BTN"/>
20577               <xs:enumeration value="BWP"/>
20578               <xs:enumeration value="BYR"/>
20579               <xs:enumeration value="BZD"/>
20580               <xs:enumeration value="CAD"/>
20581               <xs:enumeration value="CDF"/>
20582               <xs:enumeration value="CHE"/>
20583               <xs:enumeration value="CHF"/>
20584               <xs:enumeration value="CHW"/>
20585               <xs:enumeration value="CLF"/>
20586               <xs:enumeration value="CLP"/>
20587               <xs:enumeration value="CNY"/>
20588               <xs:enumeration value="COP"/>
20589               <xs:enumeration value="COU"/>
20590               <xs:enumeration value="CRC"/>
20591               <xs:enumeration value="CUC"/>
20592               <xs:enumeration value="CUP"/>
20593               <xs:enumeration value="CVE"/>
20594               <xs:enumeration value="CZK"/>
20595               <xs:enumeration value="DJF"/>
20596               <xs:enumeration value="DKK"/>
20597               <xs:enumeration value="DOP"/>
20598               <xs:enumeration value="DZD"/>
20599               <xs:enumeration value="EGP"/>
20600               <xs:enumeration value="ERN"/>
20601               <xs:enumeration value="ETB"/>
20602               <xs:enumeration value="EUR"/>
20603               <xs:enumeration value="FJD"/>
20604               <xs:enumeration value="FKP"/>
20605               <xs:enumeration value="GBP"/>
20606               <xs:enumeration value="GEL"/>
20607               <xs:enumeration value="GHS"/>
20608               <xs:enumeration value="GIP"/>
20609               <xs:enumeration value="GMD"/>
20610               <xs:enumeration value="GNF"/>
20611               <xs:enumeration value="GTQ"/>
20612               <xs:enumeration value="GYD"/>
20613               <xs:enumeration value="HKD"/>
20614               <xs:enumeration value="HNL"/>
20615               <xs:enumeration value="HRK"/>
20616               <xs:enumeration value="HTG"/>
20617               <xs:enumeration value="HUF"/>
20618               <xs:enumeration value="IDR"/>
20619               <xs:enumeration value="ILS"/>
20620               <xs:enumeration value="INR"/>
20621               <xs:enumeration value="IQD"/>
20622               <xs:enumeration value="IRR"/>
20623               <xs:enumeration value="ISK"/>
20624               <xs:enumeration value="JMD"/>
20625               <xs:enumeration value="JOD"/>
20626               <xs:enumeration value="JPY"/>
20627               <xs:enumeration value="KES"/>
20628               <xs:enumeration value="KGS"/>
20629               <xs:enumeration value="KHR"/>
20630               <xs:enumeration value="KMF"/>
20631               <xs:enumeration value="KPW"/>
```



```
            <xs:enumeration value="KRW"/>
            <xs:enumeration value="KWD"/>
            <xs:enumeration value="KYD"/>
            <xs:enumeration value="KZT"/>
            <xs:enumeration value="LAK"/>
            <xs:enumeration value="LBP"/>
            <xs:enumeration value="LKR"/>
            <xs:enumeration value="LRD"/>
            <xs:enumeration value="LSL"/>
            <xs:enumeration value="LYD"/>
            <xs:enumeration value="MAD"/>
            <xs:enumeration value="MDL"/>
            <xs:enumeration value="MGA"/>
            <xs:enumeration value="MKD"/>
            <xs:enumeration value="MMK"/>
            <xs:enumeration value="MNT"/>
            <xs:enumeration value="MOP"/>
            <xs:enumeration value="MRO"/>
            <xs:enumeration value="MUR"/>
            <xs:enumeration value="MVR"/>
            <xs:enumeration value="MWK"/>
            <xs:enumeration value="MXN"/>
            <xs:enumeration value="MXV"/>
            <xs:enumeration value="MYR"/>
            <xs:enumeration value="MZN"/>
            <xs:enumeration value="NAD"/>
            <xs:enumeration value="NGN"/>
            <xs:enumeration value="NIO"/>
            <xs:enumeration value="NOK"/>
            <xs:enumeration value="NPR"/>
            <xs:enumeration value="NZD"/>
            <xs:enumeration value="OMR"/>
            <xs:enumeration value="PAB"/>
            <xs:enumeration value="PEN"/>
            <xs:enumeration value="PGK"/>
            <xs:enumeration value="PHP"/>
            <xs:enumeration value="PKR"/>
            <xs:enumeration value="PLN"/>
            <xs:enumeration value="PYG"/>
            <xs:enumeration value="QAR"/>
            <xs:enumeration value="RON"/>
            <xs:enumeration value="RSD"/>
            <xs:enumeration value="RUB"/>
            <xs:enumeration value="RWF"/>
            <xs:enumeration value="SAR"/>
            <xs:enumeration value="SBD"/>
            <xs:enumeration value="SCR"/>
            <xs:enumeration value="SDG"/>
            <xs:enumeration value="SEK"/>
            <xs:enumeration value="SGD"/>
            <xs:enumeration value="SHP"/>
            <xs:enumeration value="SLL"/>
            <xs:enumeration value="SOS"/>
            <xs:enumeration value="SRD"/>
            <xs:enumeration value="SSP"/>
            <xs:enumeration value="STD"/>
            <xs:enumeration value="SVC"/>
            <xs:enumeration value="SYP"/>
            <xs:enumeration value="SZL"/>
            <xs:enumeration value="THB"/>
            <xs:enumeration value="TJS"/>
            <xs:enumeration value="TMT"/>
            <xs:enumeration value="TND"/>
            <xs:enumeration value="TOP"/>
            <xs:enumeration value="TRY"/>
            <xs:enumeration value="TTD"/>
            <xs:enumeration value="TWD"/>
            <xs:enumeration value="TZS"/>
            <xs:enumeration value="UAH"/>
            <xs:enumeration value="UGX"/>
            <xs:enumeration value="USD"/>
            <xs:enumeration value="USN"/>
            <xs:enumeration value="UYI"/>
            <xs:enumeration value="UYU"/>
            <xs:enumeration value="UZS"/>
            <xs:enumeration value="VEF"/>
            <xs:enumeration value="VND"/>
            <xs:enumeration value="VUV"/>
```



```
            <xs:enumeration value="WST"/>
            <xs:enumeration value="XAF"/>
            <xs:enumeration value="XAG"/>
            <xs:enumeration value="XAU"/>
            <xs:enumeration value="XBA"/>
            <xs:enumeration value="XBB"/>
            <xs:enumeration value="XBC"/>
            <xs:enumeration value="XBD"/>
            <xs:enumeration value="XCD"/>
            <xs:enumeration value="XDR"/>
            <xs:enumeration value="XOF"/>
            <xs:enumeration value="XPD"/>
            <xs:enumeration value="XPF"/>
            <xs:enumeration value="XPT"/>
            <xs:enumeration value="XSU"/>
            <xs:enumeration value="XTS"/>
