This document contains the specification draft for the Lemonbeat Application Layer.
Contents

1 Introduction ........................................... 4
   1.1 Document revision history .......................... 5

2 Application ............................................. 6
   2.1 Application layer .................................. 7
      2.1.1 NTP ........................................ 7
         2.1.1.1 NTP timestamps .......................... 7
         2.1.1.2 NTP frame .............................. 7
         2.1.1.3 Clock synchronization algorithm .... 7
   2.2 Protocol description ............................... 9
      2.2.1 Services .................................... 9
      2.2.2 Network management ......................... 10
      2.2.3 Device Configuration ......................... 11
         2.2.3.1 Virtual Values ......................... 11
         2.2.3.2 Partner Service ......................... 12
         2.2.3.3 Timer Service ......................... 12
         2.2.3.4 Calendar Service ....................... 12
         2.2.3.5 Action Service ........................ 12
         2.2.3.6 Calculation Service .................... 12
         2.2.3.7 State Machine Service ............... 12
         2.2.3.8 Configuration Service ................. 13
         2.2.3.9 Status Service ........................ 14
         2.2.3.10 Firmware Update Service ............. 15
   2.3 XML description .................................. 16
      2.3.1 Network and device .......................... 16
         2.3.1.1 Tag description ....................... 16
         2.3.1.2 Example ................................ 16
         2.3.1.3 Network and device tags .............. 16
      2.3.2 Network management .......................... 17
         2.3.2.1 Tag description ....................... 17
         2.3.2.2 Inclusion data ........................ 17
         2.3.2.3 Examples ................................ 18
      2.3.3 Public key .................................... 19
         2.3.3.1 Tag description ....................... 19
         2.3.3.2 Key types ................................ 19
         2.3.3.3 Examples ................................ 20
      2.3.4 Service description ........................... 21
         2.3.4.1 Tag description ....................... 21
         2.3.4.2 Service ................................. 21
         2.3.4.3 Example ................................ 22
      2.3.5 Memory information ............................ 23
         2.3.5.1 Tag description ....................... 23
         2.3.5.2 Memory .................................. 23
         2.3.5.3 Examples ................................ 24
      2.3.6 Device description ............................ 25
         2.3.6.1 Tag description ....................... 25
2.4 User stories

2.4.1 Remote Control with wall switch controlling multiple devices
2.4.1.1 Illustration
2.4.1.2 XML
2.4.2 Temperature Control with calendar tasks
2.4.2.1 Illustration
2.4.2.2 XML
2.4.3 Temperature Control with window and temperature sensor
2.4.3.1 Illustration
2.4.3.2 XML
2.4.4 Light control with movement and luminance sensor
2.4.4.1 Illustration
2.4.4.2 XML
2.4.5 Washing machine control
2.4.5.1 Illustration
2.4.5.2 XML
2.4.6 Electricity pricing
2.4.6.1 Illustration
2.4.6.2 XML
List of Figures 126
List of Corrections 127
Chapter 1

Introduction

The Lemonbeat Protocol is a Protocol aimed to provide a solution for low cost and low power connectivity for devices using batteries as a power source. The Protocol aims to solve the problems that exist in currently available protocols. The problems in current protocols are among others:

- Missing or bad security
- Bad robustness
- Interference with other technology
- Requirements for a gateway centric system are not met
- Duty cycle limits
- Range
- Bad routing strategies
- Not standards based
- No good application layer

All layers and services in Lemonbeat from the data link layer and up are based on Internet standards. The application layer described in this document is self-descriptive and extendable and based on standard technology that is widely used in the internet today.
### 1.1 Document revision history

<table>
<thead>
<tr>
<th>Ver.</th>
<th>Rev.</th>
<th>Description</th>
<th>Authors</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0</td>
<td>Initial draft version</td>
<td>Andreas Madsen (Seluxit), Daniel Lux (Seluxit), Henrik Sorensen (Seluxit), Leni Lausdahl (Seluxit), Morten Frederiksen (Seluxit)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reviewed by</td>
<td>Michael Westermeier (RWE)</td>
</tr>
<tr>
<td>1</td>
<td>3</td>
<td>Updated value types and units. Updated network inclusion. Removed unused tags and attributes.</td>
<td>Andreas Madsen (Seluxit), Daniel Lux (Seluxit), Henrik Sorensen (Seluxit), Morten Frederiksen (Seluxit)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reviewed by</td>
<td>Michael Westermeier (RWE)</td>
</tr>
<tr>
<td>1</td>
<td>4</td>
<td>Changed memory id's Renamed conditions to calculation</td>
<td>Andreas Madsen (Seluxit), Daniel Lux (Seluxit), Henrik Sorensen (Seluxit), Morten Frederiksen (Seluxit)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reviewed by</td>
<td>Michael Westermeier (RWE)</td>
</tr>
<tr>
<td>1</td>
<td>5</td>
<td>Changed PHY header to better support forward error correction.</td>
<td>Andreas Madsen (Seluxit), Daniel Lux (Seluxit), Henrik Sorensen (Seluxit), Morten Frederiksen (Seluxit)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reviewed by</td>
<td>Michael Westermeier (RWE)</td>
</tr>
<tr>
<td>1</td>
<td>6</td>
<td>Converted document to \LaTeX \ \ Change PHY header for Forward Error Correction Updated services due to new XSD's Added status and configuration service Remove Partner Link service</td>
<td>Andreas Bomholtz (Seluxit), Daniel Lux (Seluxit), Henrik Sorensen (Seluxit), Morten Frederiksen (Seluxit)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reviewed by</td>
<td>Michael Westermeier (RWE)</td>
</tr>
<tr>
<td>1</td>
<td>7</td>
<td>Updated the status service</td>
<td>Andreas Bomholtz (Seluxit), Daniel Lux (Seluxit), Henrik Sorensen (Seluxit), Morten Frederiksen (Seluxit)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reviewed by</td>
<td>Michael Westermeier (RWE)</td>
</tr>
<tr>
<td>1</td>
<td>8</td>
<td>Added new Generic MAC options Added new special virtual values</td>
<td>Andreas Bomholtz (Seluxit), Daniel Lux (Seluxit), Henrik Sorensen (Seluxit), Morten Frederiksen (Seluxit)</td>
</tr>
<tr>
<td></td>
<td></td>
<td>Reviewed by</td>
<td>Michael Westermeier (RWE)</td>
</tr>
</tbody>
</table>
Chapter 2

Application

In Section 2.1 the application layer is described. In Section 2.2 the Lemonbeat Application Protocol is described and in the next section XML example for the protocol is presented. In the last section in this chapter, user stories for some scenarios and the xml solutions are presented.
2.1 Application layer

This chapter defines the different frames used in the application layer of the Lemonbeat protocol.

2.1.1 NTP

The Network Time Protocol is used to synchronize the clocks of Lemonbeat devices. Based on the synchronized clocks a Wake on radio enabled device must calculate the points in time where the receiver is in the RX on state. Devices that need to wake up Wake on radio devices also use the synchronized clocks to calculate the correct time for sending a wake up frame. NTP uses the UDP on port number 123. Lemonbeat devices must implement the NTP protocol as defined in RFC-5905.

2.1.1.1 NTP timestamps

Lemonbeat devices use the native NTP Timestamp format which is defined as in Table 2.1.

<table>
<thead>
<tr>
<th>Data Size in Bits</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>32</td>
<td>Seconds since start of era (era 0 starts 0h 1 January 1900 UTC, negative numbers are before this date)</td>
</tr>
<tr>
<td>32</td>
<td>Fraction of a second</td>
</tr>
</tbody>
</table>

Table 2.1: NTP Timestamp

2.1.1.2 NTP frame

The NTP frame is built up as in Table 2.2.

<table>
<thead>
<tr>
<th>Size in Bits</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>2</td>
<td>Leap Indicator</td>
</tr>
<tr>
<td>3</td>
<td>Version Number</td>
</tr>
<tr>
<td>3</td>
<td>Mode</td>
</tr>
<tr>
<td>8</td>
<td>Stratum</td>
</tr>
<tr>
<td>8</td>
<td>Poll max interval</td>
</tr>
<tr>
<td>8</td>
<td>Precision</td>
</tr>
<tr>
<td>32</td>
<td>Root delay</td>
</tr>
<tr>
<td>32</td>
<td>Root dispersion</td>
</tr>
<tr>
<td>32</td>
<td>Reference ID</td>
</tr>
<tr>
<td>64</td>
<td>Reference timestamp</td>
</tr>
<tr>
<td>64</td>
<td>Origin timestamp</td>
</tr>
<tr>
<td>64</td>
<td>Receive timestamp</td>
</tr>
<tr>
<td>64</td>
<td>Transmit timestamp</td>
</tr>
<tr>
<td>X</td>
<td>Extension Field 1</td>
</tr>
<tr>
<td>X</td>
<td>Extension Field 2</td>
</tr>
<tr>
<td>32</td>
<td>Key Identifier</td>
</tr>
<tr>
<td>128</td>
<td>Digest</td>
</tr>
</tbody>
</table>

Table 2.2: NTP Frame

In Lemonbeat all fields but the Extension Field 1, Extension Field 2, Key Identifier and Digest are used.

2.1.1.3 Clock synchronization algorithm

Lemonbeat devices must only synchronize time with devices that are in direct range, i.e. no routers are used when communicating with the other device is needed. When sending NTP frames, the CSMA-CA maximum wait time must be set to 0. The reception timestamp of a NTP frame must be set to the time when the Sync-word has been detected.

The round-trip delay is computed as in Equation 2.1, where $t_0$ is the time of the request packet transmission, $t_1$ is the time of the request packet reception, $t_2$ is the time of the response packet transmission and $t_3$ is the time of the response packet reception. $t_3 - t_0$ is the time elapsed on the client side between the emission of the request packet and...
the reception of the response packet, while $t_2 - t_1$ is the time the server waited before sending the answer. The offset is given in Equation 2.2.

$$\delta = (t_3 - t_0) - (t_2 - t_1) \quad (2.1)$$

$$\theta = \frac{(t_1 - t_0) - (t_2 - t_3)}{2} \quad (2.2)$$
2.2 Protocol description

The application protocol is defined in XML to make the messages more human readable and easier to define the protocol messages. But XML is too big to use in an embedded application protocol, so it is compressed to be more usable in the embedded system. The XML messages are compressed using a method called Efficient XML Interchange (EXI) that converts XML into a binary event stream. The specification is defined by the W3C (www.w3.org) and the specification can be found at http://www.w3.org/TR/2011/REC-exi-20110310/. In this way an example XML document of 274 bytes can be compressed to be 4 bytes without loss of information. See XML 2.1 for an example of an XML message of 1253 Bytes which is compressed into an EXI message shown in XML 2.2 of 83 Bytes.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
  xsi:noNamespaceSchemaLocation="device_description.xsd">
  <device version="1">
    <device_description_report>
      <!-- Type -->
      <info number="1" type_id="1"/>
      <!-- Manufacturer -->
      <info hex="00112233445566778899AABB" type_id="2"/>
      <!-- Sgtin -->
      <info hex="AABBCCDDEEFF" type_id="3"/>
      <!-- Mac Address -->
      <info number="1" type_id="4"/>
      <!-- Hardware Version -->
      <info number="14" type_id="5"/>
      <!-- Bootloader Version -->
      <info number="1" type_id="6"/>
      <!-- Stack Version -->
      <info hex="00340080" type_id="7"/>
      <!-- Application Version -->
      <info number="500" type_id="8"/>
      <!-- Protocol -->
      <info number="10000" type_id="9"/>
      <!-- Product -->
      <info number="150" type_id="10"/>
      <!-- Included -->
      <info number="2" type_id="11"/>
      <!-- Name -->
      <info number="4" type_id="12"/>
      <!-- Radio Mode -->
      <info number="1" type_id="13"/>
      <!-- Wakeup Interval -->
      <info string="Device name" type_id="14"/>
    </device_description_report>
  </device>
</network>
```

XML 2.1: XML to EXI compression XML version

| 80 00 50 09 10 14 04 00 c0 01 12 23 34 45 56 67 78 89 9a ab | b8 08 00 6a ab bc cd de ef f8 0c 10 |
| 14 10 10 e4 14 10 14 18 00 40 03 40 08 08 1c 1f 40 34 20 19 | 04 e4 24 19 60 14 28 10 24 2c 10 44 |
| 30 10 14 34 20 44 46 57 66 96 36 52 06 e6 16 d6 50 71 68 |

XML 2.2: XML to EXI compression EXI version (values in hexadecimal notation)

2.2.1 Services

The applications listen on TCP and UDP ports, and depending on which port the message arrives at, it can be determined what kind of message it is. The ports are defined in Table 2.3. In a message there is a source and destination port. The destination port is the defined service port, and the source port can be a randomly selected port from 20128 to 20256 or one of the defined service ports. This port pair gives a connection between the sender and receiver. All answers will be sent back to the source port, so the sender knows that the incoming message is the answer. So when sending a value get message, then the destination port will be the value port and the source port will be a random port, because the message...
that has been sent is a value message. The reply will be send back with the ports flipped, so that the source port is the value port.

The port the devices listens on start from port 20000. The ports can be compressed so port 20000 is mapped to port 0 and port 20256 will be 256 as defined in Table 2.3. In the next section the inclusion and exclusion of devices will be described.

<table>
<thead>
<tr>
<th>Port number</th>
<th>Compressed port</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>20000</td>
<td>0</td>
<td>Value</td>
</tr>
<tr>
<td>20001</td>
<td>1</td>
<td>Device Description</td>
</tr>
<tr>
<td>20002</td>
<td>2</td>
<td>Public Key</td>
</tr>
<tr>
<td>20003</td>
<td>3</td>
<td>Network Management</td>
</tr>
<tr>
<td>20004</td>
<td>4</td>
<td>Value description</td>
</tr>
<tr>
<td>20005</td>
<td>5</td>
<td>Service description</td>
</tr>
<tr>
<td>20006</td>
<td>6</td>
<td>Memory information</td>
</tr>
<tr>
<td>20007</td>
<td>7</td>
<td>Partner information</td>
</tr>
<tr>
<td>20008</td>
<td>8</td>
<td>Action</td>
</tr>
<tr>
<td>20009</td>
<td>9</td>
<td>Calculation</td>
</tr>
<tr>
<td>20010</td>
<td>10</td>
<td>Timer</td>
</tr>
<tr>
<td>20011</td>
<td>11</td>
<td>Calendar</td>
</tr>
<tr>
<td>20012</td>
<td>12</td>
<td>State machine</td>
</tr>
<tr>
<td>20013</td>
<td>13</td>
<td>Firmware update</td>
</tr>
<tr>
<td>20014</td>
<td>14</td>
<td>Channel Scan</td>
</tr>
<tr>
<td>20015</td>
<td>15</td>
<td>Status</td>
</tr>
<tr>
<td>20016</td>
<td>16</td>
<td>Configuration</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
<td></td>
</tr>
</tbody>
</table>

Table 2.3: List of port numbers

2.2.2 Network management

When a device A starts up and it is not included, it must start by sending out a device description message, signaling that device A wants to be included into a network. When a Network Controller receives a device description message, the Network Controller can decide if device A should be included or not. If the Network Controller decides to include device A, it needs the RSA public key of device A. The Network Controller can receive this key from the back end, or if device A supports it, the key can be retrieved from device A.

When the Network Controller has the public key of device A, it will generate a new unique AES key (Controller key) that will be used to encrypt the Network Key. The controller key and the encrypted network key is then added to a message along with a crc of them both. The message format can be seen in Table 2.4.

This message in then encrypted with the public key of Device A, so that only Device A with its private key can decrypt the message and obtain the network key. See Figure 2.1 for an illustration of the inclusion of device A.

<table>
<thead>
<tr>
<th>Bytes</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0 - 15</td>
<td>A 16 byte AES controller key</td>
</tr>
<tr>
<td>16 - 31</td>
<td>A 16 byte AES network key AES encrypted with the controller key</td>
</tr>
<tr>
<td>32 - 33</td>
<td>A CRC 16 of the controller key and encrypted network key</td>
</tr>
</tbody>
</table>

Table 2.4: The format of the inclusion message

Alongside the Network Key, there is a number that tells the device how many bytes of its mac address it should use as its new address. The new address will for example be converted from 0x1234567890ab to 0x90ab if the address size is 2.

In the device when it receives the Network Key and the address size, it changes the value of its included property to 1. It also adds the compressed mac address to the list of address it listens on. Now device A is included and can communicate to the other devices in the network.

When device A needs to be excluded from the network, the Network Controller sends a message that sets the included property to 0. When device A receives this message it do factory reset, which clears all its values and parameters. Now the device is ready to be included again. See Figure 2.2 for an illustration of an exclusion of a device.

In the next sections there will be described how a device can be configured.
2.2.3 Device Configuration

Normally the behavior of a device is hard coded in the application of the device. Sometimes the device supports some configuration parameters to make small changes to the behavior. Often the behavior of the device does not meet future requirements of a user. If the complete behavior of a device could be configured, such a device would be more future proof. This goal can be obtained using state machines. These state machines can be constructed and tested on a PC and then be transferred to the device.

When a device is included into a network, only the default configuration, if any, is present on the device. If the device supports configuration then the device can be configured. When ever a device has been configured, the new configuration has to be saved before it becomes active. The configuration service is used to handle the saving and clearing the configuration on the device. The configuration service is described in Section 2.2.3.8. The rest of the configuration services is described in the following sections.

2.2.3.1 Virtual Values

A device can support virtual values, which is values that can be configured externally. These values can be used as a variable when doing complex calculations and state machines. If a virtual value is added with a type that the device do not support, it will send a status report back.

There is special configurations for a virtual value, that has a special meaning for the stack. The different configurations is described in Table 2.5. All of these special virtual values must be configured with `persistent` set to zero.
<table>
<thead>
<tr>
<th>Name</th>
<th>Type</th>
<th>Mode</th>
<th>Unit</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>Stop Watch</td>
<td>TIME</td>
<td>R/W</td>
<td>ms</td>
<td>The value can be set to zero to reset the stop watch, and when ever it is read, it returns the elapsed time.</td>
</tr>
<tr>
<td>Timezone Offset</td>
<td>TIMEZONE_OFFSET</td>
<td>R/W</td>
<td>s</td>
<td>The value is used to read and set the current timezone offset.</td>
</tr>
<tr>
<td>Year</td>
<td>YEAR</td>
<td>R</td>
<td>y</td>
<td>The value is used to get the current year.</td>
</tr>
<tr>
<td>Month</td>
<td>MONTH</td>
<td>R</td>
<td>mo</td>
<td>The value is used to get the current month.</td>
</tr>
<tr>
<td>Day of Month</td>
<td>DAY_OF_MONTH</td>
<td>R</td>
<td>d</td>
<td>The value is used to get the current day of month.</td>
</tr>
<tr>
<td>Weekday</td>
<td>WEEKDAY</td>
<td>R</td>
<td>d</td>
<td>The value is used to get the current weekday.</td>
</tr>
<tr>
<td>Hour of Day</td>
<td>HOUR</td>
<td>R</td>
<td>h</td>
<td>The value is used to get the current hour of the day.</td>
</tr>
<tr>
<td>Minute of Hour</td>
<td>MINUTE</td>
<td>R</td>
<td>min</td>
<td>The value is used to get the current minute of the hour.</td>
</tr>
</tbody>
</table>

Table 2.5: Special Virtual Values

### 2.2.3.2 Partner Service

The partner service is used to map an address to a simple id. In this way it is simple to reference other devices by their address by only using the id. This partner id can be used in all the other services to get easy method to address other devices. A partner consistent of an address and information about how to communicate with the device. Only devices which is configuration as partners can talk to the device. If a multicast address is configurated as a partner, then the device will listen and accept messages send to that this multicast address.

The partner service also support grouping multiple partners into a group with a id. If the device sends to a group, the message will be send to all the partners in the group. If one or more of the partners in the group is a multicast address, then the message is first send to all multicast address in the group.

### 2.2.3.3 Timer Service

The timer service is used for executing action with a delay. The delay is specified in milliseconds. Further more, a condition can be put on the timer, so that some condition needs to be met before the timer will execute the action.

The timer will also signal the state machine that the timer has triggered, so that the state machine can check it’s states.

### 2.2.3.4 Calendar Service

A device can also be configured to execute an action on a specific moment in time, using the calendar service. A calendar task can also be set to repeat by a interval. A filter value for which weekdays the calendar task should execute on is also supported.

The calendar will also signal the state machine that the calendar task has triggered, so that the state machine can check it’s states.

The calendar service will only execute when the device has been synchronized with ntp.

### 2.2.3.5 Action Service

An action can get, set a value on the device itself or on a other device. It can also send a report with the status of one its own values. An action can also start and stop a timer. When an action is set to be executed, it is added to a queue. If the previous action is a similar action to the same partner, then the action is combined with the previous action instead of being added. See Figure 2.3 for the flow chart of the action enqueue process.

### 2.2.3.6 Calculation Service

A calculation consistes of two sides, left and right, and a operator. A side can be a constant value, a reference to a value, local or from a partner, or an other calcualtion. It can also check if a timer or calculation has executed or if a state machine is in a spefic state. Calculations can be consists of other calculations to generate complex calculations.

### 2.2.3.7 State Machine Service

A state machine consists of states and transactions. A state has an ID so that it can be referenced. A transaction can have a calculation, action and the state the state machine should go to. If there is no calculation present on the transaction, then it is interpreted as always true. If there is no next state the transaction should take, then the state machine will just keep it’s current state.
The state machine can be triggered by 4 different events, timer event, calendar event, incoming value report and a local value update. When the state machine is triggered, it will loop though all the state machines and check the current state of each machine. It will execute the first transaction that are true from the current state of the machine. It will only execute each transaction once per event. In this way there will not be any run away loops due to invalid configuration of the state machine. When the statemachine is done handling the event, it will check if there was any state change in the state machines and then evaluate the calculations again, so that any calculations relaying on state machine states can be checked. It will then run though all the state machine again. See Figure 2.4 for the flow chart of state machine execution.

### 2.2.3.8 Configuration Service

The configuration service is used to persistent and enable the current configuration. When ever the device receives any new configuration, the configuration status is marked as started, and the device sends a Status message with the information about that the status has changed to started. When the configuration is started the state machine, timer and calendar is halted, until the configuration is switch backed to idle again. This can be done either by an incoming configuration mode setting the mode, or by a timeout that happens when there has not been any new configuration for sometime. If the timeout happen the device will do a rollback of any received configurations.
2.2.3.9 Status Service

The status service is used to report errors in the device. A status report contains the type of error and a code that describes what happen. There is also an optional data field that can be used to report additional information about the error.
The application can also send error messages using the status service. The application has its own type id, but the code are application specific.

2.2.3.10 Firmware Update Service

The firmware update service is used to updated the bootloader, stack and application of a device. Application can be updated without bootloader and stack, so that changes in application code, can easily be updated on a device, because application code is limited in size. The bootloader can only be updated together with stack and application, because there might be a change in the bootloader so that the stack is placed different in memory or something similar.
2.3 XML description

In this chapter the xml messages will be described. In every section there are descriptions of xml tags and examples of how to use them.

2.3.1 Network and device

The network and device tags are used to encapsulate all the messages. They both have an attribute labeled version. The version on the network tag, describes the version of the network and device tag. The version on the device tag describes which version is used of the message format.

The device tag also has a device ID for addressing multiple devices in a single device and go to sleep for defining the amount of time, before the device should go to sleep.

2.3.1.1 Tag description

The message tag for the network and device is described in Table 2.6.

<table>
<thead>
<tr>
<th>XML tag</th>
<th>Attribute</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>network</td>
<td></td>
<td>Yes</td>
<td>Root tag of each message</td>
</tr>
<tr>
<td>version</td>
<td></td>
<td>Yes</td>
<td>Version of the network</td>
</tr>
<tr>
<td>device</td>
<td></td>
<td>Yes</td>
<td>Current device</td>
</tr>
<tr>
<td>version</td>
<td></td>
<td>Yes</td>
<td>Version of the device message</td>
</tr>
<tr>
<td>device_id</td>
<td>No</td>
<td></td>
<td>ID of the device</td>
</tr>
<tr>
<td>goto_sleep</td>
<td>No</td>
<td></td>
<td>Time to wait before device goes to sleep</td>
</tr>
</tbody>
</table>

Table 2.6: Network and device tags

2.3.1.2 Example

The following section contains an XML example for using the network and device tags.

2.3.1.3 Network and device tags

See XML 2.3 for an example of how to encapsulate the messages.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<network version="1">
  <device version="1" device_id="1" goto_sleep="10000">
  </device>
</network>
```