            <xs:enumeration value="XUA"/>
            <xs:enumeration value="XXX"/>
            <xs:enumeration value="YER"/>
            <xs:enumeration value="ZAR"/>
            <xs:enumeration value="ZMW"/>
            <xs:enumeration value="ZWL"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="AddressDeviceType">
        <xs:restriction base="xs:string">
            <xs:pattern value="d:_(i:[1-9][0-9]*|n:[a-zA-Z0-9-]+)_[^\p{Cc}\p{Cf}\p{Z}]+"/>
            <xs:maxLength value="256"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="AddressEntityType">
        <xs:restriction base="xs:unsignedInt"/>
    </xs:simpleType>
    <xs:simpleType name="AddressFeatureType">
        <xs:restriction base="xs:unsignedInt"/>
    </xs:simpleType>
    <xs:element name="device" type="ns_p:AddressDeviceType"/>
    <xs:element name="entity" type="ns_p:AddressEntityType"/>
    <xs:element name="feature" type="ns_p:AddressFeatureType"/>
    <xs:complexType name="DeviceAddressType">
        <xs:sequence>
            <xs:element minOccurs="0" ref="ns_p:device"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="DeviceAddressElementsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="device" type="ns_p:ElementTagType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="EntityAddressType">
        <xs:complexContent>
            <xs:extension base="ns_p:DeviceAddressType">
                <xs:sequence>
                    <xs:element minOccurs="0" ref="ns_p:entity" maxOccurs="unbounded"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="EntityAddressElementsType">
        <xs:complexContent>
            <xs:extension base="ns_p:DeviceAddressElementsType">
                <xs:sequence>
                    <xs:element minOccurs="0" name="entity" type="ns_p:ElementTagType"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="FeatureAddressType">
        <xs:complexContent>
            <xs:extension base="ns_p:EntityAddressType">
                <xs:sequence>
                    <xs:element minOccurs="0" ref="ns_p:feature"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="FeatureAddressElementsType">
        <xs:complexContent>
```



```
            <xs:extension base="ns_p:EntityAddressElementsType">
                <xs:sequence>
                    <xs:element minOccurs="0" name="feature" type="ns_p:ElementTagType"/>
                </xs:sequence>
            </xs:extension>
        </xs:complexContent>
    </xs:complexType>
    <xs:simpleType name="ScopeTypeType">
        <xs:union memberTypes="ns_p:ScopeTypeEnumType ns_p:EnumExtendType"/>
    </xs:simpleType>
    <xs:simpleType name="ScopeTypeEnumType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="ac"/>
            <xs:enumeration value="acCosPhiGrid"/>
            <xs:enumeration value="acCurrentA"/>
            <xs:enumeration value="acCurrentB"/>
            <xs:enumeration value="acCurrentC"/>
            <xs:enumeration value="acFrequencyGrid"/>
            <xs:enumeration value="acPowerA"/>
            <xs:enumeration value="acPowerB"/>
            <xs:enumeration value="acPowerC"/>
            <xs:enumeration value="acPowerLimitPct"/>
            <xs:enumeration value="acPowerTotal"/>
            <xs:enumeration value="acVoltageA"/>
            <xs:enumeration value="acVoltageB"/>
            <xs:enumeration value="acVoltageC"/>
            <xs:enumeration value="acYieldDay"/>
            <xs:enumeration value="acYieldTotal"/>
            <xs:enumeration value="dcCurrent"/>
            <xs:enumeration value="dcPower"/>
            <xs:enumeration value="dcString1"/>
            <xs:enumeration value="dcString2"/>
            <xs:enumeration value="dcString3"/>
            <xs:enumeration value="dcString4"/>
            <xs:enumeration value="dcString5"/>
            <xs:enumeration value="dcString6"/>
            <xs:enumeration value="dcTotal"/>
            <xs:enumeration value="dcVoltage"/>
            <xs:enumeration value="dhwTemperature"/>
            <xs:enumeration value="flowTemperature"/>
            <xs:enumeration value="outsideAirTemperature"/>
            <xs:enumeration value="returnTemperature"/>
            <xs:enumeration value="roomAirTemperature"/>
            <xs:enumeration value="charge"/>
            <xs:enumeration value="stateOfCharge"/>
            <xs:enumeration value="discharge"/>
            <xs:enumeration value="gridConsumption"/>
            <xs:enumeration value="gridFeedIn"/>
            <xs:enumeration value="selfConsumption"/>
            <xs:enumeration value="overloadProtection"/>
            <xs:enumeration value="acPower"/>
            <xs:enumeration value="acEnergy"/>
            <xs:enumeration value="acCurrent"/>
            <xs:enumeration value="acVoltage"/>
            <xs:enumeration value="batteryControl"/>
            <xs:enumeration value="simpleIncentiveTable"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="RoleType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="client"/>
            <xs:enumeration value="server"/>
            <xs:enumeration value="special"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="FeatureGroupType">
        <xs:restriction base="xs:string">
            <xs:pattern value="(#[1-9][0-9]*)*"/>
            <xs:maxLength value="128"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="DeviceTypeType">
        <xs:union memberTypes="ns_p:DeviceTypeEnumType ns_p:EnumExtendType"/>
    </xs:simpleType>
    <xs:simpleType name="DeviceTypeEnumType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="Dishwasher"/>
            <xs:enumeration value="Dryer"/>
```



```
            <xs:enumeration value="EnvironmentSensor"/>
            <xs:enumeration value="Generic"/>
            <xs:enumeration value="HeatGenerationSystem"/>
            <xs:enumeration value="HeatSinkSystem"/>
            <xs:enumeration value="HeatStorageSystem"/>
            <xs:enumeration value="HVACController"/>
            <xs:enumeration value="SubMeter"/>
            <xs:enumeration value="Washer"/>
            <xs:enumeration value="ElectricitySupplySystem"/>
            <xs:enumeration value="EnergyManagementSystem"/>
            <xs:enumeration value="Inverter"/>
            <xs:enumeration value="ChargingStation"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="EntityTypeType">
        <xs:union memberTypes="ns_p:EntityTypeEnumType ns_p:EnumExtendType"/>
    </xs:simpleType>
    <xs:simpleType name="EntityTypeEnumType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="Battery"/>
            <xs:enumeration value="Compressor"/>
            <xs:enumeration value="DeviceInformation"/>
            <xs:enumeration value="DHWCircuit"/>
            <xs:enumeration value="DHWStorage"/>
            <xs:enumeration value="Dishwasher"/>
            <xs:enumeration value="Dryer"/>
            <xs:enumeration value="ElectricalImmersionHeater"/>
            <xs:enumeration value="Fan"/>
            <xs:enumeration value="GasHeatingAppliance"/>
            <xs:enumeration value="Generic"/>
            <xs:enumeration value="HeatingBufferStorage"/>
            <xs:enumeration value="HeatingCircuit"/>
            <xs:enumeration value="HeatingObject"/>
            <xs:enumeration value="HeatingZone"/>
            <xs:enumeration value="HeatPumpAppliance"/>
            <xs:enumeration value="HeatSinkCircuit"/>
            <xs:enumeration value="HeatSourceCircuit"/>
            <xs:enumeration value="HeatSourceUnit"/>
            <xs:enumeration value="HVACController"/>
            <xs:enumeration value="HVACRoom"/>
            <xs:enumeration value="InstantDHWHeater"/>
            <xs:enumeration value="Inverter"/>
            <xs:enumeration value="OilHeatingAppliance"/>
            <xs:enumeration value="Pump"/>
            <xs:enumeration value="RefrigerantCircuit"/>
            <xs:enumeration value="SmartEnergyAppliance"/>
            <xs:enumeration value="SolarDHWStorage"/>
            <xs:enumeration value="SolarThermalCircuit"/>
            <xs:enumeration value="SubMeterElectricity"/>
            <xs:enumeration value="TemperatureSensor"/>
            <xs:enumeration value="Washer"/>
            <xs:enumeration value="BatterySystem"/>
            <xs:enumeration value="ElectricityGenerationSystem"/>
            <xs:enumeration value="ElectricityStorageSystem"/>
            <xs:enumeration value="GridConnectionPointOfPremises"/>
            <xs:enumeration value="Household"/>
            <xs:enumeration value="PVSystem"/>
            <xs:enumeration value="EV"/>
            <xs:enumeration value="EVSE"/>
            <xs:enumeration value="ChargingOutlet"/>
            <xs:enumeration value="CEM"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="FeatureTypeType">
        <xs:union memberTypes="ns_p:FeatureTypeEnumType ns_p:EnumExtendType"/>
    </xs:simpleType>
    <xs:simpleType name="FeatureTypeEnumType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="ActuatorLevel"/>
            <xs:enumeration value="ActuatorSwitch"/>
            <xs:enumeration value="Alarm"/>
            <xs:enumeration value="DataTunneling"/>
            <xs:enumeration value="DeviceClassification"/>
            <xs:enumeration value="DeviceDiagnosis"/>
            <xs:enumeration value="DirectControl"/>
            <xs:enumeration value="ElectricalConnection"/>
            <xs:enumeration value="Generic"/>
            <xs:enumeration value="HVAC"/>
```



```
            <xs:enumeration value="LoadControl"/>
            <xs:enumeration value="Measurement"/>
            <xs:enumeration value="Messaging"/>
            <xs:enumeration value="NetworkManagement"/>
            <xs:enumeration value="NodeManagement"/>
            <xs:enumeration value="OperatingConstraints"/>
            <xs:enumeration value="PowerSequences"/>
            <xs:enumeration value="Sensing"/>
            <xs:enumeration value="Setpoint"/>
            <xs:enumeration value="SmartEnergyManagementPs"/>
            <xs:enumeration value="TaskManagement"/>
            <xs:enumeration value="Threshold"/>
            <xs:enumeration value="TimeInformation"/>
            <xs:enumeration value="TimeTable"/>
            <xs:enumeration value="DeviceConfiguration"/>
            <xs:enumeration value="SupplyCondition"/>
            <xs:enumeration value="TimeSeries"/>
            <xs:enumeration value="TariffInformation"/>
            <xs:enumeration value="IncentiveTable"/>
            <xs:enumeration value="Bill"/>
            <xs:enumeration value="Identification"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="FeatureSpecificUsageType">
        <xs:union memberTypes="ns_p:FeatureSpecificUsageEnumType ns_p:EnumExtendType"/>
    </xs:simpleType>
    <xs:simpleType name="FeatureSpecificUsageEnumType">
        <xs:union
            memberTypes="ns_p:FeatureDirectControlSpecificUsageEnumType
ns_p:FeatureHvacSpecificUsageEnumType ns_p:FeatureMeasurementSpecificUsageEnumType
ns_p:FeatureSetpointSpecificUsageEnumType
ns_p:FeatureSmartEnergyManagementPsSpecificUsageEnumType"
        />
    </xs:simpleType>
    <xs:simpleType name="FeatureDirectControlSpecificUsageEnumType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="History"/>
            <xs:enumeration value="RealTime"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="FeatureHvacSpecificUsageEnumType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="OperationMode"/>
            <xs:enumeration value="Overrun"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="FeatureMeasurementSpecificUsageEnumType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="Contact"/>
            <xs:enumeration value="Electrical"/>
            <xs:enumeration value="Heat"/>
            <xs:enumeration value="Level"/>
            <xs:enumeration value="Pressure"/>
            <xs:enumeration value="Temperature"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="FeatureSetpointSpecificUsageEnumType">
        <xs:restriction base="ns_p:FeatureMeasurementSpecificUsageEnumType"/>
    </xs:simpleType>
    <xs:simpleType name="FeatureSmartEnergyManagementPsSpecificUsageEnumType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="FixedForecast"/>
            <xs:enumeration value="FlexibleChosenForecast"/>
            <xs:enumeration value="FlexibleOptionalForecast"/>
            <xs:enumeration value="OptionalSequenceBasedImmediateControl"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:simpleType name="FunctionType">
        <xs:union memberTypes="ns_p:FunctionEnumType ns_p:EnumExtendType"/>
    </xs:simpleType>
    <xs:simpleType name="FunctionEnumType">
        <xs:restriction base="xs:string">
            <xs:enumeration value="actuatorLevelData"/>
            <xs:enumeration value="actuatorLevelDescriptionData"/>
            <xs:enumeration value="actuatorSwitchData"/>
            <xs:enumeration value="actuatorSwitchDescriptionData"/>
            <xs:enumeration value="alarmListData"/>
            <xs:enumeration value="bindingManagementDeleteCall"/>
```



```
21022             <xs:enumeration value="bindingManagementEntryListData"/>
21023             <xs:enumeration value="bindingManagementRequestCall"/>
21024             <xs:enumeration value="dataTunnelingCall"/>
21025             <xs:enumeration value="deviceClassificationManufacturerData"/>
21026             <xs:enumeration value="deviceClassificationUserData"/>
21027             <xs:enumeration value="deviceDiagnosisHeartbeatData"/>