XML 2.3: Encapsulation using network and device tags
2.3.2 Network management

The network management message is used to transmit the Network Controller key and the Network key, that are used for encrypting communication between the network controller and a specific device.

The network management message only consists of the network_include tag.

2.3.2.1 Tag description

The message tag for the network management is described in Table 2.7

<table>
<thead>
<tr>
<th>XML tag</th>
<th>Attribute</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>network_include</td>
<td>-</td>
<td>Yes</td>
<td>Is used by the network controller to include a device into the network.</td>
</tr>
</tbody>
</table>

Table 2.7: Network management tags

2.3.2.2 Inclusion data

The inclusion data is RSA encrypted with the public key of the device. The decrypted data consists of a controller key, the network key AES encrypted with the controller key, and a CRC of the complete message. The format is described in Table 2.4.

<table>
<thead>
<tr>
<th>Type</th>
<th>Data</th>
</tr>
</thead>
<tbody>
<tr>
<td>Controller key (hex)</td>
<td>0102030405060708090A0B0C0D0E0F00</td>
</tr>
<tr>
<td>Network key (hex)</td>
<td>001122334455667788999AABBCCDDEEFF</td>
</tr>
<tr>
<td>Encrypted Network key (hex)</td>
<td>E9855F797E25CDF8E9A4ECB83E6632B</td>
</tr>
<tr>
<td>CRC (hex)</td>
<td>5948</td>
</tr>
<tr>
<td>Complete message (hex)</td>
<td>0102030405060708090A0B0C0D0E0F00E9855F797E25CDF8E9A4ECB83E6632B5948</td>
</tr>
<tr>
<td>RSA public key (decimal)</td>
<td>65537</td>
</tr>
<tr>
<td>RSA common key (decimal)</td>
<td>544707154624265000829311500031052527271190401635629791278140655528</td>
</tr>
<tr>
<td></td>
<td>74241070617335139097638777900966260067194263263804785187364816327</td>
</tr>
<tr>
<td></td>
<td>27898011931008015117548459</td>
</tr>
<tr>
<td>RSA encrypted message (hex)</td>
<td>014CAE536800D9114DB19E3EC7EF857658BBA06F145B00B09ACE371F003BB9C2</td>
</tr>
<tr>
<td></td>
<td>FBF2C50076D38EB22FCAC92D3C245CB98C244EBAE48AD8B10CEE139209796F1</td>
</tr>
</tbody>
</table>

Table 2.8: The values used in the example
2.3.2.3 Examples

The following section contains a XML example for using the network_include messages.

**Network include**

To include a device a network_include message with a controller key and the network key must be sent from the network controller to the device. See XML 2.4 for an example of the include message and see Table 2.8 for the used keys.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
  xsi:noNamespaceSchemaLocation="networkmanagement.xsd">
  <device version="1">
    <network_include>
      014CAE53680D9114DB19EEEC7EF8576588BA06F145B0069ACE371F003BB9C2
      FBF2C500768DEEB22FCAC92DEC605CB998C244EBAE48AD8B10CEE139209796F1
    </network_include>
  </device>
</network>
```

XML 2.4: Network controller key
### 2.3.3 Public key

The public key message is used to get and report the devices public key.

#### 2.3.3.1 Tag description

The message tag for the public key is described in Table 2.9.

<table>
<thead>
<tr>
<th>XML tag</th>
<th>Attribute</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>publickey_get</td>
<td>-</td>
<td>Yes</td>
<td>Get the public key from the device</td>
</tr>
<tr>
<td>publickey_report</td>
<td>-</td>
<td>Yes</td>
<td>Report the public key</td>
</tr>
<tr>
<td>key_type</td>
<td>No</td>
<td></td>
<td>The type of the public key. See Table 2.10 for allowed values.</td>
</tr>
</tbody>
</table>

Table 2.9: Public key tags

#### 2.3.3.2 Key types

The supported public key types is listed in Table 2.10.

<table>
<thead>
<tr>
<th>Key Type Id</th>
<th>Key Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>RSA</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Table 2.10: Public key types
2.3.3.3 Examples

The following section contains XML examples for using the `publickey_get` and `publickey_report` messages.

**Get a public key**

In order to get a public key, a `publickey_get` message should be sent to the device. See XML 2.5 for an example of the get message.

<? xml version="1.0" encoding="UTF-8"?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
 xsi:noNamespaceSchemaLocation="public_key.xsd">
 <device version="1">
  <publickey_get/>
 </device>
</network>

XML 2.5: Get a public key

**Report a public key**

In order to report a public key, a `publickey_report` message should be sent from the device. See XML 2.6 for an example of the report message.

<? xml version="1.0" encoding="UTF-8"?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
 xsi:noNamespaceSchemaLocation="public_key.xsd">
 <device version="1">
  <publickey_report>4528289723859298309384092735</publickey_report>
 </device>
</network>

XML 2.6: Report a public key
2.3.4 Service description

The service description message is used to obtain and report descriptions of specific services.

The standalone tag service for each description allows the user to get a list of supported services for a specific device. A service is described by a type tag, which indicates whether the service allows e.g. memory information, device description or value description. Furthermore a version tag indicates the highest supported version of this service.

2.3.4.1 Tag description

The message tag for the service description is described in Table 2.11.

<table>
<thead>
<tr>
<th>XML tag</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>service_description_get</td>
<td>Yes</td>
<td>Get the service description</td>
</tr>
<tr>
<td>service_description_report</td>
<td>Yes</td>
<td>Report the service description</td>
</tr>
</tbody>
</table>

Table 2.11: Service description tags

The child tags for the service_description_report message are described in Table 2.12.

<table>
<thead>
<tr>
<th>XML tag</th>
<th>Attribute</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>service</td>
<td>-</td>
<td>Yes*</td>
<td>The services the device supports</td>
</tr>
<tr>
<td>service_id</td>
<td></td>
<td>Yes</td>
<td>The service ID. See Table 2.13</td>
</tr>
<tr>
<td>version</td>
<td></td>
<td>Yes</td>
<td>The maximum supported version of the service</td>
</tr>
</tbody>
</table>

Table 2.12: Service description message child tags

2.3.4.2 Service

The valid options for the service_id attributes in the service are listed in Table 2.13.

<table>
<thead>
<tr>
<th>Service Id</th>
<th>Device Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Public key</td>
</tr>
<tr>
<td>2</td>
<td>Memory description</td>
</tr>
<tr>
<td>3</td>
<td>Device description</td>
</tr>
<tr>
<td>4</td>
<td>Value description</td>
</tr>
<tr>
<td>5</td>
<td>Value</td>
</tr>
<tr>
<td>6</td>
<td>Partner information</td>
</tr>
<tr>
<td>7</td>
<td>Action</td>
</tr>
<tr>
<td>8</td>
<td>Calculation</td>
</tr>
<tr>
<td>9</td>
<td>Timer</td>
</tr>
<tr>
<td>10</td>
<td>Calendar</td>
</tr>
<tr>
<td>11</td>
<td>State machine</td>
</tr>
<tr>
<td>12</td>
<td>Firmware update</td>
</tr>
<tr>
<td>13</td>
<td>Channel Scan</td>
</tr>
<tr>
<td>14</td>
<td>Status</td>
</tr>
<tr>
<td>15</td>
<td>Configuration</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Table 2.13: Service Description Types
2.3.4.3 Example

The following section contains XML examples for using the `service_description_get` and `service_description_report` messages.

Get a service description

In order to get a service description, a `service_description_get` message should be sent to the device. See XML 2.7 for an example of the get message.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
xsi:noNamespaceSchemaLocation="service_description.xsd">
  <device version="1">
    <service_description_get/>
  </device>
</network>
```

XML 2.7: Get a service description

Report a service description

To report a service description, the `service_description_report` message should be sent from the device with its corresponding child tags as described earlier.

See XML 2.8 for an example of the report message.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
xsi:noNamespaceSchemaLocation="service_description.xsd">
  <device version="1">
    <service_description_report>
      <service service_id="1" version="1"/>
      <service service_id="3" version="1"/>
      <service service_id="4" version="1"/>
      <service service_id="2" version="1"/>
      <service service_id="6" version="1"/>
      <service service_id="5" version="1"/>
      <service service_id="13" version="1"/>
    </service_description_report>
  </device>
</network>
```

XML 2.8: Report a service description
2.3.5 Memory information

The memory information message is used to describe each possible number of timers, actions, etc. for a specific device, and is maintained using a report or get message.

2.3.5.1 Tag description

The message tag for the memory description is described in Table 2.14.

<table>
<thead>
<tr>
<th>XML tag</th>
<th>Attribute</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>memory_information_get</td>
<td>-</td>
<td>Yes</td>
<td>Get memory information of the different services of the device.</td>
</tr>
<tr>
<td>memory_information_report</td>
<td>-</td>
<td>Yes</td>
<td>Report memory information of the different services of the device.</td>
</tr>
</tbody>
</table>

Table 2.14: Memory description tags

The child tags for the memory_information_report message are described in Table 2.15.

<table>
<thead>
<tr>
<th>XML tag</th>
<th>Attribute</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>memory_information</td>
<td>-</td>
<td>Yes</td>
<td>The memory the device supports</td>
</tr>
<tr>
<td>memory_id</td>
<td>Yes</td>
<td></td>
<td>The service ID, See Table 2.16</td>
</tr>
<tr>
<td>count</td>
<td>Yes</td>
<td></td>
<td>The maximum number of the service the device supports</td>
</tr>
<tr>
<td>free_count</td>
<td>Yes</td>
<td></td>
<td>The number of free memory slots</td>
</tr>
</tbody>
</table>

Table 2.15: Memory_information_report child tags

2.3.5.2 Memory

The valid options for the memory_id attributes in the memory_information are listed in Table 2.16.

<table>
<thead>
<tr>
<th>Memory Id</th>
<th>The count type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Value</td>
</tr>
<tr>
<td>2</td>
<td>Partner Information</td>
</tr>
<tr>
<td>3</td>
<td>Action Items</td>
</tr>
<tr>
<td>4</td>
<td>Calculation</td>
</tr>
<tr>
<td>5</td>
<td>Timer</td>
</tr>
<tr>
<td>6</td>
<td>Calendar</td>
</tr>
<tr>
<td>7</td>
<td>Statemachine</td>
</tr>
<tr>
<td>8</td>
<td>Statemachine Transactions</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Table 2.16: Memory Types
2.3.5.3 Examples

The following section contains XML examples for using the `memory_information` messages.

**Get the memory information**

In order to get the memory information, a `memory_information_get` message has to be sent to the device. See XML 2.9 for an example of the get message.

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
  xsi:noNamespaceSchemaLocation="memory_information.xsd">
  <device version="1">
    <memory_information_get />
  </device>
</network>
```

XML 2.9: Get the memory information

**Report the memory information**

To report the memory information, the `memory_information_report` message has to be sent from the device with its corresponding child tag as described earlier. See XML 2.10 for an example of the report message.

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
  xsi:noNamespaceSchemaLocation="memory_information.xsd">
  <device version="1">
    <memory_information_report>
      <memory_information memory_id="1" count="5" free_count="3"/>
      <memory_information memory_id="2" count="10" free_count="8"/>
      <memory_information memory_id="3" count="5" free_count="1"/>
      <memory_information memory_id="4" count="6" free_count="4"/>
      <memory_information memory_id="5" count="8" free_count="6"/>
      <memory_information memory_id="6" count="8" free_count="5"/>
      <memory_information memory_id="7" count="9" free_count="0"/>
      <memory_information memory_id="8" count="3" free_count="1"/>
    </memory_information_report>
  </device>
</network>
```

XML 2.10: Report the memory information

**FiXme Note:** Add example for service get memory
2.3.6 Device description

The device description message is used to describe information about the device, e.g. type, manufacture, etc. and is maintained using a report, get and set message.

2.3.6.1 Tag description

The message tag for the device description is described in Table 2.17.

<table>
<thead>
<tr>
<th>XML tag</th>
<th>Attribute</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>device_description_get</td>
<td>-</td>
<td>Yes</td>
<td>This tag is used to request a device description from a device</td>
</tr>
<tr>
<td>device_description_report</td>
<td>-</td>
<td>Yes</td>
<td>This tag is used to report a device description from a device</td>
</tr>
<tr>
<td>device_description_set</td>
<td>-</td>
<td>Yes</td>
<td>This tag is used to set some device properties on a device</td>
</tr>
</tbody>
</table>

Table 2.17: Device Description tags

The child tags for the `device_description_report` and `device_description_set` message are described in Table 2.18.

<table>
<thead>
<tr>
<th>XML tag</th>
<th>Attribute</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>info</td>
<td>-</td>
<td>Yes</td>
<td>Set a device property</td>
</tr>
<tr>
<td>type_id</td>
<td>Yes</td>
<td></td>
<td>The id of the info. See Table 2.19</td>
</tr>
<tr>
<td>number</td>
<td>No</td>
<td></td>
<td>A number value</td>
</tr>
<tr>
<td>string</td>
<td>No</td>
<td></td>
<td>A string value</td>
</tr>
<tr>
<td>hex</td>
<td>No</td>
<td></td>
<td>A hex binary value</td>
</tr>
</tbody>
</table>

Table 2.18: Device Description child tags

<table>
<thead>
<tr>
<th>Id</th>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Type</td>
<td>Number</td>
<td>Manufacture type id</td>
</tr>
<tr>
<td>2</td>
<td>Manufacturer</td>
<td>Number</td>
<td>The manufacture of the device. See Table 2.22 for a list of manufactures</td>
</tr>
<tr>
<td>3</td>
<td>SGTIN</td>
<td>Hex</td>
<td>Serialized Global Trade Identification, which is a unique device number</td>
</tr>
<tr>
<td>4</td>
<td>Mac Address</td>
<td>Hex</td>
<td>The mac address of the device, which is a unique device address</td>
</tr>
<tr>
<td>5</td>
<td>Hardware Version</td>
<td>String</td>
<td>The version of the hardware</td>
</tr>
<tr>
<td>6</td>
<td>Bootloader Version</td>
<td>String</td>
<td>The version of the bootloader</td>
</tr>
<tr>
<td>7</td>
<td>Stack Version</td>
<td>String</td>
<td>The version of the stack</td>
</tr>
<tr>
<td>8</td>
<td>Application Version</td>
<td>String</td>
<td>The version of the application</td>
</tr>
<tr>
<td>9</td>
<td>Protocol</td>
<td>Number</td>
<td>The communication protocol the device uses. See Table 2.20 for a list of protocols</td>
</tr>
<tr>
<td>10</td>
<td>Product</td>
<td>Number</td>
<td>Manufacture product id</td>
</tr>
<tr>
<td>11</td>
<td>Included</td>
<td>Number</td>
<td>Boolean value that indicates if the device is included</td>
</tr>
<tr>
<td>12</td>
<td>Name</td>
<td>String</td>
<td>The name of the device</td>
</tr>
<tr>
<td>13</td>
<td>Radio Mode</td>
<td>Number</td>
<td>Device communication mode. See Table 2.21</td>
</tr>
<tr>
<td>14</td>
<td>Wakeup Interval</td>
<td>Number</td>
<td>The time in milliseconds that the device is offline between 2 consecutive RX active periods</td>
</tr>
<tr>
<td>15</td>
<td>Wakeup Offset</td>
<td>Number</td>
<td>The offset in milliseconds needed for the calculation of the RX on period</td>
</tr>
<tr>
<td>16</td>
<td>Wakeup Channel</td>
<td>Number</td>
<td>The radio channel the device listens on for wakeup frames</td>
</tr>
<tr>
<td>17</td>
<td>Channel Map</td>
<td>Hex</td>
<td>The current channel map. The channel map will always have 4 channels defined and 2 of them will be the synchrozations channels.</td>
</tr>
<tr>
<td>18</td>
<td>Channel Scan Time</td>
<td>Number</td>
<td>The time is takes to scan a single channel</td>
</tr>
<tr>
<td>19</td>
<td>IPv6 Address</td>
<td>Hex</td>
<td>IPv6 Address</td>
</tr>
</tbody>
</table>

Table 2.19: Device Description Types. Types in **bold** are settable
Protocol

The different protocols are described in Table 2.20.

<table>
<thead>
<tr>
<th>Protocol Id</th>
<th>Protocol</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Lemonbeat</td>
</tr>
<tr>
<td>2</td>
<td>Wi-Fi</td>
</tr>
<tr>
<td>3</td>
<td>Ethernet</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Table 2.20: Protocols

Radio Mode

The radio mode describes how a device communicates over the air. The default value for this is *Always Online*, so this info can be omitted in a `device_description_report` tag, if the uses the default radio mode. The different radio modes are defined in Table 2.21.

<table>
<thead>
<tr>
<th>Radio Mode Id</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Always Online</td>
</tr>
<tr>
<td>1</td>
<td>Wake on Radio</td>
</tr>
<tr>
<td>2</td>
<td>Wake on Event</td>
</tr>
<tr>
<td>3</td>
<td>TX Only</td>
</tr>
<tr>
<td>4</td>
<td>RX Only</td>
</tr>
</tbody>
</table>

Table 2.21: Radio Mode

Manufacturer

The different manufactures are defined in Table 2.22.

<table>
<thead>
<tr>
<th>Manufacture Id</th>
<th>Manufacture</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>RWE</td>
</tr>
<tr>
<td>2</td>
<td>Seluxit</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Table 2.22: Manufactures

Sgtin

Every device must have a unique identification number that is assigned to the device under the manufacturing process. The identification number of the device must remain constant throughout the devices entire lifetime. The identification number used is a 96 bits Serialized Global Trade Identification Number (SGTIN-96), which is a standard for making identification numbers.

The SGTIN-96 is specified in the *EPC global Tag Data Standards Version 1.4*\(^1\). The SGTIN-96 consists of a *Header* and five fields: *Filter Value*, *Partition*, *Company Prefix*, *Item Reference* and *Serial Number*, as shown in Table 2.23.

<table>
<thead>
<tr>
<th>SGTIN-96</th>
<th>Header</th>
<th>Filter value</th>
<th>Partition</th>
<th>Company prefix</th>
<th>Item Reference</th>
<th>Serial Number</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>00110000 (Binary)</td>
<td>See description below</td>
<td>See description below</td>
<td>999,999 - 999,999,999,999 (Max decimal range)</td>
<td>9,999,999-9 (Max decimal range)</td>
<td>274,877,906,943 (Max decimal value)</td>
</tr>
</tbody>
</table>

Table 2.23: SGTIN-96

The *Filter Value* is not part of the SGTIN-96 pure identity, but is additional data that is used for fast filtering and pre-selection of basic logistics types. The normative specifications for it are specified by GS-1.

\(^1\)http://www.gs1.org/gsmp/kc/epcglobal/tds/
Partition is an indication of where the subsequent Company Prefix and Item Reference numbers are divided. The SGTIN-96 and GS-1 GTIN structure matches each other in which the Company Prefix added to the Item Reference number (prefixed by the single Indicator Digit). This gives 13 digits in total; the Company Prefix can vary from 6 to 12 digits and the concatenation of single Indicator Digit and Item Reference from 7 to 1 digit(s).

The values of Partition and the corresponding sizes of the Company Prefix and Item Reference fields are defined in Table 2.24.

<table>
<thead>
<tr>
<th>Partition Value(P)</th>
<th>Bits (M)</th>
<th>Company Prefix Digits (L)</th>
<th>Indicator Digit and Item Reference Bits (N)</th>
<th>Digits</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>40</td>
<td>12</td>
<td>4</td>
<td>1</td>
</tr>
<tr>
<td>1</td>
<td>37</td>
<td>11</td>
<td>7</td>
<td>2</td>
</tr>
<tr>
<td>2</td>
<td>34</td>
<td>10</td>
<td>10</td>
<td>3</td>
</tr>
<tr>
<td>3</td>
<td>30</td>
<td>9</td>
<td>14</td>
<td>4</td>
</tr>
<tr>
<td>4</td>
<td>27</td>
<td>8</td>
<td>17</td>
<td>5</td>
</tr>
<tr>
<td>5</td>
<td>24</td>
<td>7</td>
<td>20</td>
<td>6</td>
</tr>
<tr>
<td>6</td>
<td>20</td>
<td>6</td>
<td>24</td>
<td>7</td>
</tr>
</tbody>
</table>

Table 2.24: SGTIN-96 Partition Values

Channel map

The channel map is a bit map of the selected channels the device uses for communication. The first bit (LSB) is channel 1 and the last bit (MSB) is channel 32. The channels are defined in Table ??.
2.3.6.2 Examples

The following section contains XML examples for using the `device_description_get` and `device_description_report` messages.

Get a device description

In order to get a device description, a `device_description_get` message has to be sent to the device. See XML 2.11 for an example of the get message.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
  xsi:noNamespaceSchemaLocation="device_description.xsd">
  <device version="1">
    <device_description_get/>
  </device>
</network>
```