21028             <xs:enumeration value="deviceDiagnosisServiceData"/>
21029             <xs:enumeration value="deviceDiagnosisStateData"/>
21030             <xs:enumeration value="directControlActivityListData"/>
21031             <xs:enumeration value="directControlDescriptionData"/>
21032             <xs:enumeration value="electricalConnectionDescriptionListData"/>
21033             <xs:enumeration value="electricalConnectionParameterDescriptionListData"/>
21034             <xs:enumeration value="electricalConnectionStateListData"/>
21035             <xs:enumeration value="hvacOperationModeDescriptionListData"/>
21036             <xs:enumeration value="hvacOverrunDescriptionListData"/>
21037             <xs:enumeration value="hvacOverrunListData"/>
21038             <xs:enumeration value="hvacSystemFunctionDescriptionListData"/>
21039             <xs:enumeration value="hvacSystemFunctionListData"/>
21040             <xs:enumeration value="hvacSystemFunctionOperationModeRelationListData"/>
21041             <xs:enumeration value="hvacSystemFunctionPowerSequenceRelationListData"/>
21042             <xs:enumeration value="hvacSystemFunctionSetpointRelationListData"/>
21043             <xs:enumeration value="loadControlEventListData"/>
21044             <xs:enumeration value="loadControlStateListData"/>
21045             <xs:enumeration value="measurementConstraintsListData"/>
21046             <xs:enumeration value="measurementDescriptionListData"/>
21047             <xs:enumeration value="measurementListData"/>
21048             <xs:enumeration value="measurementThresholdRelationListData"/>
21049             <xs:enumeration value="messagingListData"/>
21050             <xs:enumeration value="networkManagementAbortCall"/>
21051             <xs:enumeration value="networkManagementAddNodeCall"/>
21052             <xs:enumeration value="networkManagementDeviceDescriptionListData"/>
21053             <xs:enumeration value="networkManagementDiscoverCall"/>
21054             <xs:enumeration value="networkManagementEntityDescriptionListData"/>
21055             <xs:enumeration value="networkManagementFeatureDescriptionListData"/>
21056             <xs:enumeration value="networkManagementJoiningModeData"/>
21057             <xs:enumeration value="networkManagementModifyNodeCall"/>
21058             <xs:enumeration value="networkManagementProcessStateData"/>
21059             <xs:enumeration value="networkManagementRemoveNodeCall"/>
21060             <xs:enumeration value="networkManagementReportCandidateData"/>
21061             <xs:enumeration value="networkManagementScanNetworkCall"/>
21062             <xs:enumeration value="nodeManagementBindingData"/>
21063             <xs:enumeration value="nodeManagementBindingDeleteCall"/>
21064             <xs:enumeration value="nodeManagementBindingRequestCall"/>
21065             <xs:enumeration value="nodeManagementDestinationListData"/>
21066             <xs:enumeration value="nodeManagementDetailedDiscoveryData"/>
21067             <xs:enumeration value="nodeManagementSubscriptionData"/>
21068             <xs:enumeration value="nodeManagementSubscriptionDeleteCall"/>
21069             <xs:enumeration value="nodeManagementSubscriptionRequestCall"/>
21070             <xs:enumeration value="operatingConstraintsDurationListData"/>
21071             <xs:enumeration value="operatingConstraintsInterruptListData"/>
21072             <xs:enumeration value="operatingConstraintsPowerDescriptionListData"/>
21073             <xs:enumeration value="operatingConstraintsPowerLevelListData"/>
21074             <xs:enumeration value="operatingConstraintsPowerRangeListData"/>
21075             <xs:enumeration value="operatingConstraintsResumeImplicationListData"/>
21076             <xs:enumeration value="powerSequenceAlternativesRelationListData"/>
21077             <xs:enumeration value="powerSequenceDescriptionListData"/>
21078             <xs:enumeration value="powerSequenceNodeScheduleInformationData"/>
21079             <xs:enumeration value="powerSequencePriceCalculationRequestCall"/>
21080             <xs:enumeration value="powerSequencePriceListData"/>
21081             <xs:enumeration value="powerSequenceScheduleConfigurationRequestCall"/>
21082             <xs:enumeration value="powerSequenceScheduleConstraintsListData"/>
21083             <xs:enumeration value="powerSequenceScheduleListData"/>
21084             <xs:enumeration value="powerSequenceSchedulePreferenceListData"/>
21085             <xs:enumeration value="powerSequenceStateListData"/>
21086             <xs:enumeration value="powerTimeSlotScheduleConstraintsListData"/>
21087             <xs:enumeration value="powerTimeSlotScheduleListData"/>
21088             <xs:enumeration value="powerTimeSlotValueListData"/>
21089             <xs:enumeration value="resultData"/>
21090             <xs:enumeration value="sensingDescriptionData"/>
21091             <xs:enumeration value="sensingListData"/>
21092             <xs:enumeration value="setpointConstraintsListData"/>
21093             <xs:enumeration value="setpointDescriptionListData"/>
21094             <xs:enumeration value="setpointListData"/>
21095             <xs:enumeration value="smartEnergyManagementPsConfigurationRequestCall"/>
21096             <xs:enumeration value="smartEnergyManagementPsData"/>
21097             <xs:enumeration value="smartEnergyManagementPsPriceCalculationRequestCall"/>
21098             <xs:enumeration value="smartEnergyManagementPsPriceData"/>
21099             <xs:enumeration value="specificationVersionListData"/>
```



```
            <xs:enumeration value="subscriptionManagementDeleteCall"/>
            <xs:enumeration value="subscriptionManagementEntryListData"/>
            <xs:enumeration value="subscriptionManagementRequestCall"/>
            <xs:enumeration value="supplyConditionDescriptionListData"/>
            <xs:enumeration value="supplyConditionListData"/>
            <xs:enumeration value="supplyConditionThresholdRelationListData"/>
            <xs:enumeration value="taskManagementJobDescriptionListData"/>
            <xs:enumeration value="taskManagementJobListData"/>
            <xs:enumeration value="taskManagementJobRelationListData"/>
            <xs:enumeration value="taskManagementOverviewData"/>
            <xs:enumeration value="thresholdConstraintsListData"/>
            <xs:enumeration value="thresholdDescriptionListData"/>
            <xs:enumeration value="thresholdListData"/>
            <xs:enumeration value="timeDistributorData"/>
            <xs:enumeration value="timeDistributorEnquiryCall"/>
            <xs:enumeration value="timeInformationData"/>
            <xs:enumeration value="timePrecisionData"/>