XML 2.11: Get the device description

Report a device description

To report a device description, the `device_description_report` message has to be sent from the device with its corresponding attributes as described earlier. See XML 2.12 for an example of the report message.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
  xsi:noNamespaceSchemaLocation="device_description.xsd">
  <device version="1">
    <device_description_report>
      <!-- Type -->
      <info number="1" type_id="1"/>
      <!-- Manufacturer -->
      <info hex="00112233445566778899AABB" type_id="2"/>
      <!-- SgpIn -->
      <info hex="AABBCDEEFF" type_id="3"/>
      <!-- Mac Address -->
      <info number="1" type_id="4"/>
      <!-- Hardware Version -->
      <info number="14" type_id="5"/>
      <!-- Bootloader Version -->
      <info number="1" type_id="6"/>
      <!-- Stack Version -->
      <info hex="00340080" type_id="7"/>
      <!-- Application Version -->
      <info number="500" type_id="8"/>
      <!-- Protocol -->
      <info number="10000" type_id="9"/>
      <!-- Product -->
      <info number="150" type_id="10"/>
      <!-- Included -->
      <info number="2" type_id="11"/>
      <!-- Name -->
      <info number="4" type_id="12"/>
      <!-- Radio Mode -->
      <info number="1" type_id="13"/>
      <!-- Wakeup Interval -->
      <info string="Device name" type_id="14"/>
    </device_description_report>
  </device>
</network>
```

XML 2.12: Report of the device description
Set device description properties

In order to set a device description property, the `device_description_set` message has to be sent to the device with the corresponding attributes as described earlier. See XML 2.13 for an example of the set message.

```
<?xml version="1.0" encoding="UTF-8"?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
  xsi:noNamespaceSchemaLocation="device_description.xsd">
  <device version="1">
    <device_description_set>
      <!-- Included -->
      <info type_id="11" number="1"/>
      <!-- Wakeup Interval -->
      <info type_id="14" number="10000"/>
      <!-- wakeup_offset -->
      <info type_id="15" number="150"/>
      <!-- wakeup_channel -->
      <info type_id="16" number="2"/>
      <!-- name -->
      <info type_id="12" string="New value name"/>
    </device_description_set>
  </device>
</network>
```

XML 2.13: Set the device description property
2.3.7 Value Description

The value description message is used to describe each possible value for a specific device, e.g. type, mode etc. and is accessed using a report or get message.

A number value has min, max and step. If a value is out side of the min-max range, then it is capped to be with in this range. If the value do not conform the the step size, then the value is rounded down.

2.3.7.1 Tag description

The message tag for the value description is described in Table 2.25.

<table>
<thead>
<tr>
<th>XML tag</th>
<th>Attribute</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>value_description_get</td>
<td>-</td>
<td>Yes</td>
<td>Get the description value</td>
</tr>
<tr>
<td>value_description_id</td>
<td>No</td>
<td></td>
<td>ID of the description value</td>
</tr>
<tr>
<td>value_description_report</td>
<td>-</td>
<td>Yes</td>
<td>Report the description value</td>
</tr>
<tr>
<td>value_description_add</td>
<td>-</td>
<td>Yes</td>
<td>Creates a virtual value</td>
</tr>
<tr>
<td>value_description_delete</td>
<td>-</td>
<td>Yes</td>
<td>Delete a virtual value</td>
</tr>
<tr>
<td>value_description_get_memory</td>
<td>-</td>
<td>Yes</td>
<td>Request a memory report for this service</td>
</tr>
<tr>
<td>value_description_report_memory</td>
<td>-</td>
<td>Yes</td>
<td>Report the memory information for this service</td>
</tr>
<tr>
<td>count</td>
<td>Yes</td>
<td></td>
<td>The maximum number of value descriptions the device supports</td>
</tr>
<tr>
<td>free_count</td>
<td>Yes</td>
<td></td>
<td>The used number of value descriptions</td>
</tr>
</tbody>
</table>

Table 2.25: Value Description tags

The child tags for the value_description_report and value_description_add message are described in Table 2.26.

<table>
<thead>
<tr>
<th>XML tag</th>
<th>Attribute</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>value_description</td>
<td>-</td>
<td>Yes</td>
<td>The value description</td>
</tr>
<tr>
<td>value_id</td>
<td>Yes</td>
<td></td>
<td>ID of the description value</td>
</tr>
<tr>
<td>type_id</td>
<td>Yes</td>
<td></td>
<td>Specifies type of value, see Table 2.29</td>
</tr>
<tr>
<td>mode</td>
<td>Yes</td>
<td></td>
<td>Possible interaction with value:</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1. Read Only (Only read from the device)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>2. Read/Write (Read and write to/from the device)</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>3. Write Only (Write to the device)</td>
</tr>
<tr>
<td>persistent</td>
<td>Yes</td>
<td></td>
<td>Specifies if the value is persistent doing a power cycling</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>0. Not persistent</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>1. Is persistent</td>
</tr>
<tr>
<td>name</td>
<td>No</td>
<td></td>
<td>The name of the device</td>
</tr>
<tr>
<td>min_log_interval</td>
<td>No</td>
<td></td>
<td>Minimum time between log values in seconds</td>
</tr>
<tr>
<td>max_log_values</td>
<td>No</td>
<td></td>
<td>Maximum numbers of stored values for this device</td>
</tr>
<tr>
<td>virtual</td>
<td>No</td>
<td></td>
<td>Specifies if the value is a virtual value</td>
</tr>
</tbody>
</table>

Table 2.26: Value_description_report message child tags
The child tags for the `value_description` in Table 2.27. The sections that follow will describe all child tags in more detail.

<table>
<thead>
<tr>
<th>XML tag</th>
<th>Attribute</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>number_format</td>
<td>-</td>
<td>Yes*</td>
<td>When the format of the value is a number</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>*If not defined, then another format tag must be defined</td>
</tr>
<tr>
<td>unit</td>
<td>Yes</td>
<td></td>
<td>Specifies which unit the value holds</td>
</tr>
<tr>
<td>min</td>
<td>Yes</td>
<td></td>
<td>Minimum value</td>
</tr>
<tr>
<td>max</td>
<td>Yes</td>
<td></td>
<td>Maximum value</td>
</tr>
<tr>
<td>step</td>
<td>Yes</td>
<td></td>
<td>The step-size by which the value can</td>
</tr>
</tbody>
</table>

| string_format   | -         | Yes*     | When the format of the value is a string                                     |
|                 |           |          | *If not defined, then another format tag must be defined                     |
| max_length      | Yes       |          | Maximum length of the string                                                 |

| hexBinary_format| -         | Yes*     | When the format of the value is hex binary                                   |
|                |           |          | *If not defined, then another format tag must be defined                     |
| max_length     | Yes       |          | The maximum length of the hex binary data                                   |

Table 2.27: Value description child tags

The child tags for the `string_format` are described in Table 2.28.

<table>
<thead>
<tr>
<th>XML tag</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>valid_value</td>
<td>No</td>
<td>The values that are valid for the string value</td>
</tr>
</tbody>
</table>

Table 2.28: Value description String format child tags

**Type**

The valid options for the `type_id` attribute in the value description are listed in Table 2.29. The `type_id` attribute is a required and should be set to a valid option.

**Unit**

The `unit` attribute is in the International System of Units (SI) format. The `unit` attribute is a required and must be set to a valid value.
<table>
<thead>
<tr>
<th>Type Id</th>
<th>Name</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Temperature</td>
</tr>
<tr>
<td>2</td>
<td>Luminance</td>
</tr>
<tr>
<td>3</td>
<td>Power</td>
</tr>
<tr>
<td>4</td>
<td>Electricity</td>
</tr>
<tr>
<td>5</td>
<td>Humidity</td>
</tr>
<tr>
<td>6</td>
<td>Velocity</td>
</tr>
<tr>
<td>7</td>
<td>Direction</td>
</tr>
<tr>
<td>8</td>
<td>Atmospheric</td>
</tr>
<tr>
<td>9</td>
<td>Barometric</td>
</tr>
<tr>
<td>10</td>
<td>Solar Radiation</td>
</tr>
<tr>
<td>11</td>
<td>Dew Point</td>
</tr>
<tr>
<td>12</td>
<td>Rain Rate</td>
</tr>
<tr>
<td>13</td>
<td>Tide level</td>
</tr>
<tr>
<td>14</td>
<td>ON/OFF</td>
</tr>
<tr>
<td>15</td>
<td>Awake State</td>
</tr>
<tr>
<td>16</td>
<td>Event</td>
</tr>
<tr>
<td>17</td>
<td>General Purpose</td>
</tr>
<tr>
<td>18</td>
<td>Counter</td>
</tr>
<tr>
<td>19</td>
<td>Energy</td>
</tr>
<tr>
<td>20</td>
<td>Level</td>
</tr>
<tr>
<td>21</td>
<td>CO2</td>
</tr>
<tr>
<td>22</td>
<td>Air Flow</td>
</tr>
<tr>
<td>23</td>
<td>Tank Capacity</td>
</tr>
<tr>
<td>24</td>
<td>Distance</td>
</tr>
<tr>
<td>25</td>
<td>Climate Control</td>
</tr>
<tr>
<td>26</td>
<td>Program</td>
</tr>
<tr>
<td>27</td>
<td>Fan Speed</td>
</tr>
<tr>
<td>28</td>
<td>Error Code</td>
</tr>
<tr>
<td>29</td>
<td>Operation Mode</td>
</tr>
<tr>
<td>30</td>
<td>Louvre</td>
</tr>
<tr>
<td>31</td>
<td>Mode</td>
</tr>
<tr>
<td>32</td>
<td>Time</td>
</tr>
<tr>
<td>33</td>
<td>Duty Cycle</td>
</tr>
<tr>
<td>34</td>
<td>Voltage</td>
</tr>
<tr>
<td>35</td>
<td>Current</td>
</tr>
<tr>
<td>36</td>
<td>Frequency</td>
</tr>
<tr>
<td>37</td>
<td>Battery</td>
</tr>
<tr>
<td>38</td>
<td>Timezone Offset</td>
</tr>
<tr>
<td>39</td>
<td>Year</td>
</tr>
<tr>
<td>40</td>
<td>Month</td>
</tr>
<tr>
<td>41</td>
<td>Day Of Month</td>
</tr>
<tr>
<td>42</td>
<td>Weekday</td>
</tr>
<tr>
<td>43</td>
<td>Hour</td>
</tr>
<tr>
<td>44</td>
<td>Minute</td>
</tr>
</tbody>
</table>

Table 2.29: Value description types
2.3.7.2 Examples

The following section contains XML examples for using the `value_description_get`, `value_description_report`, `value_description_add`, and `value_description_delete` messages.

Get a specific value description

In order to get a specific value description, a `value_description_get` message has to be sent to the device with a value description ID. See XML 2.14 for an example of the get message.

```
<?xml version="1.0" encoding="UTF-8"?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
    xsi:noNamespaceSchemaLocation="value_description.xsd">
    <device version="1">
        <value_description_get value_description_id="1"/>
    </device>
</network>
```

XML 2.14: Get the value description with value description ID 1

Get all value descriptions

In order to get all value description, a `value_description_get` message has to be sent to the device without a value description ID. See XML 2.15 for an example of the get all message.

```
<?xml version="1.0" encoding="UTF-8"?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
    xsi:noNamespaceSchemaLocation="value_description.xsd">
    <device version="1">
        <value_description_get/>
    </device>
</network>
```

XML 2.15: Get all value descriptions

Report a specific value description for number format

To report a specific value description for number format, the `value_description_report` message has to be sent from the device with its child tag `value_description` and its child tag `number_format`. See XML 2.16 for an example of the report for number format message.

```
<?xml version="1.0" encoding="UTF-8"?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
    xsi:noNamespaceSchemaLocation="value_description.xsd">
    <device version="1">
        <value_description_report>
            <value_description value_id="1" type_id="14" mode="2" persistent="0" name="light"
                min_log_interval="300" max_log_values="20">
                <number_format unit="W" min="0" max="1" step="1"/>
            </value_description>
        </value_description_report>
    </device>
</network>
```

XML 2.16: Report of the value description for number format
Report a specific value description for string format

To report a specific value description for string format, the value_description_report message has to be sent from the device with its child tag "value_description" and its child tag string_format. See XML 2.17 for an example of the report string format message.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
    xsi:noNamespaceSchemaLocation="value_description.xsd">
    <device version="1">
        <value_description_report>
            <value_description value_id="2" type_id="26" mode="1" persistent="1" name="washing program">
                <string_format max_length="20">
                    <valid_value>wool</valid_value>
                    <valid_value>cotton</valid_value>
                </string_format>
            </value_description>
        </value_description_report>
    </device>
</network>
```

XML 2.17: Report of the value description for string format

Add a virtual value description

To add a virtual value description to a device, the value_description_add message has to be sent to the device. See XML 2.18 for an example of the add message.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
    xsi:noNamespaceSchemaLocation="value_description.xsd">
    <device version="1">
        <value_description_add>
            <value_description value_id="5" type_id="14" mode="2" persistent="0" name="Virtual Value" virtual="1">
                <number_format unit="W" min="0" max="100" step="1"/>
            </value_description>
        </value_description_add>
    </device>
</network>
```

XML 2.18: Add virtual value description

Delete a virtual value description

To delete a virtual value description from a device, the value_description_delete message has to be sent to the device. See XML 2.20 for an example of the delete message.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
    xsi:noNamespaceSchemaLocation="value_description.xsd">
    <device version="1">
        <value_description_delete value_description_id="5"/>
    </device>
</network>
```

XML 2.19: Delete virtual value description

====== »»» other
<? xml version="1.0" encoding="UTF-8" ?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"  
xsi:noNamespaceSchemaLocation="value_description.xsd">  
<device version="1">  
<value_description_delete value_description_id="5" />  
</device>  
</network>

XML 2.20: Delete virtual value description
2.3.8 Value

The value message is used to get, set, report and logging a value and is maintained using a report, get, set and get log message.

If the device is not synchronized with NTP, then the reported timestamp will be zero.

2.3.8.1 Tag description

The message tag for the value is described in Table 2.30.

<table>
<thead>
<tr>
<th>XML tag</th>
<th>Attribute</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>value_get</td>
<td>-</td>
<td>Yes</td>
<td>The value_get tag is used to get all the values. See examples below.</td>
</tr>
<tr>
<td>value_id</td>
<td>No</td>
<td></td>
<td>ID of the value. To get all the values omit value_id</td>
</tr>
<tr>
<td>value_report</td>
<td>-</td>
<td>Yes</td>
<td>The value_report tag is used to report all the values. See examples below.</td>
</tr>
<tr>
<td>value_id</td>
<td>Yes</td>
<td></td>
<td>ID of the value</td>
</tr>
<tr>
<td>number</td>
<td>Yes*</td>
<td></td>
<td>The number value</td>
</tr>
<tr>
<td>string</td>
<td>Yes*</td>
<td></td>
<td>The string value</td>
</tr>
<tr>
<td>hex</td>
<td>Yes*</td>
<td></td>
<td>The hex value</td>
</tr>
<tr>
<td>timestamp</td>
<td>Yes</td>
<td></td>
<td>Timestamp of when the current value was send</td>
</tr>
<tr>
<td>value_set</td>
<td>-</td>
<td>Yes</td>
<td>The value_set tag is used to set multiple the values. See examples below.</td>
</tr>
<tr>
<td>value_id</td>
<td>Yes</td>
<td></td>
<td>ID of the value</td>
</tr>
<tr>
<td>number</td>
<td>Yes*</td>
<td></td>
<td>The number value</td>
</tr>
<tr>
<td>string</td>
<td>Yes*</td>
<td></td>
<td>The string value</td>
</tr>
<tr>
<td>hex</td>
<td>Yes*</td>
<td></td>
<td>The hex value</td>
</tr>
<tr>
<td>timestamp</td>
<td>Yes</td>
<td></td>
<td>Timestamp of when the current value was send</td>
</tr>
<tr>
<td>value_get_log</td>
<td>-</td>
<td>Yes</td>
<td>The value_get_log tag is used to get all the logged values. See examples below.</td>
</tr>
<tr>
<td>Value_id</td>
<td>No</td>
<td></td>
<td>ID of the value</td>
</tr>
<tr>
<td>start_time</td>
<td>No</td>
<td></td>
<td>To request all the logged values omit value_id</td>
</tr>
<tr>
<td>log_count</td>
<td>No</td>
<td></td>
<td>To request up to log count number of logged events.</td>
</tr>
</tbody>
</table>

Table 2.30: Value tags
2.3.8.2 Examples

This flowering section consists of examples in xml for how to use the 'value_get', 'value_report', 'value_report', 'value_set', 'value_set' and 'value_get_log' messages.

Get a specific value

In order to get a specific value, a 'value_get' message has to be sent to the device with a value ID. See XML 2.21 for an example of the 'get' message.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
  xsi:noNamespaceSchemaLocation="value.xsd">
  <device version="1">
    <value_get value_id="1"/>
  </device>
</network>
```

XML 2.21: Get the value with ID 1

Get all values

In order to get all values, a 'value_get' message has to be sent to the device without a value ID. See XML 2.22 for an example of the 'get all' message.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
  xsi:noNamespaceSchemaLocation="value.xsd">
  <device version="1">
    <value_get/>
  </device>
</network>
```

XML 2.22: Get all values

Report a specific number value

To report a specific number value, the 'value_report' message has to be sent from the the device with a value ID and a number. See XML 2.23 for an example of the 'report' message.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
  xsi:noNamespaceSchemaLocation="value.xsd">
  <device version="1">
    <value_report value_id="1" timestamp="0" number="55"/>
  </device>
</network>
```

XML 2.23: Report of the value with ID 1 has the value 55
Report a specific string value

To report a specific string value, the "value_report" message has to be sent from the the device with a value ID and a string. See XML 2.24 for an example of the 'report' message.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
    xsi:noNamespaceSchemaLocation="value.xsd">
  <device version="1">
    <value_report value_id="2" string="ON" timestamp="0"/>
  </device>
</network>
```

XML 2.24: Report of the value with ID 2 has the value ON

Report all values

To report all values, the "value_report" message has to be sent from the device with a value ID and a number or a string. See XML 2.25 for an example of the 'report' message.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
    xsi:noNamespaceSchemaLocation="value.xsd">
  <device version="1">
    <value_report value_id="1" number="55" timestamp="0"/>
    <value_report value_id="2" string="ON" timestamp="0"/>
  </device>
</network>
```

XML 2.25: Report of all the values

Set a specific number value

In order to set a specific number value, a "value_set" message has to be sent with a value ID and a number. See XML 2.26 for an example of the 'set' message.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
    xsi:noNamespaceSchemaLocation="value.xsd">
  <device version="1">
    <value_set value_id="1" timestamp="0" number="55"/>
  </device>
</network>
```

XML 2.26: Set the value with ID 1 to 55

Set a specific string value

In order to set a specific string value, a "value_set" message has to be sent with a value ID and a string. See XML 2.27 for an example of the 'set' message.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
    xsi:noNamespaceSchemaLocation="value.xsd">
  <device version="1">
    <value_set value_id="2" timestamp="0" string="ON"/>
  </device>
</network>
```

XML 2.27: Set the value with ID 2 to ON
Set multiple values

In order to set multiple values, a "value_set" with a value ID and a number or a string. See XML 2.28 for an example of the "set" message.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
    xsi:noNamespaceSchemaLocation="value.xsd">
    <device version="1">
        <value_set value_id="1" timestamp="0" number="55"/>
        <value_set value_id="2" timestamp="0" string="ON"/>
    </device>
</network>
```

XML 2.28: Set the value with ID 1 to 55 and the value with ID 2 to ON

Get Log values for a specific value

In order to get a log for a specific value, a "value_get_log" message has to be sent to the device with a value ID, a start time and a log count. See XML 2.29 for an example of the "get log" message.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
    xsi:noNamespaceSchemaLocation="value.xsd">
    <device version="1">
        <value_get_log value_id="1" start_time="1314980580" log_count="2"/>
    </device>
</network>
```

XML 2.29: Get the log for the value with ID 1

Report log values for a specific value

To report all logs for a specific value, the "value_report" message has to be sent from the device with a value ID, a timestamp and a number or a string. See XML 2.30 for an example of the "report" message.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
    xsi:noNamespaceSchemaLocation="value.xsd">
    <device version="1">
        <value_report value_id="1" timestamp="1314980580" number="55"/>
        <value_report value_id="2" timestamp="1314980585" string="ON"/>
    </device>
</network>
```

XML 2.30: Report all the log values for the value with ID 1 and 2
2.3.9 Partner information

The partner information message is used to obtain detailed information about existing partners and groups as well as creating additional information or deleting information from these.

The partner information message is maintained by get, set, report and delete tags. Each partner information is labeled with a "partner id" for identification purposes, and contains either a partner or group.

A Partner is described by an address, encryption key and radio mode, where as a group is described as list of partners for the purpose of sending message to all partners in the group at once, through a multicast address.

2.3.9.1 Tag description

The message tag for the partner information is described in Table 2.31.

<table>
<thead>
<tr>
<th>XML tag</th>
<th>Attribute</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>partner_information_get</td>
<td>-</td>
<td>Yes</td>
<td>The partner_information_get tag is used to get partners. See examples below.</td>
</tr>
<tr>
<td>partner_id</td>
<td>No</td>
<td></td>
<td>ID of the partner. To get all the partners omit partner_id.</td>
</tr>
<tr>
<td>partner_information_report</td>
<td>-</td>
<td>Yes</td>
<td>The partner_information_report tag is used to report partners. See examples below.</td>
</tr>
<tr>
<td>partner_information_set</td>
<td>-</td>
<td>Yes</td>
<td>The partner_information_set tag is used to set partners. See examples below.</td>
</tr>
<tr>
<td>partner_information_delete</td>
<td>-</td>
<td>Yes</td>
<td>The partner_information_delete tag is used to delete partners. See examples below.</td>
</tr>
<tr>
<td>partner_id</td>
<td>No</td>
<td></td>
<td>ID of the partner. To delete all the partners omit partner_id.</td>
</tr>
<tr>
<td>partner_information_get_memory</td>
<td>-</td>
<td>Yes</td>
<td>This tag is used to request the partner information memory information from a device</td>
</tr>
<tr>
<td>partner_information_report_memory</td>
<td>-</td>
<td>Yes</td>
<td>This tag is used to report the partner information memory information from a device</td>
</tr>
<tr>
<td>count</td>
<td>Yes</td>
<td></td>
<td>The maximum number of partners the device supports</td>
</tr>
<tr>
<td>free_count</td>
<td>Yes</td>
<td></td>
<td>The used number of partners</td>
</tr>
</tbody>
</table>

Table 2.31: Partner tags

The child tags for the partner_information_report and partner_information_set message are described in Table 2.32.

<table>
<thead>
<tr>
<th>XML tag</th>
<th>Attribute</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>partner</td>
<td>-</td>
<td>Yes*</td>
<td>Describes a partners address, encryption key and radio mode. See examples below.</td>
</tr>
<tr>
<td>partner_id</td>
<td>Yes</td>
<td></td>
<td>ID of the partner</td>
</tr>
<tr>
<td>group</td>
<td>-</td>
<td>Yes*</td>
<td>Is used to group partners to a multicast address</td>
</tr>
<tr>
<td>partner_id</td>
<td>Yes</td>
<td></td>
<td>ID of the partner</td>
</tr>
</tbody>
</table>

Table 2.32: Partner information child tags
The child tags for partner are described in Table 2.33.

<table>
<thead>
<tr>
<th>XML tag</th>
<th>Attribute</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>info</td>
<td>–</td>
<td>Yes</td>
<td>Set a device property</td>
</tr>
<tr>
<td>type_id</td>
<td>Yes</td>
<td></td>
<td>The id of the info. See Table 2.34</td>
</tr>
<tr>
<td>number</td>
<td>No</td>
<td></td>
<td>A number value</td>
</tr>
<tr>
<td>string</td>
<td>No</td>
<td></td>
<td>A string value</td>
</tr>
<tr>
<td>hex</td>
<td>No</td>
<td></td>
<td>A hex binary value</td>
</tr>
</tbody>
</table>

Table 2.33: Partner Information partner child tags

The valid values for **type_id** in partner information is described in Table 2.34.

<table>
<thead>
<tr>
<th>Id</th>
<th>Name</th>
<th>Type</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>13</td>
<td>Radio Mode</td>
<td>Number</td>
<td>The partners radio mode. See Table 2.21</td>
</tr>
<tr>
<td>14</td>
<td>Wakeup Interval</td>
<td>Number</td>
<td>The offset in milliseconds of when the partner starts to wakeup</td>
</tr>
<tr>
<td>15</td>
<td>Wakeup Offset</td>
<td>Number</td>
<td>How often in milliseconds the partner will wakeup</td>
</tr>
<tr>
<td>16</td>
<td>Wakeup Channel</td>
<td>Number</td>
<td>The radio channel the partner listens on to see if it has to wakeup</td>
</tr>
<tr>
<td>17</td>
<td>Channel Map</td>
<td>Hex</td>
<td>The current channel map</td>
</tr>
<tr>
<td>18</td>
<td>Channel Scan Time</td>
<td>Number</td>
<td>The time it takes to scan a single channel</td>
</tr>
<tr>
<td>19</td>
<td>IPv6 Address</td>
<td>Hex</td>
<td>The full IPv6 address of the partner</td>
</tr>
<tr>
<td>20</td>
<td>Wakeup Now</td>
<td>Number</td>
<td>The time in milliseconds the partner should be a wake</td>
</tr>
</tbody>
</table>

Table 2.34: Device Description Types. Types in **bold** are settable

The child tags for the group are described in Table 2.35.

<table>
<thead>
<tr>
<th>XML tag</th>
<th>Attribute</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>partner</td>
<td>–</td>
<td>Yes</td>
<td>The partner in the group</td>
</tr>
<tr>
<td>partner_id</td>
<td>Yes</td>
<td></td>
<td>ID of the partner</td>
</tr>
</tbody>
</table>

Table 2.35: Group child tags
2.3.9.2 Examples

The following section contains XML examples for using the "partner_information_get", "partner_information_report", "partner_information_set" and "partner_information_delete" messages.

Get a specific partner

In order to get a specific partner, a "partner_information_get" message has to be sent to the device with a partner ID. See XML 2.31 for an example of the 'get' message.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
    xsi:noNamespaceSchemaLocation="partner_information.xsd">
  <device version="1">
    <partner_information_get partner_id="1"/>
  </device>
</network>
```

XML 2.31: Get the partner with the ID 1

Get all partners

In order to get all partners, a "partner_information_get" message has to be sent to the device without a partner ID. See XML 2.32 for an example of the 'get all' message.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
    xsi:noNamespaceSchemaLocation="partner_information.xsd">
  <device version="1">
    <partner_information_get/>
  </device>
</network>
```

XML 2.32: Get all partners

Report a specific partner

To report a specific partner, the "partner_information_report" message has to be sent from the device with a partner ID on its corresponding child tags as described earlier. See XML 2.33 for an example of the 'report' message.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
    xsi:noNamespaceSchemaLocation="partner_information.xsd">
  <device version="1">
    <partner_information_report>
      <partner partner_id="1">
        <!-- Radio Mode -->
        <info type_id="13" number="1"/>
        <!-- Wakeup Interval -->
        <info type_id="14" number="10000"/>
        <!-- wakeup_offset -->
        <info type_id="15" number="150"/>
        <!-- wakeup_channel -->
        <info type_id="16" number="2"/>
        <!-- IPv6 Address -->
        <info type_id="19" hex="fc0000000000000000100000000012"/>
      </partner>
    </partner_information_report>
  </device>
</network>
```

XML 2.33: Report for the partner with ID 1
Report all partners

To report all partners, the 'partner_information_report' message has to be sent from the device with a partner ID and its corresponding child tags as described earlier. See XML 2.34 for an example of the 'report all' message.

```
<?xml version="1.0" encoding="UTF-8"?>
<network xmlns="http://www.w3.org/2001/XMLSchema-instance" version="1"
xsi:noNamespaceSchemaLocation="partner_information.xsd">
  <device version="1">
    <partner_information_report>
      <partner partner_id="1">
        <!-- Radio Mode -->
        <info type_id="13" number="1" />
        <!-- Wakeup Interval -->
        <info type_id="14" number="10000" />
        <!-- wakeup_offset -->
        <info type_id="15" number="150" />
        <!-- wakeup_channel -->
        <info type_id="16" number="2" />
        <!-- IPv6 Address -->
        <info type_id="19" hex="fc000000000000010000000012" />
      </partner>
    </partner_information_report>
  </device>
</network>
```

XML 2.34: Report all partners

Set a specific partner

In order to set a specific partner, a 'partner_information_set' message has to be sent to the device with a partner ID on its corresponding child tags as described earlier. See XML 2.35 for an example of the 'set' message.

```
<?xml version="1.0" encoding="UTF-8"?>
<network xmlns="http://www.w3.org/2001/XMLSchema-instance" version="1"
xsi:noNamespaceSchemaLocation="partner_information.xsd">
  <device version="1">
    <partner_information_set>
      <partner partner_id="1">
        <!-- Radio Mode -->
        <info type_id="13" number="1" />
        <!-- Wakeup Interval -->
        <info type_id="14" number="10000" />
        <!-- wakeup_offset -->
        <info type_id="15" number="150" />
        <!-- wakeup_channel -->
        <info type_id="16" number="2" />
        <!-- IPv6 Address -->
        <info type_id="19" hex="fc000000000000010000000012" />
      </partner>
    </partner_information_set>
  </device>
</network>
```

XML 2.35: Set the partner with ID 1
Set multiple partners

In order to set multiple partners, a 'partner_information_set' message has to be sent to the device with a partner and its corresponding child tags as described earlier. See XML 2.36 for an example of the 'set multiple' message.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
    xsi:noNamespaceSchemaLocation="partner_information.xsd">
  <device version="1">
    <partner_information_set>
      <partner partner_id="1">
        <!-- Radio Mode -->
        <info type_id="13" number="1" />
        <!-- Wakeup Interval -->
        <info type_id="14" number="10000" />
        <!-- wakeup_offset -->
        <info type_id="15" number="150" />
        <!-- wakeup_channel -->
        <info type_id="16" number="2" />
        <!-- IPv6 Address -->
        <info type_id="19" hex="fc000000000000000100000000000012"/>
      </partner>
      <partner partner_id="2">
        <!-- IPv6 Address -->
        <info type_id="19" hex="fc000000000000000100000000000017"/>
      </partner>
    </group>
  </partner_information_set>
</device>
</network>
```

XML 2.36: Set multiple partners

Delete a specific partner

In order to delete a specific partner, a 'partner_information_delete' message has to be sent to the device with a partner ID. See XML 2.37 for an example of the 'delete' message.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
    xsi:noNamespaceSchemaLocation="partner_information.xsd">
  <device version="1">
    <partner_information_delete partner_id="1"/>
  </device>
</network>
```

XML 2.37: Delete the partners with ID 1

Delete all partners

In order to delete all partners, a 'partner_information_delete' message has to be sent to the device without a partner ID. See XML 2.38 for an example of the 'delete all' message.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
    xsi:noNamespaceSchemaLocation="partner_information.xsd">
  <device version="1">
    <partner_information_delete/>
  </device>
</network>
```

XML 2.38: Delete all partners
2.3.10  Action

The action message is used to obtain a list of all current actions on a specific device. The action message is maintained by get, report, set and delete tags. Each action specifies a value which can be both get and set by their corresponding child tags. Furthermore it is possible to group different actions together, as well as setting utilizing a timer to determine the run-time of a specific action.

2.3.10.1  Tag description

The message tag for the action is described in Table 2.36.

<table>
<thead>
<tr>
<th>XML tag</th>
<th>Attribute</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>action_get</td>
<td>-</td>
<td>Yes</td>
<td>The action_get tag is used to get actions</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>See examples below.</td>
</tr>
<tr>
<td>action_id</td>
<td>No</td>
<td></td>
<td>ID of the action</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>To get all the actions omit action_id</td>
</tr>
<tr>
<td>action_report</td>
<td>-</td>
<td>Yes</td>
<td>The action_report tag is used to report actions.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>See examples below.</td>
</tr>
<tr>
<td>action_set</td>
<td>-</td>
<td>Yes</td>
<td>The action_set tag is used to set actions.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>See examples below.</td>
</tr>
<tr>
<td>action_delete</td>
<td>-</td>
<td>Yes</td>
<td>The action_delete tag is used to delete actions.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>See examples below.</td>
</tr>
<tr>
<td>action_id</td>
<td>No</td>
<td></td>
<td>ID of the action</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>To delete all the actions omit action_id</td>
</tr>
<tr>
<td>action_invoke</td>
<td>-</td>
<td>Yes</td>
<td>This tag is used to execute an action directly</td>
</tr>
<tr>
<td>action_id</td>
<td>Yes</td>
<td></td>
<td>ID of the action</td>
</tr>
<tr>
<td>action_get_memory</td>
<td>-</td>
<td>Yes</td>
<td>This tag is used to request the action memory information from a device</td>
</tr>
<tr>
<td>action_report_memory</td>
<td>-</td>
<td>Yes</td>
<td>This tag is used to report the action memory information from a device</td>
</tr>
<tr>
<td>count</td>
<td>Yes</td>
<td></td>
<td>The maximum number of actions the device supports</td>
</tr>
<tr>
<td>free_count</td>
<td>Yes</td>
<td></td>
<td>The used number of actions</td>
</tr>
</tbody>
</table>

Table 2.36: Action tags

The child tags for the action_report and action_set message are described in Table 2.37.

<table>
<thead>
<tr>
<th>XML tag</th>
<th>Attribute</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>action</td>
<td>-</td>
<td>Yes</td>
<td>The action</td>
</tr>
<tr>
<td>action_id</td>
<td>Yes</td>
<td></td>
<td>ID of the action</td>
</tr>
</tbody>
</table>

Table 2.37: Action_report and action_set child tags
The child tags for the action are described in Table 2.38.

<table>
<thead>
<tr>
<th>XML tag</th>
<th>Attribute</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>get</td>
<td></td>
<td></td>
<td>Get the value for the specified value on the specified device</td>
</tr>
<tr>
<td></td>
<td>value_id</td>
<td>Yes</td>
<td>ID of the value</td>
</tr>
<tr>
<td></td>
<td>partner_id</td>
<td>No</td>
<td>ID for the device where the value should be retrieved from. If no partner_id is present then the value is from the device itself.</td>
</tr>
<tr>
<td></td>
<td>transport_mode</td>
<td>No</td>
<td>The transport mode the action should be send with. See Table 2.39</td>
</tr>
<tr>
<td>set</td>
<td></td>
<td></td>
<td>Set the value of the specified device.</td>
</tr>
<tr>
<td></td>
<td>value_id</td>
<td>Yes</td>
<td>ID of the value</td>
</tr>
<tr>
<td></td>
<td>partner_id</td>
<td>No</td>
<td>ID for the device where the value should be set. If no partner_id is present then the value will be set on the device itself.</td>
</tr>
<tr>
<td></td>
<td>number</td>
<td>Yes*</td>
<td>The number value *If not defined string, number or calculation_id must be defined</td>
</tr>
<tr>
<td></td>
<td>string</td>
<td>Yes*</td>
<td>The string value *If not defined number, hexBinary or calculation_id must be defined</td>
</tr>
<tr>
<td></td>
<td>hexBinary</td>
<td>Yes*</td>
<td>The hex value *If not defined number, string or calculation_id must be defined</td>
</tr>
<tr>
<td></td>
<td>calculation_id</td>
<td>Yes*</td>
<td>The calculation ID *If not defined number or string must be defined</td>
</tr>
<tr>
<td></td>
<td>transport_mode</td>
<td>No</td>
<td>The transport mode the action should be send with. See Table 2.39</td>
</tr>
<tr>
<td>report</td>
<td></td>
<td></td>
<td>Sending a report to the specified device.</td>
</tr>
<tr>
<td></td>
<td>my_value_id</td>
<td>Yes</td>
<td>ID of the value from the device itself that is used in the report</td>
</tr>
<tr>
<td></td>
<td>partner_id</td>
<td>No</td>
<td>ID for the device where the report should be sent to. If no partner_id is present then the report will be sent to the device itself.</td>
</tr>
<tr>
<td></td>
<td>transport_mode</td>
<td>No</td>
<td>The transport mode the action should be send with. See Table 2.39</td>
</tr>
<tr>
<td>timer_start</td>
<td></td>
<td>Yes</td>
<td>Start a timer</td>
</tr>
<tr>
<td></td>
<td>timer_id</td>
<td>Yes</td>
<td>ID of the timer</td>
</tr>
<tr>
<td>timer_stop</td>
<td></td>
<td>Yes</td>
<td>Stop a timer</td>
</tr>
<tr>
<td></td>
<td>timer_id</td>
<td>Yes</td>
<td>ID of the timer</td>
</tr>
</tbody>
</table>

Table 2.38: Action child tags

### Transport mode

<table>
<thead>
<tr>
<th>Mode</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>UDP</td>
<td>Fast and simple transport mode</td>
</tr>
<tr>
<td>1</td>
<td>TCP</td>
<td>Slow but more reliable transport mode</td>
</tr>
</tbody>
</table>

Table 2.39: Transport mode
2.3.10.2 Examples

The following section contains XML examples for using the 'action_get', 'action_report', 'action_set' and 'action_delete' messages.

Get a specific action

In order to get a specific action, an 'action_get' message has to be sent to the device with an action ID. See XML 2.39 for an example of the 'get' message.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
  xsi:noNamespaceSchemaLocation="action.xsd">
  <device version="1">
    <action_get action_id="1"/>
  </device>
</network>
```

XML 2.39: Get the action with ID 1

Get all actions

In order to get all actions, an 'action_get' message has to be sent to the device without an action ID. See XML 2.40 for an example of the 'get all' message.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
  xsi:noNamespaceSchemaLocation="action.xsd">
  <device version="1">
    <action_get/>
  </device>
</network>
```

XML 2.40: Get all actions

Report an action

To report a specific action, the 'action_report' message has to be sent from the device with an action ID and its corresponding child tags as described earlier. See XML 2.41 for an example of the 'report' message.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
  xsi:noNamespaceSchemaLocation="action.xsd">
  <device version="1">
    <action_report>
      <action action_id="2">
        <set value_id="3" partner_id="23" number="12"/>
      </action>
    </action_report>
  </device>
</network>
```

XML 2.41: Report the set action with ID 2
Report all actions

To report all actions, the "action_report" message has to be sent from the device with an action ID and its corresponding child tags as described earlier. See XML 2.42 for an example of the "report all" message.

```
<?xml version="1.0" encoding="UTF-8"?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
  xsi:noNamespaceSchemaLocation="action.xsd">
  <device version="1">
    <action_report>
      <action action_id="1">
        <get value_id="2" partner_id="34"/>
        <set value_id="2" partner_id="34" number="12"/>
      </action>
      <action action_id="3">
        <report my_value_id="4" partner_id="12"/>
      </action>
      <action action_id="4">
        <timer_start timer_id="1"/>
        <timer_stop timer_id="2"/>
      </action>
    </action_report>
  </device>
</network>
```

XML 2.42: Report all actions

Set a specific action

In order to set a specific action, an "action_set" message has to be sent to the device with an action ID and its corresponding child tags as described earlier. See XML 2.43 for an example of the "set" message.