            <xs:enumeration value="timeTableConstraintsListData"/>
            <xs:enumeration value="timeTableDescriptionListData"/>
            <xs:enumeration value="timeTableListData"/>
            <xs:enumeration value="deviceConfigurationKeyValueConstraintsListData"/>
            <xs:enumeration value="deviceConfigurationKeyValueListData"/>
            <xs:enumeration value="deviceConfigurationKeyValueDescriptionListData"/>
            <xs:enumeration value="loadControlLimitConstraintsListData"/>
            <xs:enumeration value="loadControlLimitDescriptionListData"/>
            <xs:enumeration value="loadControlLimitListData"/>
            <xs:enumeration value="loadControlNodeData"/>
            <xs:enumeration value="timeSeriesConstraintsListData"/>
            <xs:enumeration value="timeSeriesDescriptionListData"/>
            <xs:enumeration value="timeSeriesListData"/>
            <xs:enumeration value="tariffOverallConstraintsData"/>
            <xs:enumeration value="tariffListData"/>
            <xs:enumeration value="tariffBoundaryRelationListData"/>
            <xs:enumeration value="tariffTierRelationListData"/>
            <xs:enumeration value="tariffDescriptionListData"/>
            <xs:enumeration value="tierBoundaryListData"/>
            <xs:enumeration value="tierBoundaryDescriptionListData"/>
            <xs:enumeration value="commodityListData"/>
            <xs:enumeration value="tierListData"/>
            <xs:enumeration value="tierIncentiveRelationListData"/>
            <xs:enumeration value="tierDescriptionListData"/>
            <xs:enumeration value="incentiveListData"/>
            <xs:enumeration value="incentiveDescriptionListData"/>
            <xs:enumeration value="incentiveTableData"/>
            <xs:enumeration value="incentiveTableDescriptionData"/>
            <xs:enumeration value="incentiveTableConstraintsData"/>
            <xs:enumeration value="electricalConnectionPermittedValueSetListData"/>
            <xs:enumeration value="useCaseInformationListData"/>
            <xs:enumeration value="nodeManagementUseCaseData"/>
            <xs:enumeration value="billConstraintsListData"/>
            <xs:enumeration value="billDescriptionListData"/>
            <xs:enumeration value="billListData"/>
            <xs:enumeration value="identificationListData"/>
        </xs:restriction>
    </xs:simpleType>
    <xs:complexType name="PossibleOperationsClassifierType">
        <xs:sequence>
            <xs:element minOccurs="0" name="partial" type="ns_p:ElementTagType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="PossibleOperationsReadType">
        <xs:complexContent>
            <xs:extension base="ns_p:PossibleOperationsClassifierType"/>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="PossibleOperationsWriteType">
        <xs:complexContent>
            <xs:extension base="ns_p:PossibleOperationsClassifierType"/>
        </xs:complexContent>
    </xs:complexType>
    <xs:complexType name="PossibleOperationsType">
        <xs:sequence>
            <xs:element name="read" minOccurs="0" type="ns_p:PossibleOperationsReadType"/>
            <xs:element name="write" minOccurs="0" type="ns_p:PossibleOperationsWriteType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="PossibleOperationsElementsType">
        <xs:sequence>
```



```
            <xs:element name="read" minOccurs="0" type="ns_p:ElementTagType"/>
            <xs:element name="write" minOccurs="0" type="ns_p:ElementTagType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="FunctionPropertyType">
        <xs:sequence>
            <xs:element minOccurs="0" name="function" type="ns_p:FunctionType"/>
            <xs:element minOccurs="0" name="possibleOperations"
type="ns_p:PossibleOperationsType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:complexType name="FunctionPropertyElementsType">
        <xs:sequence>
            <xs:element minOccurs="0" name="function" type="ns_p:ElementTagType"/>
            <xs:element minOccurs="0" name="possibleOperations"
type="ns_p:PossibleOperationsElementsType"/>
        </xs:sequence>
    </xs:complexType>
</xs:schema>
```

A.3.4 Datagram

File name: EEBus_SPINE_TS_Datagram.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
    Smart Premises Interoperable Neutral-Message Exchange (SPINE)
    Version 1.1.1
    2018-12-21
    Copyright (c) 2018 EEBus Initiative e.V.  All Rights Reserved.
    Source: https://www.eebus.org/en/specifications/
-->
<xs:schema xmlns:ns_p="http://docs.eebus.org/spine/xsd/v1"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://docs.eebus.org/spine/xsd/v1" version="1.1.1" blockDefault="#all"
elementFormDefault="qualified">
    <xs:annotation>
        <xs:documentation>EEBus SPINE Specification schema. Copyright 2018 EEBus Initiative
e.V.  All rights reserved.</xs:documentation>
    </xs:annotation>
    <xs:include schemaLocation="EEBus_SPINE_TS_CommandFrame.xsd"/>
    <xs:complexType name="HeaderType">
        <xs:sequence>
            <xs:element minOccurs="0" name="specificationVersion"
type="ns_p:SpecificationVersionType"/>
            <xs:element minOccurs="0" name="addressSource" type="ns_p:FeatureAddressType"/>
            <xs:element minOccurs="0" name="addressDestination"
type="ns_p:FeatureAddressType"/>
            <xs:element minOccurs="0" name="addressOriginator"
type="ns_p:FeatureAddressType"/>
            <xs:element minOccurs="0" name="msgCounter" type="ns_p:MsgCounterType"/>
            <xs:element minOccurs="0" name="msgCounterReference" type="ns_p:MsgCounterType"/>
            <xs:element minOccurs="0" name="cmdClassifier" type="ns_p:CmdClassifierType"/>
            <xs:element minOccurs="0" name="ackRequest" type="xs:boolean"/>
            <xs:element minOccurs="0" name="timestamp"
type="ns_p:AbsoluteOrRelativeTimeType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="header" type="ns_p:HeaderType"/>
    <xs:complexType name="PayloadType">
        <xs:sequence>
            <xs:element maxOccurs="unbounded" minOccurs="0" name="cmd" type="ns_p:CmdType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="payload" type="ns_p:PayloadType"/>
    <xs:complexType name="DatagramType">
        <xs:sequence>
            <xs:element ref="ns_p:header"/>
            <xs:element ref="ns_p:payload"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="datagram" type="ns_p:DatagramType"/>
```



```
</xs:schema>
```

A.3.5 Result

File name: EEBus_SPINE_TS_Result.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
    Smart Premises Interoperable Neutral-Message Exchange (SPINE)
    Version 1.1.1
    2018-12-21
    Copyright (c) 2018 EEBus Initiative e.V.  All Rights Reserved.
    Source: https://www.eebus.org/en/specifications/
-->
<xs:schema xmlns:ns_p="http://docs.eebus.org/spine/xsd/v1"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
targetNamespace="http://docs.eebus.org/spine/xsd/v1" version="1.1.1"
    blockDefault="#all" elementFormDefault="qualified">
    <xs:annotation>
        <xs:documentation>EEBus SPINE Specification schema. Copyright 2018 EEBus Initiative
e.V.  All rights reserved.</xs:documentation>
    </xs:annotation>
    <xs:include schemaLocation="EEBus_SPINE_TS_CommonDataTypes.xsd"/>
    <xs:simpleType name="ErrorNumberType">
        <xs:restriction base="xs:unsignedInt"/>
    </xs:simpleType>
    <xs:complexType name="ResultDataType">
        <xs:sequence>
            <xs:element minOccurs="0" name="errorNumber" type="ns_p:ErrorNumberType"/>
            <xs:element minOccurs="0" name="description" type="ns_p:DescriptionType"/>
        </xs:sequence>
    </xs:complexType>
    <xs:element name="resultData" type="ns_p:ResultDataType"/>
</xs:schema>
```

A.3.6 Overview example

For each SPINE class, an additional *_overview.xsd is defined that references the functions defined by this class. The following example shows this for the class ActuatorLevel.

File name: EEBus_SPINE_TS_ActuatorLevel_overview.xsd

```
<?xml version="1.0" encoding="UTF-8"?>
<!--
    Smart Premises Interoperable Neutral-Message Exchange (SPINE)
    Version 1.1.1
    2018-12-21
    Copyright (c) 2018 EEBus Initiative e.V.  All Rights Reserved.
    Source: https://www.eebus.org/en/specifications/
-->
<xs:schema xmlns:ns_p="http://docs.eebus.org/spine/xsd/v1"
xmlns:xs="http://www.w3.org/2001/XMLSchema"
    targetNamespace="http://docs.eebus.org/spine/xsd/v1" version="1.1.1" blockDefault="#all"
elementFormDefault="qualified">
    <xs:annotation>
        <xs:documentation>EEBus SPINE Specification schema. Copyright 2018 EEBus Initiative
e.V.  All rights reserved.</xs:documentation>
    </xs:annotation>
    <xs:include schemaLocation="EEBus_SPINE_TS_ActuatorLevel.xsd"/>
    <xs:group name="group_actuatorLevel">
        <xs:sequence>
            <xs:group ref="ns_p:actuatorLevel"/>
            <xs:group ref="ns_p:actuatorLevelDescription"/>
```



```
        </xs:sequence>
    </xs:group>
    <xs:group name="actuatorLevel">
        <xs:sequence>
            <xs:element ref="ns_p:actuatorLevelData"/>
        </xs:sequence>
    </xs:group>
    <xs:group name="actuatorLevelDescription">
        <xs:sequence>
            <xs:element ref="ns_p:actuatorLevelDescriptionData"/>
        </xs:sequence>
    </xs:group>
</xs:schema>
```


Annex B - Resource Definitions Overview

B.1 Introduction

All Well Known Identifiers, defined by the EEBus Initiative e.V., are listed in this chapter. Detailed specifications can be found in the referenced sections.

B.2 Device Types

For detailed descriptions, please consider the corresponding device type definitions.

Device Type	Section	Description
ChargingStation	4.1.1	Represents a Charging Station, typically used in the context of electrical vehicle charging.
Dishwasher	4.1.2	A machine that cleans the dishes.
Dryer	4.1.3	A dryer for clothes (laundry) after they were washed.
ElectricitySupplySystem	4.1.4	Represents a set of sources and storages of electric power of a premises.
EnergyManagementSystem	4.1.5	Represents an Energy Management System (EMS).
EnvironmentSensor	4.1.6	Represents an environment sensor appliance.
Generic	4.1.7	Device with no specific content. May be used for developing, testing or evaluating purposes.
HeatGenerationSystem	4.1.8	Represents a heat generation system consisting of one or more heat generation appliances (for example gas boilers, heat pumps, solar thermal systems).
HeatSinkSystem	4.1.9	Represents the end user relevant HVAC system parts (heating circuits, DHW circuits, ventilation system) and system functions (heating, cooling, DHW, ventilation).
HeatStorageSystem	4.1.10	Represents a system of heat storages consisting of one or more heat storage appliances.
HVACController	4.1.11	Represents (as SPINE Device) the central controller appliance of the HVAC System.
Inverter	4.1.12	Represents a power Inverter.
SubMeter	4.1.13	Metering device for households or devices that is not billing relevant.
Washer	4.1.14	A washer for clothes (laundry).

Table 334: Well known device types

B.3 Entity Types

For detailed descriptions and rules, please consider the corresponding entity type definitions.

Entity Type	Section	Description
Battery	4.2.1	Appliance for storing (electrical) energy.
BatterySystem	4.2.2	A system consisting of one or more battery storages and one or more battery inverters.

CEM	4.2.3	Abbreviation for Customer Energy Manager and typically used to represent an Energy Manager within SPINE.
ChargingOutlet	4.2.4	Represents a Charging Outlet of a Charging Station.
Compressor	4.2.5	Represents the compressor of (e.g.) a heat pump appliance.
DeviceInformation	4.2.6	A special entity representing the complete device! All features that are included here hold information about the device and all entities.
DHWCircuit	4.2.7	Represents all elements for domestic hot water supply like cold water line, domestic hot water line, circulation line, tapping points. Should not include DHW storages, these should be represented as Entity Type “DHWStorage”.
DHWStorage	4.2.8	Represents a heat storage appliance for storing domestic hot water.
Dishwasher	4.2.9	A functionality that cleans the dishes.
Dryer	4.2.10	A dryer for clothes (laundry) after they were washed.
ElectricalImmersionHeater	4.2.11	Represents an electrical immersion heater (electrical resistance heater) for example as part of a heat pump appliance, a heat source unit or a DHW storage.
ElectricityGenerationSystem	4.2.12	A system that is capable of producing electric power through local generation.
ElectricityStorageSystem	4.2.13	A system that is capable of storing electric power and releasing the majority of it as electric power.
EV	4.2.14	Represents an Electrical Vehicle.