```
<?xml version="1.0" encoding="UTF-8"?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
  xsi:noNamespaceSchemaLocation="action.xsd">
  <device version="1">
    <action_set>
      <action action_id="1">
        <get value_id="2" partner_id="34"/>
      </action>
    </action_set>
  </device>
</network>
```

XML 2.43: Set the action with ID 1
Set multiple actions

In order to set multiple actions, an 'action_set' message has to be sent to the device with an action ID and its corresponding child tags as described earlier. See XML 2.44 for an example of the 'set multiple' message.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
xsi:noNamespaceSchemaLocation="action.xsd"/>
<device version="1">
  <action_set>
    <get value_id="2" partner_id="34"/>
    <set value_id="2" partner_id="34" string="ON"/>
  </action_set>
</device>
</network>

XML 2.44: Set multiple actions
```

Delete a specific action

In order to delete a specific action, an 'action_delete' message has to be sent to the device with an action ID. See XML 2.45 for an example of the 'delete' message.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
xsi:noNamespaceSchemaLocation="action.xsd"/>
<device version="1">
  <action_delete action_id="5"/>
</device>
</network>

XML 2.45: Delete the action with ID 5
```

Delete all actions

In order to delete all actions, an 'action_delete' message has to be sent to the device without an action ID. See XML 2.46 for an example of the 'delete all' message.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
xsi:noNamespaceSchemaLocation="action.xsd"/>
<device version="1">
  <action_delete/>
</device>
</network>

XML 2.46: Delete all actions
2.3.11 Calculation

The calculation message is used to perform specific calculations.

The calculation message is maintained by get, report, set and delete tags. Each calculation can be the result of an addition, subtraction, multiplication or division, defined by the Method attribute. The result of a calculation is split into two parts, a left and right which can be readout separately.

2.3.11.1 Tag description

The message tag for the calculation is described in Table 2.40.

<table>
<thead>
<tr>
<th>XML tag</th>
<th>Attribute</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>calculation_get</td>
<td>-</td>
<td>Yes</td>
<td>The calculation_get tag is used to get calculations. See examples below.</td>
</tr>
<tr>
<td></td>
<td>calculation_id</td>
<td>No</td>
<td>ID of the calculation. To get all the calculation omit calculation_id</td>
</tr>
<tr>
<td>calculation_report</td>
<td>-</td>
<td>Yes</td>
<td>The calculation_report tag is used to report calculations. See examples below</td>
</tr>
<tr>
<td>calculation_set</td>
<td>-</td>
<td>Yes</td>
<td>The calculation_set tag is used to set calculations. See examples below</td>
</tr>
<tr>
<td>calculation_delete</td>
<td>-</td>
<td>Yes</td>
<td>The calculation_delete tag is used to delete calculations. See examples below</td>
</tr>
<tr>
<td></td>
<td>calculation_id</td>
<td>No</td>
<td>ID of the calculation. To delete all the calculation omit calculation_id</td>
</tr>
<tr>
<td>calculation_get_memory</td>
<td>-</td>
<td>Yes</td>
<td>This tag is used to request the calculation memory information from a device</td>
</tr>
<tr>
<td>calculation_report_memory</td>
<td>-</td>
<td>Yes</td>
<td>This tag is used to report the calculation memory information from a device</td>
</tr>
<tr>
<td>count</td>
<td>Yes</td>
<td></td>
<td>The maximum number of calculations the device supports</td>
</tr>
<tr>
<td>free_count</td>
<td>Yes</td>
<td></td>
<td>The used number of calculations</td>
</tr>
</tbody>
</table>

Table 2.40: Calculation tags

The child tags for the calculation_report and calculation_set message are described in Table 2.41.

<table>
<thead>
<tr>
<th>XML tag</th>
<th>Attribute</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>calculation</td>
<td>-</td>
<td>Yes</td>
<td>The calculation tag is used define a single calculation</td>
</tr>
<tr>
<td></td>
<td>calculation_id</td>
<td>Yes</td>
<td>ID of the calculation</td>
</tr>
<tr>
<td></td>
<td>method_id</td>
<td>Yes</td>
<td>The calculation method is used to calculate the result using the two values, left and right. See Table 2.42</td>
</tr>
</tbody>
</table>

Table 2.41: Calculation message child tags
The child tags for the calculation are described in Table 2.42.

<table>
<thead>
<tr>
<th>XML tag</th>
<th>Attribute</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>left / right</td>
<td>-</td>
<td>Yes</td>
<td>The left or right value of the calculation</td>
</tr>
<tr>
<td>value_id</td>
<td>Yes*</td>
<td></td>
<td>ID of the value</td>
</tr>
<tr>
<td>calculation_id</td>
<td>Yes*</td>
<td></td>
<td>ID of the calculation*</td>
</tr>
<tr>
<td>partner_id</td>
<td>No</td>
<td></td>
<td>ID for the device with the value</td>
</tr>
<tr>
<td>statemachine_id</td>
<td>No</td>
<td></td>
<td>ID of the state machine</td>
</tr>
<tr>
<td>timer_id</td>
<td>Yes</td>
<td></td>
<td>ID of the timer</td>
</tr>
<tr>
<td>calendar_id</td>
<td>Yes</td>
<td></td>
<td>ID of the calendar task</td>
</tr>
<tr>
<td>constant_string</td>
<td>No</td>
<td></td>
<td>A constant string value</td>
</tr>
<tr>
<td>constant_number</td>
<td>No</td>
<td></td>
<td>A constant number value</td>
</tr>
<tr>
<td>constant_hexBinary</td>
<td>No</td>
<td></td>
<td>A constant hex value</td>
</tr>
<tr>
<td>is_updated</td>
<td>No</td>
<td></td>
<td>If this is set to the value 1, then the calculation will return true</td>
</tr>
</tbody>
</table>

*If no value_id is present then calculation_id must be defined
*If no calculation_id is present then value_id must be defined

If no partner_id is present then the value is from the device itself.

Table 2.42: Calculation child tags

**Method**

The valid options for the "method_id" attribute in the calculation are listed in Table 2.43. The "method_id" attribute is a required and should be set to a valid option.

<table>
<thead>
<tr>
<th>Method Id</th>
<th>Method</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Add</td>
</tr>
<tr>
<td>2</td>
<td>Subtract</td>
</tr>
<tr>
<td>3</td>
<td>Multiply</td>
</tr>
<tr>
<td>4</td>
<td>Divide</td>
</tr>
<tr>
<td>5</td>
<td>Modulo</td>
</tr>
<tr>
<td>6</td>
<td>Equal</td>
</tr>
<tr>
<td>7</td>
<td>Not Equal</td>
</tr>
<tr>
<td>8</td>
<td>Smaller</td>
</tr>
<tr>
<td>9</td>
<td>Greater</td>
</tr>
<tr>
<td>10</td>
<td>Smaller or equal</td>
</tr>
<tr>
<td>11</td>
<td>Greater or equal</td>
</tr>
<tr>
<td>12</td>
<td>And</td>
</tr>
<tr>
<td>13</td>
<td>Or</td>
</tr>
<tr>
<td>...</td>
<td>...</td>
</tr>
</tbody>
</table>

Table 2.43: Method types
2.3.11.2 Examples

The following section contains XML examples for using the 'calculation_get', 'calculation_report', 'calculation_set' and 'calculation_delete' messages.

Get a specific calculation

In order to get a specific calculation, a 'calculation_get' message has to be sent to the device with a calculation ID. See XML 2.47 for an example of the 'get' message.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1">
  <device version="1">
    <calculation_get calculation_id="1"/>
  </device>
</network>
```

XML 2.47: Get the calculation with ID 1

Get all calculations

In order to get all calculations, a 'calculation_get' message has to be sent to the device without a calculation ID. See XML 2.48 for an example of the 'get all' message.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1">
  <device version="1">
    <calculation_get/>
  </device>
</network>
```

XML 2.48: Get all calculations

Report a specific calculation

To report a calculation, the 'calculation_report' message has to be sent from the device with its corresponding child tags as described earlier. See XML 2.49 for an example of the 'report' message.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1">
  <device version="1">
    <calculation_report>
      <calculation calculation_id="2" method_id="3">
        <left calculation_id="1"/>
        <right constant_number="4"/>
      </calculation>
    </calculation_report>
  </device>
</network>
```

XML 2.49: Report the calculation multiply-method
Report all calculations

To report all calculations, the 'calculation_report' message has to be sent from the device with its corresponding child tags as described earlier. See XML 2.50 for an example of the 'report all' message.

```
<?xml version="1.0" encoding="UTF-8"?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
 xsi:noNamespaceSchemaLocation="calculation.xsd">
 <device version="1">
   <calculation_report>
     <calculation calculation_id="1" method_id="1">
       <left value_id="1"/>
     </calculation>
     <calculation calculation_id="2" method_id="3">
       <left calculation_id="1"/>
       <right constant_number="4"/>
     </calculation>
     <calculation calculation_id="3" method_id="6">
       <left value_id="1"/>
       <right constant_number="1"/>
     </calculation>
   </calculation_report>
 </device>
</network>
```

XML 2.50: Report all calculations

Set a specific calculation

In order to set a specific calculation, a 'calculation_set' message has to be sent to the device with its corresponding child tags as described earlier. See XML 2.51 for an example of the 'set' message.

```
<?xml version="1.0" encoding="UTF-8"?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
 xsi:noNamespaceSchemaLocation="calculation.xsd">
 <device version="1">
   <calculation_set>
     <calculation calculation_id="1" method_id="1">
       <left value_id="1"/>
     </calculation>
   </calculation_set>
 </device>
</network>
```

XML 2.51: Set the calculation add-method
Set multiple calculations

In order to set multiple calculations, a 'calculation_set' message has to be sent to the device with its corresponding child tags as described earlier. See XML 2.52 for an example of the 'set multiple' message.

```
<?xml version='1.0' encoding='UTF-8'?>
<network xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance' version='1' xsi:noNamespaceSchemaLocation='calculation.xsd'>
  <device version='1'>
    <calculation_set>
      <calculation calculation_id='1' method_id='1'>
        <left value_id='1'/>
        <right value_id='1' partner_id='1'/>
      </calculation>
      <calculation calculation_id='2' method_id='3'>
        <left calculation_id='1'/>
        <right constant_number='4'/>
      </calculation>
      <calculation calculation_id='3' method_id='6'>
        <left value_id='1'/>
        <right constant_string='1'/>
      </calculation>
    </calculation_set>
  </device>
</network>
```

XML 2.52: Set multiple calculations

Delete a specific calculation

In order to delete a specific calculation, a 'calculation_delete' message has to be sent to the device with a calculation ID. See XML 2.53 for an example of the 'delete' message.

```
<?xml version='1.0' encoding='UTF-8'?>
<network xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance' version='1' xsi:noNamespaceSchemaLocation='calculation.xsd'>
  <device version='1'>
    <calculation_delete calculation_id='1'/>
  </device>
</network>
```

XML 2.53: Delete the calculation with ID 1

Delete all calculations

In order to delete all calculations, a 'calculation_delete' message has to be sent to the device without a calculation ID. See XML 2.54 for an example of the 'delete all' message.

```
<?xml version='1.0' encoding='UTF-8'?>
<network xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance' version='1' xsi:noNamespaceSchemaLocation='calculation.xsd'>
  <device version='1'>
    <calculation_delete/>
  </device>
</network>
```

XML 2.54: Delete all calculations

Fixme Note: Add is_updated example
2.3.12 Timer

The timer message is used to manage a list of conditional timers. The timer message is maintained by get, report, set and delete tags.

Each timer will performed a specified action after the set number of milliseconds defined by the after attribute. Furthermore a timer will only execute the action, if a specific calculation is met as defined by the calculation_id attribute. Both are attributes of the child tag Execute.

2.3.12.1 Tag description

The message tag for the timer is described in Table 2.44.

<table>
<thead>
<tr>
<th>XML tag</th>
<th>Attribute</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>timer_get</td>
<td>-</td>
<td>Yes</td>
<td>The timer_get tag is used to get timers.</td>
</tr>
<tr>
<td></td>
<td>timer_id</td>
<td>No</td>
<td>ID of the timer</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>To get all the timers omit timer_id</td>
</tr>
<tr>
<td>timer_report</td>
<td>-</td>
<td>Yes</td>
<td>The timer_report tag is used to report timers.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>See examples below.</td>
</tr>
<tr>
<td>timer_set</td>
<td>-</td>
<td>Yes</td>
<td>The timer_set tag is used to set timers.</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>See examples below.</td>
</tr>
<tr>
<td>timer_delete</td>
<td>-</td>
<td>Yes</td>
<td>The timer_delete tag is used to delete timers.</td>
</tr>
<tr>
<td></td>
<td>timer_id</td>
<td>No</td>
<td>ID of the timer</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>To delete all the timers omit timer_id</td>
</tr>
<tr>
<td>timer_get_memory</td>
<td>-</td>
<td>Yes</td>
<td>This tag is used to request the timer memory information from a device</td>
</tr>
<tr>
<td>timer_report_memory</td>
<td>-</td>
<td>Yes</td>
<td>This tag is used to report the timer memory information from a device</td>
</tr>
<tr>
<td></td>
<td>count</td>
<td>Yes</td>
<td>The maximum number of timers the device supports</td>
</tr>
<tr>
<td></td>
<td>free_count</td>
<td>Yes</td>
<td>The used number of timers</td>
</tr>
</tbody>
</table>

Table 2.44: Timer tags

The child tags for the timer_report and timer_set message are described in Table 2.45.

<table>
<thead>
<tr>
<th>XML tag</th>
<th>Attribute</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>execute</td>
<td>-</td>
<td>Yes</td>
<td>The timer will execute the action after the specified milliseconds in the</td>
</tr>
<tr>
<td></td>
<td>timer_id</td>
<td>Yes</td>
<td>ID of the timer</td>
</tr>
<tr>
<td></td>
<td>after</td>
<td>Yes</td>
<td>Number of milliseconds before execution</td>
</tr>
<tr>
<td></td>
<td>calculation_id</td>
<td>No</td>
<td>ID of the calculation. The timer will only execute if the calculation is</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>true</td>
</tr>
<tr>
<td></td>
<td>action_id</td>
<td>No</td>
<td>ID of the action that will be executed. If the no action_id is present</td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>then an event will be sent to the state machine.</td>
</tr>
</tbody>
</table>

Table 2.45: Timer message child tags
2.3.12.2 Examples

The following section contains XML examples for using the 'timer_get', 'timer_report', 'timer_set' and 'timer_delete' messages.

Get a specific timer

In order to get a specific timer, a 'timer_get' message has to be sent to the device with a timer ID. See XML 2.55 for an example of the 'get' message.

```
<?xml version="1.0" encoding="UTF-8"?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
    xsi:noNamespaceSchemaLocation="timer.xsd">
    <device version="1">
        <timer_get timer_id="1"/>
    </device>
</network>
```

XML 2.55: Get the timer with ID 1

Get all timers

In order to get all timers, a 'timer_get' message has to be sent to the device without a timer ID. See XML 2.56 for an example of the 'get all' message.

```
<?xml version="1.0" encoding="UTF-8"?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
    xsi:noNamespaceSchemaLocation="timer.xsd">
    <device version="1">
        <timer_get/>
    </device>
</network>
```

XML 2.56: Get all the timers

Report timer with an action

To report a timer with an action, the 'timer_report' message has to be sent to the device with its corresponding child tags as described earlier. See XML 2.57 for an example of the 'report with an action' message.

```
<?xml version="1.0" encoding="UTF-8"?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
    xsi:noNamespaceSchemaLocation="timer.xsd">
    <device version="1">
        <timer_report>
            <execute timer_id="1" after="2000" action_id="2"/>
        </timer_report>
    </device>
</network>
```

XML 2.57: Report for the timer with ID 1 that will execute action 2 after 2000 ms
Report timer without an action

To report a timer without an action, the "timer_report" message has to be sent to the device with its corresponding child tags as described earlier. See XML 2.58 for an example of the 'report without an action' message.

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
 xsi:noNamespaceSchemaLocation="timer.xsd">
 <device version="1">
  <timer_report>
   <execute timer_id="2" after="4000"/>
  </timer_report>
 </device>
</network>
```

XML 2.58: Report for the timer with ID 2 that will execute after 4000 ms

Report all timers

To report all timers, the "timer_report" message has to be sent to the device with its corresponding child tags as described earlier. See XML 2.59 for an example of the 'report all' message.

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
 xsi:noNamespaceSchemaLocation="timer.xsd">
 <device version="1">
  <timer_report>
   <execute timer_id="1" after="2000" action_id="2"/>
   <execute timer_id="2" after="4000"/>
  </timer_report>
 </device>
</network>
```

XML 2.59: Report all timers

Set a specific timer

In order to set a timer, a 'timer_set' message has to be sent to the device with its corresponding child tags as described earlier. See XML 2.60 for an example of the 'set' message.

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
 xsi:noNamespaceSchemaLocation="timer.xsd">
 <device version="1">
  <timer_set>
   <execute timer_id="2" after="2000" action_id="3"/>
  </timer_set>
 </device>
</network>
```

XML 2.60: Set the timer with ID 2 to execute action 3 after 2000 ms
Set multiple timers

In order to set multiple timers, a "timer_set" message has to be sent to the device with its corresponding child tags as described earlier. See XML 2.61 for an example of the 'set multiple' message.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
         xsi:noNamespaceSchemaLocation="timer.xsd">
  <device version="1">
    <timer_set>
      <execute timer_id="1" after="2000" action_id="2"/>
      <execute timer_id="2" after="4000"/>
    </timer_set>
  </device>
</network>
```

XML 2.61: Set the timers with ID 1 and 2

Delete a specific timer

In order to delete a specific timer, a "timer_delete" message has to be sent to the device with a timer ID. See XML 2.62 for an example of the 'delete' message.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
         xsi:noNamespaceSchemaLocation="timer.xsd">
  <device version="1">
    <timer_delete timer_id="1"/>
  </device>
</network>
```

XML 2.62: Delete the timer with ID 1

Delete all timers

In order to delete all timers, a "timer_delete" message has to be sent to the device without a timer ID. See XML 2.63 for an example of the 'delete all' message.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
         xsi:noNamespaceSchemaLocation="timer.xsd">
  <device version="1">
    <timer_delete/>
  </device>
</network>
```

XML 2.63: Delete all the timers
2.3.13 Calendar

The calendar message is used to obtain a list of scheduled tasks.

The calendar message is maintained by get, report, set and delete tags. Each task in the calendar will execute at the date and time set by the start attribute, and continue until the end date is reached which is set by the end attribute. It is possible to define that a task in the calendar should be repeated either with specific number seconds between each pass. The weekdays attribute is a filter for which weekdays the repeated task should trigger.

2.3.13.1 Tag description

The message tag for the calendar is described in Table 2.46.

<table>
<thead>
<tr>
<th>XML tag</th>
<th>Attribute</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>calendar_get</td>
<td>-</td>
<td>Yes</td>
<td>The calendar_get tag is used to get all the calendar tasks. See examples below.</td>
</tr>
<tr>
<td></td>
<td>task_id</td>
<td>No</td>
<td>ID of the task. To get all the calendar tasks omit task_id.</td>
</tr>
<tr>
<td>calendar_report</td>
<td>-</td>
<td>Yes</td>
<td>The calendar_report tag is used to report all the calendar tasks. See examples below.</td>
</tr>
<tr>
<td>calendar_set</td>
<td>-</td>
<td>Yes</td>
<td>The calendar_set tag is used to set multiple the calendar tasks. See examples below.</td>
</tr>
<tr>
<td>calendar_delete</td>
<td>-</td>
<td>Yes</td>
<td>The calendar_delete tag is used to delete all the calendar tasks. See examples below.</td>
</tr>
<tr>
<td></td>
<td>task_id</td>
<td>No</td>
<td>ID of the task. To delete all the calendar task omit task_id</td>
</tr>
<tr>
<td>calendar_get_timezone</td>
<td>-</td>
<td>Yes</td>
<td>This tag is used to request the timezone offset from a device</td>
</tr>
<tr>
<td>calendar_set_timezone</td>
<td>-</td>
<td>Yes</td>
<td>This tag is used the set the timezone offset on a device</td>
</tr>
<tr>
<td></td>
<td>offset</td>
<td>Yes</td>
<td>The timezone offset</td>
</tr>
<tr>
<td>calendar_report_timezone</td>
<td>-</td>
<td>Yes</td>
<td>This tag is used to report the timezone offset from a device</td>
</tr>
<tr>
<td></td>
<td>offset</td>
<td>Yes</td>
<td>The timezone offset</td>
</tr>
<tr>
<td>calendar_get_memory</td>
<td>-</td>
<td>Yes</td>
<td>This tag is used to request the calendar memory information from a device</td>
</tr>
<tr>
<td>calendar_report_memory</td>
<td>-</td>
<td>Yes</td>
<td>This tag is used to report the calendar memory information from a device</td>
</tr>
<tr>
<td></td>
<td>count</td>
<td>Yes</td>
<td>The maximum number of calendar task the device supports</td>
</tr>
<tr>
<td></td>
<td>free_count</td>
<td>Yes</td>
<td>The used number of calendar tasks</td>
</tr>
</tbody>
</table>

Table 2.46: Calendar tags

The child tags for the calendar_report and calendar_set message are described in Table 2.47.

<table>
<thead>
<tr>
<th>XML tag</th>
<th>Attribute</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>task</td>
<td>-</td>
<td>Yes*</td>
<td>The task will execute a the specified date and time</td>
</tr>
<tr>
<td></td>
<td>task_id</td>
<td>Yes</td>
<td>ID of the task</td>
</tr>
<tr>
<td>start</td>
<td>Yes</td>
<td></td>
<td>The start date of the task in seconds since 01/01-1970.</td>
</tr>
<tr>
<td>action_id</td>
<td>Yes</td>
<td></td>
<td>ID for the action that will be executed.</td>
</tr>
<tr>
<td>end</td>
<td>No</td>
<td></td>
<td>The end date of the task in seconds since 01/01-1970.</td>
</tr>
<tr>
<td>repeat</td>
<td>No</td>
<td></td>
<td>When the task shall be repeated. The time is in seconds (Real number ex. 3600)</td>
</tr>
<tr>
<td>weekdays</td>
<td>No</td>
<td></td>
<td>On which weekdays the task should be executed. This attribute is a bitmap, where bit 1 is monday and bit 7 is sunday. Bit 8 is not used.</td>
</tr>
</tbody>
</table>

Table 2.47: Calendar child tags
2.3.13.2 Examples

The following section contains XML examples for using the `calendar_get`, `calendar_report`, `calendar_set` and `calendar_delete` messages.

Get a specific calendar task

In order to get a specific calendar task, a `calendar_get` message has to be sent to the device with a task ID. See XML 2.64 for an example of the get message.

```
<?xml version="1.0" encoding="UTF-8"?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
    xsi:noNamespaceSchemaLocation="calendar.xsd">
  <device version="1">
    <calendar_get task_id="1"/>
  </device>
</network>
```

XML 2.64: Get the calendar task with task ID 1

Get all calendars

In order to get all calendar tasks, a `calendar_get` message has to be sent to the device without a task ID. See XML 2.65 for an example of the get all message.

```
<?xml version="1.0" encoding="UTF-8"?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
    xsi:noNamespaceSchemaLocation="calendar.xsd">
  <device version="1">
    <calendar_get/>
  </device>
</network>
```

XML 2.65: Get all the calendar tasks

Report a specific calendar task

To report a specific calendar task, the `calendar_report` message has to be sent from the device with a task ID and its corresponding child tags as described earlier. See XML 2.66 for an example of the report message.

```
<?xml version="1.0" encoding="UTF-8"?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
    xsi:noNamespaceSchemaLocation="calendar.xsd">
  <device version="1">
    <calendar_report>
      <task task_id="1" start="1314976980" action_id="1" repeat="86400"/>
    </calendar_report>
  </device>
</network>
```

XML 2.66: Report the calendar task with ID 1
Report all calendar tasks

To report all calendar tasks, the calendar_report message has to be sent from the device with a task ID and its corresponding child tags as described earlier. See XML 2.67 for an example of the report all message.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
    xsi:noNamespaceSchemaLocation="calendar.xsd">
  <device version="1">
    <calendar_report>
      <task task_id="1" start="1314976980" action_id="1" end="1314980580" repeat="3600"/>
      <task task_id="2" start="1314976980" action_id="2" repeat="604800"/>
    </calendar_report>
  </device>
</network>
```

XML 2.67: Report all calendar tasks

Set a specific calendar task

In order to set a specific calendar task, a calendar_set message has to be sent to the device with a task ID and its corresponding child tags as described earlier. See XML 2.68 for an example of the set message.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
    xsi:noNamespaceSchemaLocation="calendar.xsd">
  <device version="1">
    <calendar_set>
      <task task_id="1" start="1314980580" action_id="1" repeat="86400"/>
    </calendar_set>
  </device>
</network>
```

XML 2.68: Set the calendar task with ID 1

Set multiple calendar tasks

In order to set multiple calendar tasks, a calendar_set message has to be sent to the device with a task ID and its corresponding child tags as described earlier. See XML 2.69 for an example of the set multiple message.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
    xsi:noNamespaceSchemaLocation="calendar.xsd">
  <device version="1">
    <calendar_set>
      <task task_id="1" start="1314976980" action_id="1" end="1314980580" repeat="3600"/>
      <task task_id="2" start="1314976980" action_id="2" repeat="604800"/>
    </calendar_set>
  </device>
</network>
```

XML 2.69: Set multiple calendar tasks
Delete a specific calendar task

In order to delete a specific calendar task, a `calendar_delete` message has to be sent to the device with a task ID. See XML 2.70 for an example of the delete message.

```
<?xml version="1.0" encoding="UTF-8"?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
  xsi:noNamespaceSchemaLocation="calendar.xsd">
  <device version="1">
    <calendar_delete task_id="1"/>
  </device>
</network>
```

XML 2.70: Delete the calendar task with ID 1

Delete all calendar tasks

In order to delete all calendar tasks, a `calendar_delete` message has to be sent to the device without a task ID. See XML 2.71 for an example of the delete all message.

```
<?xml version="1.0" encoding="UTF-8"?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
  xsi:noNamespaceSchemaLocation="calendar.xsd">
  <device version="1">
    <calendar_delete/>
  </device>
</network>
```

XML 2.71: Delete all calendar tasks
2.3.14 State machine

The state machine message is used to obtain a list of current state operations. The state machine message is maintained by get, report, set and delete tags, as well as the state specific tags get_state and report_state. Each state machine is built from a list of valid states, and controlled by the transaction child tag. This transaction describes which action should be performed upon entering a given state, and to which state the state machine should go to next.

2.3.14.1 Tag description

The message tag for the state machine is described in Table 2.48.
<table>
<thead>
<tr>
<th>XML tag</th>
<th>Attribute</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>statemachine_get</td>
<td>-</td>
<td>Yes</td>
<td>The statemachine_get tag is used to get states and transactions of the state machine. See examples below.</td>
</tr>
<tr>
<td>statemachine_id</td>
<td>No</td>
<td></td>
<td>ID of the state machine. To get all the state machines omit statemachine_id.</td>
</tr>
<tr>
<td>statemachine_report</td>
<td>-</td>
<td>Yes</td>
<td>The statemachine_report tag is used to report states and transactions of the state machine. See examples below.</td>
</tr>
<tr>
<td>statemachine_set</td>
<td>-</td>
<td>Yes</td>
<td>The statemachine_set tag is used to set states and transactions of the state machine. See examples below.</td>
</tr>
<tr>
<td>statemachine_delete</td>
<td>-</td>
<td>Yes</td>
<td>The statemachine_delete tag is used to delete states and state machines. See examples below.</td>
</tr>
<tr>
<td>statemachine_id</td>
<td>No</td>
<td></td>
<td>ID of the state machine. To delete all the state machines omit statemachine_id.</td>
</tr>
<tr>
<td>state_id</td>
<td>No</td>
<td></td>
<td>ID of the state. To delete all the states omit state_id.</td>
</tr>
<tr>
<td>statemachine_get_state</td>
<td>-</td>
<td>Yes</td>
<td>The statemachine_get_state tag is used to get the current state of the state machine. See examples below.</td>
</tr>
<tr>
<td>statemachine_id</td>
<td>No</td>
<td></td>
<td>ID of the state machine. To get the current state of all the state machines omit statemachine_id.</td>
</tr>
<tr>
<td>statemachine_report_state</td>
<td>-</td>
<td>Yes</td>
<td>The statemachine_report_state tag is used to report the current state of the state machine. See examples below.</td>
</tr>
<tr>
<td>statemachine_id</td>
<td>No</td>
<td></td>
<td>ID of the state machine. To report the current state for all the state machines omit statemachine_id.</td>
</tr>
<tr>
<td>statemachine_set_state</td>
<td>-</td>
<td>Yes</td>
<td>This tag is used to set the current state of the state machine. See examples below.</td>
</tr>
<tr>
<td>statemachine_get_memory</td>
<td>-</td>
<td>Yes</td>
<td>This tag is used to request the state machine memory information from a device</td>
</tr>
<tr>
<td>memory_info</td>
<td>count</td>
<td>Yes</td>
<td>The maximum number of statemachines the device supports</td>
</tr>
<tr>
<td>free_count</td>
<td>Yes</td>
<td></td>
<td>The used number of statemachines</td>
</tr>
<tr>
<td>statemachine_get_state_memory</td>
<td>-</td>
<td>Yes</td>
<td>This tag is used to request the statemachine state memory information from a device</td>
</tr>
<tr>
<td>state_memory_info</td>
<td>count</td>
<td>Yes</td>
<td>The maximum number of statemachine states the device supports</td>
</tr>
<tr>
<td>free_count</td>
<td>Yes</td>
<td></td>
<td>The used number of statemachine states</td>
</tr>
</tbody>
</table>

Table 2.48: State machine tags
The child tags for the statemachine_report and statemachine_set message are described in Table 2.49.

<table>
<thead>
<tr>
<th>XML tag</th>
<th>Attribute</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>statemachine</td>
<td>–</td>
<td>Yes</td>
<td>The statemachine</td>
</tr>
<tr>
<td>statemachine_id</td>
<td>Yes</td>
<td></td>
<td>ID of the state machine</td>
</tr>
</tbody>
</table>

Table 2.49: State machine message child tags

The child tags for the statemachine message are described in Table 2.50.

<table>
<thead>
<tr>
<th>XML tag</th>
<th>Attribute</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>state</td>
<td>–</td>
<td>Yes</td>
<td>The state the statemachine can enter.</td>
</tr>
<tr>
<td>state_id</td>
<td>Yes</td>
<td></td>
<td>ID of the state. The state_id is only present when all the states are returned</td>
</tr>
</tbody>
</table>

Table 2.50: Statemachine child tags

The child tags for the state are described in Table 2.51.

<table>
<thead>
<tr>
<th>XML tag</th>
<th>Attribute</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>transaction</td>
<td>–</td>
<td>Yes</td>
<td>Transaction is used for executing actions and changing states</td>
</tr>
<tr>
<td>calculation_id</td>
<td>No</td>
<td></td>
<td>ID of the calculation that needs to be true for this transaction to be executed</td>
</tr>
<tr>
<td>action_id</td>
<td>No</td>
<td></td>
<td>ID of the action the state machine will execute when this transaction is executed</td>
</tr>
<tr>
<td>goto_state_id</td>
<td>No</td>
<td></td>
<td>ID of the state the state machine will enter when this transaction is executed</td>
</tr>
</tbody>
</table>

Table 2.51: State child tags
2.3.14.2 Examples

The following section contains XML examples for using the 'statemachine_get', 'statemachine_report', 'statemachine_set', 'statemachine_delete', 'statemachine_get_state' and 'statemachine_report_state' messages.

Get a specific state machines specific state

In order to get a specific state from a specific state machine, a 'statemachine_get' message has to be sent to the device with a state machine ID and a state ID. See XML 2.72 for an example of the 'get' message.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
    xsi:noNamespaceSchemaLocation="statemachine.xsd">
    <device version="1">
        <statemachine_get statemachine_id="1" state_id="2"/>
    </device>
</network>
```

XML 2.72: Get the state machine with ID 1 and its state with ID 2

Get a specific state machine

In order to get a specific state machine, a 'statemachine_get' message has to be sent to the device with a state machine ID and without state ID. See XML 2.73 for an example of the 'get' message.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
    xsi:noNamespaceSchemaLocation="statemachine.xsd">
    <device version="1">
        <statemachine_get statemachine_id="1"/>
    </device>
</network>
```

XML 2.73: Get the complete state machine with ID 1

Get all state machines

In order to get all state machines, a 'statemachine_get' message has to be sent to the device without a state machine ID and state ID. See XML 2.74 for an example of the 'get all' message.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
    xsi:noNamespaceSchemaLocation="statemachine.xsd">
    <device version="1">
        <statemachine_get/>
    </device>
</network>
```

XML 2.74: Get all state machines

Report a specific state machines specific state

To report a specific state from a specific state machine, the 'statemachine_report' message has to be sent from the device with a state machine ID on its corresponding child tags as described earlier. See XML 2.75 for an example of the 'report' message.

Report a specific state machine

To report a specific state machine, the 'statemachine_report' message has to be sent from the device with a state machine ID on its corresponding child tags as described earlier. See XML 2.76 for an example of the 'report' message.

Report all state machines

To report all state machines, the 'statemachine_report' message has to be sent from the device with the state machine ID on its corresponding child tags as described earlier. See XML 2.77 for an example of the 'report all' message.
Set a specific state machines specific state

In order to set a specific state on a specific state machine, a 'statemachine_set' message has to be sent to the device
with a state machine ID on its corresponding child tags as described earlier. See XML 2.78 for an example of the "set" message.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
  xsi:noNamespaceSchemaLocation="statemachine.xsd">
  <device version="1">
    <statemachine_set>
      <statemachine statemachine_id="1">
        <state state_id="1">
          <transaction calculation_id="1" action_id="2" goto_state_id="2"/>
        </state>
      </statemachine>
    </statemachine_set>
  </device>
</network>
```

XML 2.78: Set the state machine with ID 1 state 1

**Set a specific state machine**

In order to set a specific state machine, a 'statemachine_set' message has to be sent to the device with a state machine ID on its corresponding child tags as described earlier. See XML 2.79 for an example of the 'set' message.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
  xsi:noNamespaceSchemaLocation="statemachine.xsd">
  <device version="1">
    <statemachine_set>
      <statemachine statemachine_id="1">
        <state state_id="1">
          <transaction calculation_id="1" action_id="2" goto_state_id="2"/>
        </state>
        <state state_id="2">
          <transaction calculation_id="2" action_id="1" goto_state_id="1"/>
          <transaction calculation_id="1" action_id="2"/>
        </state>
      </statemachine>
    </statemachine_set>
  </device>
</network>
```

XML 2.79: Set multiple states in the state machine with ID 1

**Set multiple state machines**

In order to set multiple state machines, a 'statemachine_set' message has to be sent to the device with the state machine ID on its corresponding child tags as described earlier. See XML 2.80 for an example of the 'set multiple' message.

**Delete a specific state machines specific state**

In order to delete a specific state in a specific state machine, a 'statemachine_delete' message has to be sent to the device with a state machine ID and state ID. See XML 2.81 for an example of the 'delete' message.

**Delete a specific state machine**

In order to delete a specific state machine, a 'statemachine_delete' message has to be sent to the device with a state machine ID and without state ID. See XML 2.82 for an example of the 'delete' message.

**Delete all state machines**

In order to delete all state machines, a 'statemachine_delete' message has to be sent to the device without a state machine ID and state ID. See XML 2.83 for an example of the 'delete all' message.
XML 2.80: Set multiple states in the state machine with ID 1 and 2

XML 2.81: Delete the state with ID 2 from the state machine with ID 1

XML 2.82: Delete the state machines with ID 1

XML 2.83: Delete all state machines

Get the current state
In order to get the current state from a state machine, a 'statemachine_get_state' message has to be sent to the device with a state machine ID. See XML 2.84 for an example of the 'get' message.

Report the current state
To report the current state of a state machine, the 'statemachine_report_state' message has to be sent from the
XML 2.84: Get the current state of the state machine

device with a state machine ID and its current state. See XML 2.85 for an example of the "report" message.

XML 2.85: Report the current state of the state machine
2.3.15 Firmware update
The firmware update is used to update the firmware on the devices.

2.3.15.1 Tag description
The message tag for the firmware update is described in Table 2.52.

<table>
<thead>
<tr>
<th>XML tag</th>
<th>Attribute</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>firmware_init</td>
<td></td>
<td>Yes</td>
<td>Initialize the firmware update process.</td>
</tr>
<tr>
<td>size</td>
<td></td>
<td>Yes</td>
<td>The size of the new firmware</td>
</tr>
<tr>
<td>firmware_id</td>
<td></td>
<td>Yes</td>
<td>A id of the new firmware</td>
</tr>
<tr>
<td>checksum</td>
<td></td>
<td>Yes</td>
<td>The checksum of the new firmware</td>
</tr>
<tr>
<td>firmware_data</td>
<td></td>
<td>Yes</td>
<td>Sends a chunk of the firmware image</td>
</tr>
<tr>
<td>offset</td>
<td></td>
<td>Yes</td>
<td>The offset in the firmware image</td>
</tr>
<tr>
<td>firmware_update_start</td>
<td></td>
<td>Yes</td>
<td>Starts the firmware update, if the checksum matches</td>
</tr>
<tr>
<td>firmware_report</td>
<td></td>
<td>Yes</td>
<td>Reports the status of the last firmware update message</td>
</tr>
<tr>
<td>status</td>
<td></td>
<td>Yes</td>
<td>The status of last message</td>
</tr>
<tr>
<td>expected_offset</td>
<td></td>
<td>No</td>
<td>The expected offset of the next firmware data message</td>
</tr>
<tr>
<td>firmware_information_get</td>
<td></td>
<td>Yes</td>
<td>Requests a firmware information report</td>
</tr>
<tr>
<td>firmware_information_report</td>
<td></td>
<td>Yes</td>
<td>Reports the current information about the firmware</td>
</tr>
<tr>
<td>size</td>
<td></td>
<td>Yes</td>
<td>The size of the firmware</td>
</tr>
<tr>
<td>firmware_id</td>
<td></td>
<td>Yes</td>
<td>The firmware ID</td>
</tr>
<tr>
<td>received_size</td>
<td></td>
<td>Yes</td>
<td>The number of bytes received</td>
</tr>
<tr>
<td>chunk_size</td>
<td></td>
<td>Yes</td>
<td>The size of the chunks in firmware data</td>
</tr>
</tbody>
</table>

Table 2.52: Firmware update tags

The child tags for the firmware_data message are described in Table 2.53.

<table>
<thead>
<tr>
<th>XML tag</th>
<th>Attribute</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>chunk</td>
<td></td>
<td>Yes</td>
<td>A chunk of the firmware image</td>
</tr>
</tbody>
</table>

Table 2.53: Firmware_data message child tags

Status
The valid options for the status attribute in the firmware_report are listed in Table 2.54.

<table>
<thead>
<tr>
<th>Status Id</th>
<th>Status Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>OK</td>
</tr>
<tr>
<td>2</td>
<td>Not Initialized</td>
</tr>
<tr>
<td>3</td>
<td>Size is to big</td>
</tr>
<tr>
<td>4</td>
<td>Checksum Error in received data</td>
</tr>
<tr>
<td>5</td>
<td>Data Overflow</td>
</tr>
<tr>
<td>6</td>
<td>Wrong Offset</td>
</tr>
<tr>
<td>7</td>
<td>Chunk size is to big</td>
</tr>
<tr>
<td>8</td>
<td>Data is missing</td>
</tr>
<tr>
<td>9</td>
<td>Chunk size is to small</td>
</tr>
</tbody>
</table>

Table 2.54: Firmware Update Status Type
2.3.15.2 Examples

The following section contains XML examples for using the firmware_update messages.

Initialize the firmware

In order to initialize the firmware, a firmware_init message has to be sent to the device with a size and a firmware ID. See XML 2.86 for an example of the init message.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
    xsi:noNamespaceSchemaLocation="firmware_update.xsd">
    <device version="1">
        <firmware_init size="4000" firmware_id="1" checksum="12AB"/>
    </device>
</network>
```

XML 2.86: Initialize the firmware

Send a chunk of the firmware image

In order to send a chunk of the firmware image, a firmware_data message has to be sent to the device with an offset and its corresponding child tags as described earlier. See XML 2.87 for an example of the data message.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
    xsi:noNamespaceSchemaLocation="firmware_update.xsd">
    <device version="1">
        <firmware_data offset="256">
            <chunk>73528A75284646B8465C9743F4</chunk>
        </firmware_data>
    </device>
</network>
```

XML 2.87: A chunk of the firmware image

Start the firmware update

In order to start the firmware update, a firmware_update_start message has to be sent to the device with a checksum. See XML 2.88 for an example of the start message.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
    xsi:noNamespaceSchemaLocation="firmware_update.xsd">
    <device version="1">
        <firmware_update_start/>
    </device>
</network>
```

XML 2.88: Start firmware update
Report the status of the last firmware update

In order to report the status of the last firmware update, a firmware_report message has to be sent from the device with a status and an expected offset. See XML 2.89 for an example of the report message.

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
  xsi:noNamespaceSchemaLocation="firmware_update.xsd">
  <device version="1">
    <firmware_report status="0" expected_offset="256"/>
  </device>
</network>
```

XML 2.89: Firmware report

Get the firmware information

In order to get the firmware information, a firmware_information_get message has to be sent to the device. See XML 2.90 for an example of the get message.

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
  xsi:noNamespaceSchemaLocation="firmware_update.xsd">
  <device version="1">
    <firmware_information_get/>
  </device>
</network>
```

XML 2.90: Get the firmware information

Report the firmware information

In order to report the firmware information, a firmware_information_report message has to be sent from the device with a size, a firmware ID, a received size and a chunk size. See XML 2.91 for an example of the report message.