EVSE	4.2.15	Abbreviation for Electrical Vehicle Supply Equipment, typically used to connect an Electical Vehicle to a Charging Station.
Fan	4.2.16	Represents an air fan, for example of an air water heat pump unit or a ventilation unit.
GasHeatingAppliance	4.2.17	Represents a heating appliance that generates heat (for example for heating or domestic hot water) by burning natural gas.
Generic	4.2.18	Entity with no specific content. May be used for developing, testing or evaluating purposes. Client functionality may be implemented within this entity, too.
GridConnectionPointOfPremises	4.2.19	The electrical connection point between the utility electricity grid and the electric circuit of a premises.
HeatingBufferStorage	4.2.20	Represents a heat storage appliance for storing heating water.
HeatingCircuit	4.2.21	Represents a single heating circuit for space heating, including flow and return lines, optional mixing units (for mixed heating circuits) and the heat distribution: a) directly to one or more heated rooms (see HVACRoom) b) to one or more heating zones (see HeatingZone).

HeatingObject	4.2.22	A more general entity representing functionalities that convert (electrical) energy into heat.
HeatingZone	4.2.23	Represents a heating zone that provides heating energy to one or more heated rooms (see HVACRoom) One or more heating zones can be part of a heating circuit.
HeatPumpAppliance	4.2.24	Represents either a complete heat pump appliance (for example a brine water heat pump, indoor; an air water heat pump, completely indoor or completely outdoor) or the indoor part of a refrigerant split heat pump combination.
HeatSinkCircuit	4.2.25	Represents the circuit of a heat generator that delivers the heat energy to the heat sinks (for example to a heating circuit).
HeatSourceCircuit	4.2.26	Represents the circuit that delivers heat energy from an environmental heat source to a heat pump appliance (for example a brine circuit).
HeatSourceUnit	4.2.27	Represents an additional unit that provides the heat from the environmental heat source to the heat pump appliance (for example an outdoor ventilation unit that transfers the heat energy from the outside air to the heat source circuit of the heat pump appliance).
Household	4.2.28	A more virtual entity representing a complete household.
HVACController	4.2.29	Represents the central controller appliance of the HVAC System (as a SPINE Entity).
HVACRoom	4.2.30	Represents one physical room in a building/house that can be heated (H) / cooled, ventilated (V) or air conditioned (AC).
InstantDHWHeater	4.2.31	Represents an appliance for instant generation of domestic hot water (for example by using heat from a heating buffer storage).
Inverter	4.2.32	Represents an appliance that converts the dc-voltage (e.g. of a PV string) into ac-voltage.
OilHeatingAppliance	4.2.33	Represents a heating appliance that generates heat (for example for heating or domestic hot water) by burning oil.
Pump	4.2.34	Represents a pump to generate a flow rate (for example in a heating circuit).
PVSystem	4.2.35	A system consisting of one or more PV inverters with one or more PV modules. The PVSystem provides aggregated PV related system parts of the house or premises.
RefrigerantCircuit	4.2.36	Represents the Refrigerant circuit of e.g. a heat pump appliance or a fridge or freezer.
SmartEnergyAppliance	4.2.37	Represents an appliance that allows shifting of its load via the SmartEnergyManagementPs concepts.
SolarDHWStorage	4.2.38	Represents a heat storage appliance for storing domestic hot water. The heat energy can be provided especially by solar thermal energy.

SolarThermalCircuit	4.2.39	Represents a single circuit of a solar thermal system including a solar thermal collector field and flow and return pipes to/from the collector field and pumps.
SubMeterElectricity	4.2.40	Electricity metering for households or functionalities that is not billing relevant.
TemperatureSensor	4.2.41	Represents a sensor that can provide a temperature measurement value.
Washer	4.2.42	A washer for clothes (laundry).

21343 *Table 335: Well known entity types*

21344

21345 ## B.4 Feature Types

21346 For detailed descriptions and rules, please consider the corresponding feature type definitions.

Feature Type	Section	Description
ActuatorLevel	4.3.1	Standard Feature Type for the Standard Class ActuatorLevel.
ActuatorSwitch	4.3.2	Standard Feature Type for the Standard Class ActuatorSwitch.
Alarm	4.3.3	Standard Feature Type for the Standard Class Alarm.
Bill	4.3.4	Standard Feature Type for the Standard Class Bill.
DataTunneling	4.3.5	Standard Feature Type for the Standard Class DataTunneling.
DeviceClassification	4.3.6	Standard Feature Type for the Standard Class DeviceClassification.
DeviceConfiguration	4.3.7	Standard Feature Type for the Standard Class DeviceConfiguration.
DeviceDiagnosis	4.3.8	Standard Feature Type for the Standard Class DeviceDiagnosis.
DirectControl	4.3.9	Standard Feature Type for the Standard Class DirectControl.
ElectricalConnection	4.3.10	Standard Feature Type for the Standard Class ElectricalConnection.
Generic	4.3.11	Feature with no specific content. May be used for developing, testing or evaluating purposes. Client functionality may be implemented within this feature, too.
HVAC	4.3.12	Standard Feature Type for the Standard Class HVAC.
Identification	4.3.13	Standard Feature Type for the Standard Class Identification.
IncentiveTable	4.3.14	Standard Feature Type for the Complex Class IncentiveTable.
LoadControl	4.3.15	Standard Feature Type for the Standard Class LoadControl.
Measurement	4.3.16	Standard Feature Type for the Standard Class Measurement.
Messaging	4.3.17	Standard Feature Type for the Standard Class Messaging.

NetworkManagement	4.3.18	Standard Feature Type for the Standard Class NetworkManagement.
NodeManagement	4.3.19	Standard Feature Type for the Complex Class NodeManagement.
OperatingConstraints	4.3.20	Standard Feature Type for the Standard Class OperatingConstraints.
PowerSequences	4.3.21	Standard Feature Type for the Standard Class PowerSequences.
Sensing	4.3.22	Standard Feature Type for the Standard Class Sensing.
Setpoint	4.3.23	Standard Feature Type for the Standard Class Setpoint.
SmartEnergyManagementPs	4.3.24	Standard Feature Type for the Complex Class SmartEnergyManagementPs.
SupplyCondition	4.3.25	Standard Feature Type for the Standard Class SupplyCondition.
TariffInformation	4.3.26	Standard Feature Type for the Standard Class TariffInformation.
TaskManagement	4.3.27	Standard Feature Type for the Standard Class TaskManagement.
Threshold	4.3.28	Standard Feature Type for the Standard Class Threshold.
TimeInformation	4.3.29	Standard Feature Type for the Standard Class TimeInformation.