```xml
<?xml version="1.0" encoding="UTF-8" ?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
  xsi:noNamespaceSchemaLocation="firmware_update.xsd">
  <device version="1">
    <firmware_information_report size="4000" firmware_id="1" received_size="512" chunk_size="256"/>
  </device>
</network>
```

XML 2.91: Firmware information report
2.3.16 Status

The status is used to report information from a device.

2.3.16.1 Tag description

The message tag for the status is described in Table 2.55.

<table>
<thead>
<tr>
<th>XML tag</th>
<th>Attribute</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>status_report</td>
<td></td>
<td>Yes</td>
<td>This tag is used to report a status</td>
</tr>
<tr>
<td>type_id</td>
<td></td>
<td>Yes</td>
<td>The type of status. See Table 2.56</td>
</tr>
<tr>
<td>code</td>
<td></td>
<td>Yes</td>
<td>The code for the status</td>
</tr>
<tr>
<td>level</td>
<td></td>
<td>Yes</td>
<td>The level of the status. See Table 2.57</td>
</tr>
<tr>
<td>data</td>
<td></td>
<td>No</td>
<td>Optimal data for the status</td>
</tr>
<tr>
<td>status_get_level</td>
<td></td>
<td>Yes</td>
<td>This tag is used to request the status level from a device</td>
</tr>
<tr>
<td>status_set_level</td>
<td></td>
<td>Yes</td>
<td>This tag is used to set the status level on a device</td>
</tr>
<tr>
<td>status_report_level</td>
<td></td>
<td>Yes</td>
<td>This tag is used to report the status level from a device</td>
</tr>
</tbody>
</table>

Table 2.55: Status tags

The valid types for the status is described in Table 2.56.

<table>
<thead>
<tr>
<th>Id</th>
<th>Type</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Public key</td>
</tr>
<tr>
<td>2</td>
<td>Memory Information</td>
</tr>
<tr>
<td>3</td>
<td>Device Description</td>
</tr>
<tr>
<td>4</td>
<td>Value Description</td>
</tr>
<tr>
<td>5</td>
<td>Value</td>
</tr>
<tr>
<td>6</td>
<td>Partner Information</td>
</tr>
<tr>
<td>7</td>
<td>Action</td>
</tr>
<tr>
<td>8</td>
<td>Calculation</td>
</tr>
<tr>
<td>9</td>
<td>Timer</td>
</tr>
<tr>
<td>10</td>
<td>Calendar</td>
</tr>
<tr>
<td>11</td>
<td>State machine</td>
</tr>
<tr>
<td>12</td>
<td>Firmware update</td>
</tr>
<tr>
<td>13</td>
<td>Configuration</td>
</tr>
<tr>
<td>100</td>
<td>Exi</td>
</tr>
<tr>
<td>101</td>
<td>System</td>
</tr>
<tr>
<td>200</td>
<td>Application</td>
</tr>
</tbody>
</table>

Table 2.56: Status type

The valid level for the status level is described in Table 2.57.

<table>
<thead>
<tr>
<th>Level</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Disabled</td>
<td>No status reports will be send</td>
</tr>
<tr>
<td>1</td>
<td>Fatal</td>
<td>Fatal errors that can’t be recovered from</td>
</tr>
<tr>
<td>2</td>
<td>Error</td>
<td>An error</td>
</tr>
<tr>
<td>3</td>
<td>Warning</td>
<td>A warning</td>
</tr>
<tr>
<td>4</td>
<td>Info</td>
<td>Very verbose information</td>
</tr>
<tr>
<td>5</td>
<td>Debug</td>
<td>Debug information</td>
</tr>
</tbody>
</table>

Table 2.57: Status level
The device description code is described in Table 2.58.

<table>
<thead>
<tr>
<th>Code</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>Set wrong ID</td>
<td>Trying to set a read only value</td>
</tr>
<tr>
<td>12</td>
<td>Wrong size of Channel Map</td>
<td>The channel map must have 4 channels</td>
</tr>
<tr>
<td>13</td>
<td>Missing Synchrozations channels</td>
<td>The channel map is missing the synchrozations channels</td>
</tr>
</tbody>
</table>

Table 2.58: Status device description code

The value description code is described in Table 2.59.

<table>
<thead>
<tr>
<th>Code</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Get Wrong ID</td>
<td>Trying to get a value description with a invalid id</td>
</tr>
<tr>
<td>2</td>
<td>Set Wrong ID</td>
<td>Trying to add a virtual value description with a invalid id</td>
</tr>
<tr>
<td>3</td>
<td>Delete Wrong ID</td>
<td>Trying to delete a virtual value with a invalid id</td>
</tr>
<tr>
<td>11</td>
<td>Not Supported</td>
<td>Trying to add a virtual value with a type that is not supported</td>
</tr>
<tr>
<td>12</td>
<td>Invalid Step</td>
<td>Trying to add a virtual value with a invalid step</td>
</tr>
</tbody>
</table>

Table 2.59: Status value description code

The value code is described in Table 2.60.

<table>
<thead>
<tr>
<th>Code</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Get Wrong ID</td>
<td>Trying to get a value with a invalid id</td>
</tr>
<tr>
<td>2</td>
<td>Set Wrong ID</td>
<td>Trying to set a value with a invalid id</td>
</tr>
<tr>
<td>3</td>
<td>Check Wrong ID</td>
<td>Trying to use a value with a wrong id</td>
</tr>
<tr>
<td>11</td>
<td>Value Invalid</td>
<td>The value is invalid</td>
</tr>
<tr>
<td>12</td>
<td>Wrong Data Type</td>
<td>Trying to set a value with a wrong data type</td>
</tr>
<tr>
<td>13</td>
<td>Invalid Step</td>
<td>The step is not supported</td>
</tr>
<tr>
<td>14</td>
<td>Can’t read write only</td>
<td>Trying to read a write only value</td>
</tr>
<tr>
<td>15</td>
<td>Can’t write read only</td>
<td>Trying to write a read only value</td>
</tr>
</tbody>
</table>

Table 2.60: Status value code

The partner information code is described in Table 2.61.

<table>
<thead>
<tr>
<th>Code</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Get wrong Id</td>
<td>Trying to get a partner that is not configured</td>
</tr>
<tr>
<td>2</td>
<td>Set wrong Id</td>
<td>Trying to set a partner with a wrong Id</td>
</tr>
<tr>
<td>3</td>
<td>Delete wrong Id</td>
<td>Trying to delete a partner that is not configured</td>
</tr>
<tr>
<td>11</td>
<td>Wrong Id</td>
<td>The reference partner is not configured</td>
</tr>
<tr>
<td>12</td>
<td>Failed to send to partner</td>
<td>Sending a message to a partner but there was no reply</td>
</tr>
<tr>
<td>13</td>
<td>Set wrong type</td>
<td>Trying set a partner with an unsupported info type</td>
</tr>
</tbody>
</table>

Table 2.61: Status partner information code

The action code is described in Table 2.62.

<table>
<thead>
<tr>
<th>Code</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Get wrong Id</td>
<td>Trying to get an action that is not configured</td>
</tr>
<tr>
<td>2</td>
<td>Set wrong Id</td>
<td>Trying to set an action with an invalid id</td>
</tr>
<tr>
<td>3</td>
<td>Delete wrong Id</td>
<td>Trying to delete an action that is not configured</td>
</tr>
<tr>
<td>11</td>
<td>Execute wrong Id</td>
<td>Trying to execute an action that is not configured</td>
</tr>
<tr>
<td>12</td>
<td>Execute queue overflow</td>
<td>Trying to enqueue action, but queue is full</td>
</tr>
</tbody>
</table>

Table 2.62: Status action code
The calculation code is described in Table 2.63.

<table>
<thead>
<tr>
<th>Code</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Get wrong Id</td>
<td>Trying to get an calculation that is not configured</td>
</tr>
<tr>
<td>2</td>
<td>Set wrong Id</td>
<td>Trying to set an calculation with an invalid id</td>
</tr>
<tr>
<td>3</td>
<td>Delete wrong Id</td>
<td>Trying to delete an calculation that is not configured</td>
</tr>
<tr>
<td>11</td>
<td>Check wrong Id</td>
<td>Trying to evaluate a calculation that is not configured</td>
</tr>
</tbody>
</table>

Table 2.63: Status calculation code

The timer code is described in Table 2.64.

<table>
<thead>
<tr>
<th>Code</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Get wrong Id</td>
<td>Trying to get a timer that is not configured</td>
</tr>
<tr>
<td>2</td>
<td>Set wrong Id</td>
<td>Trying to set a timer with a wrong id</td>
</tr>
<tr>
<td>3</td>
<td>Delete wrong Id</td>
<td>Trying to delete a timer that is not configured</td>
</tr>
<tr>
<td>11</td>
<td>Start wrong Id</td>
<td>Trying to start a timer that is not configured</td>
</tr>
<tr>
<td>12</td>
<td>Stop wrong Id</td>
<td>Trying to stop a timer that is not configured</td>
</tr>
</tbody>
</table>

Table 2.64: Status timer code

The calendar code is described in Table 2.65.

<table>
<thead>
<tr>
<th>Code</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Get wrong Id</td>
<td>Trying to get a calendar task that is not configured</td>
</tr>
<tr>
<td>2</td>
<td>Set wrong Id</td>
<td>Trying to set a calendar task with a wrong id</td>
</tr>
<tr>
<td>3</td>
<td>Delete wrong Id</td>
<td>Trying to delete a calendar task that is not configured</td>
</tr>
</tbody>
</table>

Table 2.65: Status calendar code

The state machine code is described in Table 2.66.

<table>
<thead>
<tr>
<th>Code</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Get wrong Id</td>
<td>Trying to get a statemachine with an invalid id</td>
</tr>
<tr>
<td>2</td>
<td>Set wrong Id</td>
<td>Trying to set a statemachine with an invalid id</td>
</tr>
<tr>
<td>3</td>
<td>Delete wrong Id</td>
<td>Trying to delete a statemachine with an invalid id</td>
</tr>
<tr>
<td>4</td>
<td>Get wrong state Id</td>
<td>Trying to get a statemachine state with an invalid id</td>
</tr>
<tr>
<td>5</td>
<td>Set wrong state Id</td>
<td>Trying to set a statemachine state with an invalid id</td>
</tr>
<tr>
<td>6</td>
<td>Delete wrong state Id</td>
<td>Trying to delete a statemachine state with an invalid id</td>
</tr>
<tr>
<td>11</td>
<td>Check wrong id</td>
<td>Trying to get a statemachine state with invalid id</td>
</tr>
<tr>
<td>12</td>
<td>Running too long</td>
<td>The execution of the statemachine was been running too long</td>
</tr>
</tbody>
</table>

Table 2.66: Status state machine code

The firmware update code is described in Table 2.67.

<table>
<thead>
<tr>
<th>Code</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>Failed to upgrade</td>
<td>Failed to upgrade firmware from bootloader</td>
</tr>
</tbody>
</table>

Table 2.67: Status firmware update code

The configuration code is described in Table 2.68.

<table>
<thead>
<tr>
<th>Code</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>Timeout</td>
<td>The started configuration timed out and the configuration is rolled back</td>
</tr>
<tr>
<td>12</td>
<td>Invalid</td>
<td>The configuration saved on the device is invalid and needs to be validated</td>
</tr>
<tr>
<td>13</td>
<td>Started</td>
<td>The configuration has started on the device and the statemachine is stopped</td>
</tr>
</tbody>
</table>

Table 2.68: Status configuration code

The system code is described in Table 2.69.
<table>
<thead>
<tr>
<th>Code</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>11</td>
<td>No NTP</td>
<td>Failed to synchronize with the NTP Server</td>
</tr>
<tr>
<td>12</td>
<td>Awake</td>
<td>Device was been woken and is now awake</td>
</tr>
</tbody>
</table>

Table 2.69: Status system code
2.3.16.2 Examples

This flowering section consists of examples in xml for how to use the status_report, status_get_level, status_set_level and status_report_level message.

Status report

To report status, a status_report message has to be sent from the device. See XML 2.92 for an example of the report message.

```xml
<? xml version="1.0" encoding="UTF-8" ?>
< network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
 xsi:noNamespaceSchemaLocation="status.xsd">
 < device version="1">
   < status_report type_id="7" code="1" level="2"/>
 </ device>
</ network >
```

XML 2.92: Status report

Status get level

In order to get the status level, a status_get_level message has to be sent from the device. See XML 2.93 for an example of the get level message.

```xml
<? xml version="1.0" encoding="UTF-8" ?>
< network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
 xsi:noNamespaceSchemaLocation="status.xsd">
 < device version="1">
   < status_get_level/>
 </ device>
</ network >
```

XML 2.93: Status get level

Status report level

To report the status level, a status_report_level message has to be sent from the device. See XML 2.94 for an example of the report level message.

```xml
<? xml version="1.0" encoding="UTF-8" ?>
< network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
 xsi:noNamespaceSchemaLocation="status.xsd">
 < device version="1">
   < status_report_level level="2"/>
 </ device>
</ network >
```

XML 2.94: Status report level
Status set level

In order to set the status level, a `status_set_level` message has to be sent to the device. See XML 2.95 for an example of the `set level` message.

```xml
<?xml version="1.0" encoding="UTF-8"?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
    xsi:noNamespaceSchemaLocation="status.xsd">
    <device version="1">
        <status_set_level level="3"/>
    </device>
</network>
```

XML 2.95: Set status level
2.3.17 Configuration

The configuration service is used to save the configuration on a device or discard it.

2.3.17.1 Tag description

The message tag for the configuration is described in Table 2.70.

<table>
<thead>
<tr>
<th>XML tag</th>
<th>Attribute</th>
<th>Required</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>config_status_get</td>
<td>-</td>
<td>Yes</td>
<td>This tag is used to request the configuration status from a device</td>
</tr>
<tr>
<td>config_status_report</td>
<td>-</td>
<td>Yes</td>
<td>This tag is used to report the configuration status</td>
</tr>
<tr>
<td>status</td>
<td>-</td>
<td>Yes</td>
<td>The configuration status. See Table 2.71</td>
</tr>
<tr>
<td>config_mode_set</td>
<td>-</td>
<td>Yes</td>
<td>This tag is used to set the configuration mode on a device</td>
</tr>
<tr>
<td>mode</td>
<td>-</td>
<td>Yes</td>
<td>The configuration mode. See Table 2.72</td>
</tr>
</tbody>
</table>

Table 2.70: Configuration tags

The valid values for the configuration status is described in Table 2.71.

<table>
<thead>
<tr>
<th>Status</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Idle</td>
<td>When there is not configuration started</td>
</tr>
<tr>
<td>1</td>
<td>Started</td>
<td>When configuration is started</td>
</tr>
</tbody>
</table>

Table 2.71: Configuration Status

The valid configuration modes is described in Table 2.72.

<table>
<thead>
<tr>
<th>Status</th>
<th>Name</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>Rollback</td>
<td>Rollback any configuration changes</td>
</tr>
<tr>
<td>1</td>
<td>Save and Reset</td>
<td>Save any configuration changes and reset the statemachine states</td>
</tr>
<tr>
<td>1</td>
<td>Save and Preserve</td>
<td>Save any configuration changes and do not change the statemachine states</td>
</tr>
<tr>
<td>1</td>
<td>Set Default</td>
<td>Clear the configuration and set the default configuration</td>
</tr>
<tr>
<td>1</td>
<td>Clear</td>
<td>Clear the configuration</td>
</tr>
</tbody>
</table>

Table 2.72: Configuration Mode
2.3.17.2 Examples

This flowering section consists of examples in xml for how to use the `configuration_status_get`, `configuration_status_report` and `configuration_mode_set` message.

Get configuration status

In order to get the configuration status, a `configuration_status_get` message has to be sent to the device. See XML 2.96 for an example of the get message.

```
<? xml version="1.0" encoding="UTF-8"?>
< network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
          xsi:noNamespaceSchemaLocation="configuration.xsd">
  < device version="1">
    < config_status_get/>
  </ device>
</ network>
```

XML 2.96: Get configuration status

Report configuration status

To report the configuration status, a `configuration_status_report` message has to be sent from the device. See XML 2.97 for an example of the report message.

```
<? xml version="1.0" encoding="UTF-8"?>
< network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
          xsi:noNamespaceSchemaLocation="configuration.xsd">
  < device version="1">
    < config_status_report status="0"/>
  </ device>
</ network>
```

XML 2.97: Report configuration status

Set configuration mode

In order to set the the configuration mode, a `configuration_mode_set` message has to be sent to the device. See XML 2.98 for an example of the mode message.

```
<? xml version="1.0" encoding="UTF-8"?>
< network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
          xsi:noNamespaceSchemaLocation="configuration.xsd">
  < device version="1">
    < config_mode_set mode="1"/>
  </ device>
</ network>
```

XML 2.98: Set configuration mode
2.4 User stories

In this chapter the different user stories will be described. In every section there are descriptions of the user stories a drawing and examples of how to describe the scenario in xml.

2.4.1 Remote Control with wall switch controlling multiple devices

An individual from the household comes home from work and wants to turn on both the light and the TV.

He/she pushes the wall switch (Wall switch) connected to the electrical outlets of both the light and the TV, which subsequently turns both these appliances on.

Later that evening, when he/she is getting ready for bed, he/she again pushes the wall switch, which in turn shuts off all appliances connected to this switch, i.e. the lights and the TV.

2.4.1.1 Illustration

Figure 2.5: Remote control with light and TV
2.4.1.2 XML

XML 2.99: Calculation for User Story 1

<?xml version="1.0" encoding="UTF-8"?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
xsi:noNamespaceSchemaLocation="calculation.xsd">
  <device version="1">
    <calculation_report>
      <calculation calculation_id="1" method_id="6">
        <left value_id="1"/>
        <right constant_number="1"/>
      </calculation>
      <calculation calculation_id="1" method_id="6">
        <left value_id="1"/>
        <right constant_number="0"/>
      </calculation>
    </calculation_report>
  </device>
</network>

XML 2.100: Action for User Story 1

<?xml version="1.0" encoding="UTF-8"?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
xsi:noNamespaceSchemaLocation="action.xsd">
  <device version="1">
    <action_report>
      <action action_id="1">
        <set value_id="1" partner_id="1" number="0"/>
        <set value_id="1" partner_id="2" number="0"/>
      </action>
      <action action_id="2">
        <set value_id="1" partner_id="1" number="1"/>
        <set value_id="1" partner_id="2" number="1"/>
      </action>
    </action_report>
  </device>
</network>

XML 2.101: State Machine for User Story 1

<?xml version="1.0" encoding="UTF-8"?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
xsi:noNamespaceSchemaLocation="statemachine.xsd">
  <device version="1">
    <statemachine_report>
      <statemachine statemachine_id="1">
        <state state_id="1">
          <transaction calculation_id="1" action_id="1" goto_state_id="2"/>
        </state>
        <state state_id="2">
          <transaction calculation_id="2" action_id="2" goto_state_id="1"/>
        </state>
      </statemachine>
    </statemachine_report>
  </device>
</network>
2.4.2 Temperature Control with calendar tasks

The temperature control in the house is set to 16 degrees Celsius at night-time.

An individual wakes up at 6:00 every morning, and wants the house to have an ambient temperature of 19 degrees Celsius. Later when he/she leaves for work at 8:00, the temperature should return to a steady temperature of 16 degrees Celsius.

In the late afternoon, when he/she comes home at 18:00, the temperature should be at an ambient temperature of 21 degrees Celsius, followed by a decrease in temperature steadying at 16 degrees when going to bed at 22:00 in the evening.

2.4.2.1 Illustration

![Temperature Control Graph](image)

Figure 2.6: Temperature control with calendar tasks
### 2.4.2.2 XML

XML 2.102: Action for User Story 2

```xml
<?xml version="1.0" encoding="UTF-8"?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
    xsi:noNamespaceSchemaLocation="action.xsd">
    <device version="1">
        <action_report>
            <action action_id="1">
                <set value_id="1" number="16"/>
            </action>
            <action action_id="2">
                <set value_id="1" number="19"/>
            </action>
            <action action_id="3">
                <set value_id="1" number="21"/>
            </action>
        </action_report>
    </device>
</network>
```

XML 2.103: Calendar for User Story 2

```xml
<?xml version="1.0" encoding="UTF-8"?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
    xsi:noNamespaceSchemaLocation="calendar.xsd">
    <device version="1">
        <calendar_report>
            <task task_id="1" start="1319432400" repeat="86400"/>
            <task task_id="2" start="1319443200" repeat="86400"/>
            <task task_id="3" start="1319475600" repeat="86400"/>
            <task task_id="4" start="1319493600" repeat="86400"/>
        </calendar_report>
    </device>
</network>
```
2.4.3 Temperature Control with window and temperature sensor

The temperature in the house is at a steady 21 degrees Celsius.

An individual from the household opens a window in a room, which is detected by the system. The thermostat detects a possible decrease in room temperature, but allows the temperature to drop because of the opened windows.

After 5 minutes he/she closes the window again, and the temperature should rise and steady at the before mentioned 21 degrees Celsius.

2.4.3.1 Illustration

![Figure 2.7: Temperature control with window and temperature sensor](image-url)
2.4.3.2 XML

<?xml version="1.0" encoding="UTF-8"?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
xsi:noNamespaceSchemaLocation="calculation.xsd">
  <device version="1">
    <calculation_report>
      <calculation calculation_id="1" method_id="9">
        <!-- Partner 2 is temperature -->
        <left value_id="1" partner_id="2"/>
        <right constant_number="21"/>
      </calculation>
      <calculation calculation_id="2" method_id="8">
        <left value_id="1" partner_id="2"/>
        <right constant_number="20"/>
      </calculation>
      <calculation calculation_id="3" method_id="6">
        <!-- Partner 3 is window -->
        <left value_id="1" partner_id="3"/>
        <right constant_number="0"/>
      </calculation>
      <calculation calculation_id="4" method_id="6">
        <left value_id="1" partner_id="3"/>
        <right constant_number="1"/>
      </calculation>
    </calculation_report>
  </device>
</network>

XML 2.104: Calculation for User Story 3

<?xml version="1.0" encoding="UTF-8"?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
xsi:noNamespaceSchemaLocation="action.xsd">
  <device version="1">
    <action_report>
      <action action_id="1">
        <set value_id="1" number="1"/>
      </action>
      <action action_id="2">
        <set value_id="1" number="0"/>
      </action>
    </action_report>
  </device>
</network>

XML 2.105: Action for User Story 3
XML 2.106: State Machine for User Story 3
2.4.4 Light control with movement and luminance sensor

At day-time when it is light outside, an individual walks into the living room. This movement is detected by the system, but the lights are not turned on as it is not necessary.

Later that evening, when it has turned dark outside, he/she walks into the living room again, turning the lights on in this room.

He/she leaves the living room, and 5 minutes after the lights are turned off again, due to the system not picking up any movement in the room for the 5 minute duration.

2.4.4.1 Illustration

![Diagram illustrating light control with movement and luminance sensor](image)

Figure 2.8: Light control with movement and luminance sensor
2.4.4.2 XML

XML 2.107: Calculation for User Story 4