TimeSeries	4.3.30	Standard Feature Type for the Standard Class TimeSeries.
TimeTable	4.3.31	Standard Feature Type for the Standard Class TimeTable.

Table 336: Well known feature types

B.5 Complex Classes

For detailed descriptions and rules, please consider the corresponding complex class definitions.

Complex Class	Section	Description
IncentiveTable	5.2.1	The IncentiveTable data model allows to define tariffs with tiers and different incentives and boundaries.
NodeManagement	5.2.2	All data needed to manage nodes is combined in this complex class. Functions from the standard classes *NetworkManagement*, *BindingManagement*, *SubscriptionManagement* and *Version* are used.
SmartEnergyManagementPs	5.2.3	Shifting the load of devices in some given constraints is done with this complex class. Functions from the standard classes *PowerSequences* and *OperatingConstraints* are used.

Table 337: Well known complex classes

B.6 Standard Classes

For detailed descriptions and rules, please consider the corresponding standard class definitions.

Standard Class	Section	Description
ActuatorLevel	5.3.1	Dimming actuators, actuators with adjustability in stages.
ActuatorSwitch	5.3.2	Switch actuators.
Alarm	5.3.3	Alarms, generated due to exceeding thresholds (see corresponding class) or other events.
Bill	5.3.4	Bills for costs and profits of different purposes (e.g. a charging summary).
BindingManagement	5.3.5	Request / delete bindings or list existing bindings.
DataTunneling	5.3.6	Tunnels any (unspecified) data.
DeviceClassification	5.3.7	Free text information about a device (name, serial, etc.) and some enumerations like device type.
DeviceConfiguration	5.3.8	Used to model configuration data needed for interoperable functionality not fitting any other SPINE class.
DeviceDiagnosis	5.3.9	Information about the state of a device, heartbeat counting and service data.
DirectControl	5.3.10	Notification of instantaneous consumption/generation as well as instantaneous control.
ElectricalConnection	5.3.11	Information about specific parameters used for electrical connections, with relation to Measurement.
HVAC	5.3.12	Monitoring and controlling HVAC system functions (heating, cooling, ventilation, air conditioning, ...) and corresponding operation modes and overrun functionalities.
Identification	5.3.13	The Identification Class can be used to express certain identification values of the parent Entity.
LoadControl	5.3.14	Enabling (partial) (external) management of the load of a household or device by (e.g.) a DSO or some energy manager.
Measurement	5.3.15	General measurements (e.g. temperature, voltage, ...).
Messaging	5.3.16	Text messages (such as error messages, information, warnings, ...).
NetworkManagement	5.3.17	Add devices, remove etc., List of Inventory.
OperatingConstraints	5.3.18	Information on the power/energy constraints and implications.
PowerSequences	5.3.19	PowerSequences can be used to model energy management on a schedule basis.
Sensing	5.3.20	Various sensors, switches, contact sensors (e.g. window contact), smoke detectors, etc.
Setpoint	5.3.21	Setting desired values to control some functionality (with reference to Measurement and TimeTable).
SubscriptionManagement	5.3.22	Request / delete subscriptions or list existing subscriptions.
SupplyCondition	5.3.23	Condition of power supply, limitations, etc.
TariffInformation	5.3.24	The TariffInformation Class provides functions for modelling tariffs.
TaskManagement	5.3.25	General status of device- or entity-wide tasks (triggered by user, internal mechanisms or external configuration).
Threshold	5.3.26	Thresholds for automatic behaviour of a device (may generate alarms for exceeding values).
TimeInformation	5.3.27	Time information (including the approximate synchronisation).
TimeSeries	5.3.28	TimeSeries is a versatile class that allows to express time dependent values as a time series of values.

TimeTable	5.3.29	Providing some amount of tables and slots to be used by other classes (e.g. Setpoint).
UseCaseInformation	5.3.30	To model which Use Cases are supported by a node the UseCaseInformation Class can be used.
Version	5.3.31	SPINE version information.

Table 338: Well known standard classes

B.7 Identifier list

The following table list all identifiers that belong to the identifiers concept as described in section 3.4.

Type Name	Used in elements	Base Type
AbsoluteOrRelativeTimeType	timestamp	*Union of* xs:duration *and* xs:dateTime
AddressDeviceType	device	xs:string
AddressEntityType	entity	xs:unsignedInt
AddressFeatureType	feature	xs:unsignedInt
AlarmIdType	alarmId	xs:unsignedInt
AlternativesIdType	alternativesId	xs:unsignedInt
BillCostIdType	costId	xs:unsignedInt
BillIdType	billId	xs:unsignedInt
BillPositionIdType	positionId	xs:unsignedInt
BillValueIdType	valueId	xs:unsignedInt
BindingIdType	bindingId	xs:unsignedInt
CommodityIdType	commodityId	xs:unsignedInt
ConditionIdType	conditionId	xs:unsignedInt
DeviceConfigurationKeyIdType	keyId	xs:unsignedInt
ElectricalConnectionIdType	electricalConnectionId	xs:unsignedInt
ElectricalConnectionParameterIdType	parameterId	xs:unsignedInt
HvacOperationModeIdType	operationModeId	xs:unsignedInt
HvacOverrunIdType	overrunId	xs:unsignedInt
HvacSystemFunctionIdType	systemFunctionId	xs:unsignedInt
IdentificationIdType	identificationId	xs:unsignedInt
IncentiveIdType	incentiveId	xs:unsignedInt
LoadControlEventIdType	eventId	xs:unsignedInt
LoadControlLimitIdType	limitId	xs:unsignedInt
MeasurementIdType	measurementId	xs:unsignedInt
MessagingNumberType	messagingNumber	xs:unsignedInt
PowerSequenceIdType	sequenceId	xs:unsignedInt
PowerTimeSlotNumberType	slotNumber	xs:unsignedInt
SetpointIdType	setpointId	xs:unsignedInt
SubscriptionIdType	subscriptionId	xs:unsignedInt
TariffIdType	tariffId	xs:unsignedInt
TaskManagementJobIdType	jobId	xs:unsignedInt
ThresholdIdType	thresholdId	xs:unsignedInt

TierBoundaryIdType	boundaryId	xs:unsignedInt
TierIdType	tierId	xs:unsignedInt
TimeSeriesIdType	timeSeriesId	xs:unsignedInt
TimeSeriesSlotIdType	timeSeriesSlotId	xs:unsignedInt
TimeSlotIdType	timeSlotId	xs:unsignedInt
TimeTableIdType	timeTableId	xs:unsignedInt

21360 *Table 339: Identifier overview*

21361