```xml
<?xml version="1.0" encoding="UTF-8"?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
    xsi:noNamespaceSchemaLocation="calculation.xsd">
    <device version="1">
        <calculation calculation_id="1" method_id="6">
            <!-- Partner 2 is movement -->
            <left value_id="1" partner_id="2"/>
            <right constant_number="1"/>
        </calculation>
        <calculation calculation_id="2" method_id="8">
            <!-- Partner 3 is luminance -->
            <left value_id="1" partner_id="3"/>
            <right constant_number="50"/>
        </calculation>
        <calculation calculation_id="3" method_id="9">
            <left value_id="1" partner_id="3"/>
            <right constant_number="75"/>
        </calculation>
        <calculation calculation_id="4" method_id="6">
            <left timer_id="1"/>
            <right constant_number="1"/>
        </calculation>
    </device>
</network>
```

XML 2.108: Action for User Story 4

```xml
<?xml version="1.0" encoding="UTF-8"?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
    xsi:noNamespaceSchemaLocation="action.xsd">
    <device version="1">
        <action_report>
            <action action_id="1">
                <set value_id="1" number="0"/>
            </action>
            <action action_id="2">
                <set value_id="1" number="0"/>
            </action>
            <action action_id="3">
                <timer_start timer_id="1"/>
            </action>
        </action_report>
    </device>
</network>
```
<?xml version="1.0" encoding="UTF-8"?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
xsi:noNamespaceSchemaLocation="statemachine.xsd">
  <device version="1">
    <statemachine_report>
      <statemachine statemachine_id="1">
        <state state_id="1">
          <transaction calculation_id="1" action_id="3" goto_state_id="2"/>
          <transaction calculation_id="2" goto_state_id="3"/>
        </state>
        <state state_id="2">
          <transaction calculation_id="2" action_id="1" goto_state_id="4"/>
          <transaction calculation_id="4" goto_state_id="1"/>
        </state>
        <state state_id="3">
          <transaction calculation_id="3" goto_state_id="1"/>
          <transaction calculation_id="1" action_id="1" goto_state_id="4"/>
        </state>
        <state state_id="4">
          <transaction calculation_id="3" action_id="2" goto_state_id="1"/>
          <transaction calculation_id="1" goto_state_id="3"/>
          <transaction calculation_id="4" action_id="2" goto_state_id="1"/>
        </state>
      </statemachine>
    </statemachine_report>
  </device>
</network>

XML 2.109: State Machine for User Story 4

<?xml version="1.0" encoding="UTF-8"?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
xsi:noNamespaceSchemaLocation="timer.xsd">
  <device version="1">
    <timer_report>
      <execute timer_id="1" after="300000"/>
    </timer_report>
  </device>
</network>

XML 2.110: Timer for User Story 4
2.4.5 Washing machine control

An individual prepares the washing machine by putting in the clothing and applying laundry detergent and possibly fabric softener if needed.

He/she then leaves for work, but wants the washing machine to be turned on, so that it will have finished its laundry cycle when he/she comes home from work.

He/she sets the washing program to wool, and tells the system that he/she will be home at 18:00. The system determines the best time for the washing machine to start for it to be finished when the user comes home from work.

Later that day, he/she finds that they will be coming home earlier and starts the washing machine manually, thereby overruling the starting time set by the system.

2.4.5.1 Illustration

![Diagram of washing machine control](image.png)

Figure 2.9: Washing machine control
2.4.5.2 XML

XML 2.111: Value for User Story 5

<?xml version="1.0" encoding="UTF-8"?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
    xsi:noNamespaceSchemaLocation="value.xsd">
    <device version="1">
        <value_set value_id="1" timestamp="0" string="wool"/>
        <value_get value_id="2"/>
        <value_report value_id="2" timestamp="0" string="01:45:00"/>
        <value_set value_id="1" timestamp="0" string="start"/>
    </device>
</network>

XML 2.112: Action for User Story 5

<?xml version="1.0" encoding="UTF-8"?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
    xsi:noNamespaceSchemaLocation="action.xsd">
    <device version="1">
        <action_report>
            <action action_id="1">
                <set value_id="3" string="start"/>
            </action>
        </action_report>
    </device>
</network>

XML 2.113: Calendar for User Story 5

<?xml version="1.0" encoding="UTF-8"?>
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
    xsi:noNamespaceSchemaLocation="calendar.xsd">
    <device version="1">
        <calendar_report>
            <task task_id="1" start="1319472900" action_id="1"/>
        </calendar_report>
    </device>
</network>
2.4.6 Electricity pricing

The provider of electricity to the household, i.e. the electricity company, sends pricings to the electric meter on a regular daily basis.

This information is used to reduce the electricity bill. This reduction is done by performing actions that have a larger cost, in periods with low pricing, e.g. at night-time or mid-day on workdays.

2.4.6.1 Illustration

![Electricity price chart]

Figure 2.10: Electricity pricing
### 2.4.6.2 XML

XML 2.114: Action for User Story 6

```xml
<?xml version="1.0" encoding="UTF-8"?
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
xsi:noNamespaceSchemaLocation="action.xsd">
  <device version="1">
    <action_report>
      <action action_id="1">
        <set value_id="1" number="0.2"/>
      </action>
      <action action_id="2">
        <set value_id="1" number="0.4"/>
      </action>
      <action action_id="3">
        <set value_id="1" number="0.6"/>
      </action>
      <action action_id="4">
        <set value_id="1" number="0.7"/>
      </action>
    </action_report>
  </device>
</network>
```

XML 2.115: Calendar for User Story 6

```xml
<?xml version="1.0" encoding="UTF-8"?
<network xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" version="1"
xsi:noNamespaceSchemaLocation="calendar.xsd">
  <device version="1">
    <calendar_report>
      <task task_id="1" start="1319436000" action_id="3" end="1319446800" repeat="86400"/>
      <task task_id="2" start="1319446800" action_id="2" end="1319472000" repeat="86400"/>
      <task task_id="3" start="1319472000" action_id="4" end="13194972000" repeat="86400"/>
      <task task_id="4" start="13194972000" action_id="1" end="1319522400" repeat="86400"/>
    </calendar_report>
  </device>
</network>
```
List of XSD’s

```xml
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
  <xs:simpleType name="payloadType">
    <xs:restriction base="xs:hexBinary">
      <xs:minLength value="0"/>
      <xs:maxLength value="2047"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:element name="phy">
    <xs:complexType>
      <xs:attribute name="payload" type="payloadType" use="required"/>
      <xs:attribute name="phy_layer_version" type="xs:unsignedInt" use="required"/>
      <xs:attribute name="security" type="xs:unsignedInt" use="required"/>
      <xs:attribute name="foward_error_correction" type="xs:unsignedInt" use="required"/>
      <xs:attribute name="foward_error_correction_length" type="xs:unsignedInt" use="optional"/>
    </xs:complexType>
  </xs:element>
</xs:schema>
```
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
  <xs:simpleType name="frame_nonceType">
    <xs:restriction base="xs:hexBinary">
      <xs:length value="6"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="mac_source_addressType">
    <xs:restriction base="xs:hexBinary">
      <xs:minLength value="2"/>
      <xs:maxLength value="17"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="frame_integrity_codeType">
    <xs:restriction base="xs:hexBinary">
      <xs:length value="4"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="mac_destination_adressType">
    <xs:restriction base="xs:hexBinary">
      <xs:minLength value="2"/>
      <xs:maxLength value="17"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:simpleType name="mapType">
    <xs:restriction base="xs:hexBinary">
      <xs:length value="4"/>
    </xs:restriction>
  </xs:simpleType>
  <xs:complexType name="mac_option_ack_requestType">
    <xs:attribute name="nr_retries" type="xs:unsignedInt" use="required"/>
  </xs:complexType>
  <xs:complexType name="mac_option_ackType">
    <xs:attribute name="rssi" type="xs:unsignedInt" use="required"/>
  </xs:complexType>
  <xs:complexType name="mac_option_fragmentType">
    <xs:attribute name="is_last" type="xs:unsignedInt" use="required"/>
    <xs:attribute name="offset" type="xs:unsignedInt" use="required"/>
  </xs:complexType>
  <xs:complexType name="mac_option_channel_mapType">
    <xs:attribute name="type" type="xs:unsignedInt" use="required"/>
    <xs:attribute name="map" type="mapType" use="required"/>
  </xs:complexType>
  <xs:complexType name="mac_option_wake_on_radioType">
    <xs:attribute name="timestamp" type="xs:unsignedInt" use="required"/>
    <xs:attribute name="fraction" type="xs:unsignedInt" use="required"/>
    <xs:attribute name="interval" type="xs:unsignedInt" use="required"/>
    <xs:attribute name="channel" type="xs:unsignedInt" use="required"/>
  </xs:complexType>
  <xs:complexType name="mac">
    <xs:element name="mac"/>
  </xs:complexType>
</xs:schema>
<xs:element name="mac_option_ack_request" type="mac_option_ack_requestType" minOccurs="1" maxOccurs="1" />
<xs:element name="mac_option_ack" type="mac_option_ackType" minOccurs="1" maxOccurs="1" />
<xs:element name="mac_option_fragment" type="mac_option_fragmentType" minOccurs="0" maxOccurs="1" />
<xs:element name="mac_option_channel_map" type="mac_option_channel_mapType" minOccurs="0" maxOccurs="unbounded" />
<xs:element name="mac_option_wake_on_radio" type="mac_option_wake_on_radioType" minOccurs="1" maxOccurs="1" />
</xs:sequence>
<xs:attribute name="mac_layer_version" type="xs:unsignedInt" use="required"/>
<xs:attribute name="frame_nonce" type="frame_nonceType" use="required"/>
<xs:attribute name="mac_source_address" type="mac_source_addressType" use="required"/>
<xs:attribute name="frame_integrity_code" type="frame_integrity_codeType" use="required"/>
<xs:attribute name="mac_destination_address" type="mac_destination_addressType" use="required"/>
<xs:attribute name="content_type" type="xs:unsignedInt" use="required"/>
</xs:complexType>
</xs:element>
</xs:schema>
<? xml version=\"1.0\" encoding=\"UTF-8\" ?>
<xs:schema xmlns:xs=\"http://www.w3.org/2001/XMLSchema\" elementFormDefault=\"qualified\">
  <xs:complexType name=\"network_include\">
    <xs:simpleContent>
      <xs:extension base=\"xs:hexBinary\">
        <xs:attribute name=\"address_size\" type=\"xs:unsignedByte\" use=\"optional\"/>
        <xs:attribute name=\"inclusion_count\" type=\"xs:unsignedInt\" use=\"optional\"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
  <xs:element name=\"network\">
    <xs:complexType>
      <xs:sequence>
        <xs:element name=\"device\" minOccurs=\"1\" maxOccurs=\"unbounded\">
          <xs:complexType>
            <xs:choice minOccurs=\"1\" maxOccurs=\"1\">
              <xs:element name=\"network_include\" type=\"network_include\"/>
              <xs:attribute name=\"version\" type=\"xs:unsignedInt\" use=\"required\"/>
              <xs:attribute name=\"device_id\" type=\"xs:unsignedInt\" use=\"optional\"/>
              <xs:attribute name=\"go_to_sleep\" type=\"xs:unsignedInt\" use=\"optional\"/>
            </xs:choice>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name=\"version\" type=\"xs:unsignedInt\" use=\"required\"/>
    </xs:complexType>
  </xs:element>
</xs:schema>

XML 119: Network Management XSD
XML 121: Service Description XSD
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
  <xs:complexType name="memoryInformationGetType">
  </xs:complexType>
  <xs:complexType name="memoryInformationType">
    <xs:attribute name="memory_id" type="xs:unsignedInt" use="required"/>
    <xs:attribute name="count" type="xs:unsignedInt" use="required"/>
    <xs:attribute name="free_count" type="xs:unsignedInt" use="required"/>
  </xs:complexType>
  <xs:complexType name="memoryInformationReportType">
    <xs:sequence>
      <xs:element name="memory_information" type="memoryInformationType" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <xs:element name="network">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="device" minOccurs="1" maxOccurs="unbounded">
          <xs:complexType>
            <xs:choice minOccurs="0" maxOccurs="unbounded">
              <xs:element name="memory_information_get" type="memoryInformationGetType"/>
              <xs:element name="memory_information_report" type="memoryInformationReportType"/>
            </xs:choice>
            <xs:attribute name="version" type="xs:unsignedInt" use="required"/>
            <xs:attribute name="device_id" type="xs:unsignedInt" use="optional"/>
            <xs:attribute name="go_to_sleep" type="xs:unsignedInt" use="optional"/>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="version" type="xs:unsignedInt" use="required"/>
    </xs:complexType>
  </xs:element>
</xs:schema>

XML 122: Memory Information XSD
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
  <xs:complexType name="infoType">
    <xs:attribute name="type_id" type="xs:unsignedInt" use="required"/>
    <xs:attribute name="number" type="xs:unsignedLong" use="optional"/>
    <xs:attribute name="string" type="xs:string" use="optional"/>
    <xs:attribute name="hex" type="xs:hexBinary" use="optional"/>
  </xs:complexType>
  <xs:complexType name="deviceDescriptionGetType">
    <xs:sequence minOccurs="1" maxOccurs="unbounded">
      <xs:element name="info" type="infoType"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="deviceDescriptionType">
    <xs:sequence minOccurs="1" maxOccurs="unbounded">
      <xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:element name="device_description_get" type="deviceDescriptionGetType"/>
        <xs:element name="device_description_report" type="deviceDescriptionType"/>
        <xs:element name="device_description_set" type="deviceDescriptionType"/>
      </xs:choice>
      <xs:attribute name="version" type="xs:unsignedInt" use="required"/>
      <xs:attribute name="device_id" type="xs:unsignedInt" use="optional"/>
      <xs:attribute name="go_to_sleep" type="xs:unsignedInt" use="optional"/>
    </xs:sequence>
  </xs:complexType>
</xs:complexType>
</xs:schema>

XML 123: Device Description XSD
<?xml version="1.0" encoding="UTF-8"?>

<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
  <xs:complexType name="numberFormatType">
    <xs:attribute name="unit" type="xs:string" use="required"/>
    <xs:attribute name="min" type="xs:double" use="required"/>
    <xs:attribute name="max" type="xs:double" use="required"/>
    <xs:attribute name="step" type="xs:double" use="required"/>
  </xs:complexType>
  <xs:complexType name="stringFormatType">
    <xs:sequence>
      <xs:element name="valid_value" type="xs:string" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="max_length" type="xs:unsignedInt" use="required"/>
  </xs:complexType>
  <xs:complexType name="hexBinaryFormatType">
    <xs:attribute name="max_length" type="xs:unsignedInt" use="required"/>
  </xs:complexType>
  <xs:complexType name="valueDescriptionGetType">
    <xs:attribute name="value_description_id" type="xs:unsignedInt" use="optional"/>
    <xs:sequence>
      <xs:element name="number_format" type="numberFormatType"/>
      <xs:element name="string_format" type="stringFormatType"/>
      <xs:element name="hexBinary_format" type="hexBinaryFormatType"/>
    </xs:sequence>
    <xs:attribute name="value_id" type="xs:unsignedInt" use="required"/>
    <xs:attribute name="type_id" type="xs:unsignedInt" use="required"/>
    <xs:attribute name="mode" type="xs:unsignedInt" use="required"/>
    <xs:attribute name="persistent" type="xs:unsignedInt" use="required"/>
    <xs:attribute name="name" type="xs:string" use="optional"/>
    <xs:attribute name="min_log_interval" type="xs:unsignedInt" use="optional"/>
    <xs:attribute name="max_log_values" type="xs:unsignedInt" use="optional"/>
    <xs:attribute name="virtual" type="xs:unsignedInt" use="optional"/>
  </xs:complexType>
  <xs:complexType name="valueDescriptionReportType">
    <xs:sequence>
      <xs:element name="value_description" type="valueDescriptionType" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="valueDescriptionAddType">
    <xs:sequence>
      <xs:element name="value_description" type="valueDescriptionType" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="valueDescriptionDeleteType">
    <xs:attribute name="value_description_id" type="xs:unsignedInt" use="optional"/>
  </xs:complexType>
  <xs:complexType name="valueDescriptionMemoryGetType">
  </xs:complexType>
</xs:schema>
<xs:element name="network">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="device" minOccurs="1" maxOccurs="unbounded">
        <xs:complexType>
          <xs:choice minOccurs="0" maxOccurs="unbounded">
            <xs:element name="value_description_get" type="valueDescriptionGetType"/>
            <xs:element name="value_description_report" type="valueDescriptionReportType"/>
            <xs:element name="value_description_add" type="valueDescriptionAddType"/>
            <xs:element name="value_description_delete" type="valueDescriptionDeleteType"/>
            <xs:element name="value_description_get_memory" type="valueDescriptionMemoryGetType"/>
            <xs:element name="value_description_report_memory" type="valueDescriptionMemoryReportType"/>
          </xs:choice>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
    <xs:attribute name="version" type="xs:unsignedInt" use="required"/>
    <xs:attribute name="device_id" type="xs:unsignedInt" use="optional"/>
    <xs:attribute name="go_to_sleep" type="xs:unsignedInt" use="optional"/>
  </xs:complexType>
</xs:element>
"
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
  <xs:complexType name="valueGetType">
    <xs:attribute name="value_id" type="xs:unsignedInt" use="optional"/>
  </xs:complexType>
  <xs:complexType name="valueReportType">
    <xs:attribute name="timestamp" type="xs:unsignedLong" use="required"/>
    <xs:attribute name="value_id" type="xs:unsignedInt" use="optional"/>
    <xs:attribute name="number" type="xs:double" use="optional"/>
    <xs:attribute name="string" type="xs:string" use="optional"/>
    <xs:attribute name="hexBinary" type="xs:hexBinary" use="optional"/>
  </xs:complexType>
  <xs:complexType name="valueSetType">
    <xs:attribute name="value_id" type="xs:unsignedInt" use="required"/>
    <xs:attribute name="timestamp" type="xs:unsignedLong" use="required"/>
    <xs:attribute name="number" type="xs:double" use="optional"/>
    <xs:attribute name="string" type="xs:string" use="optional"/>
    <xs:attribute name="hexBinary" type="xs:hexBinary" use="optional"/>
  </xs:complexType>
  <xs:complexType name="valueGetLogType">
    <xs:attribute name="value_id" type="xs:unsignedInt" use="optional"/>
    <xs:attribute name="start_time" type="xs:unsignedLong" use="optional"/>
    <xs:attribute name="log_count" type="xs:unsignedInt" use="optional"/>
  </xs:complexType>
  <xs:complexType name="network">
    <xs:attribute name="version" type="xs:unsignedInt" use="required"/>
    <xs:attribute name="device_id" type="xs:unsignedInt" use="optional"/>
    <xs:attribute name="go_to_sleep" type="xs:unsignedInt" use="optional"/>
  </xs:complexType>
  <xs:element name="network" type="network">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="device" minOccurs="1" maxOccurs="unbounded">
          <xs:complexType>
            <xs:choice minOccurs="0" maxOccurs="unbounded">
              <xs:element name="value_get" type="valueGetType"/>
              <xs:element name="value_report" type="valueReportType"/>
              <xs:element name="value_set" type="valueSetType"/>
              <xs:element name="value_get_log" type="valueGetLogType"/>
            </xs:choice>
            <xs:attribute name="version" type="xs:unsignedInt" use="required"/>
            <xs:attribute name="device_id" type="xs:unsignedInt" use="optional"/>
            <xs:attribute name="go_to_sleep" type="xs:unsignedInt" use="optional"/>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
  <xs:complexType name="infoType">
    <xs:attribute name="type_id" type="xs:unsignedInt" use="required"/>
    <xs:attribute name="number" type="xs:unsignedLong" use="optional"/>
    <xs:attribute name="string" type="xs:string" use="optional"/>
    <xs:attribute name="hex" type="xs:hexBinary" use="optional"/>
  </xs:complexType>
  <xs:complexType name="partnerType">
    <xs:sequence minOccurs="1" maxOccurs="unbounded">
      <xs:element name="info" type="infoType"/>
    </xs:sequence>
    <xs:attribute name="partner_id" type="xs:unsignedInt" use="required"/>
  </xs:complexType>
  <xs:complexType name="groupPartnerType">
    <xs:attribute name="partner_id" type="xs:unsignedInt" use="required"/>
  </xs:complexType>
  <xs:complexType name="groupType">
    <xs:sequence minOccurs="1" maxOccurs="unbounded">
      <xs:element name="partner" type="groupPartnerType"/>
      <xs:element name="group" type="groupType"/>
    </xs:sequence>
    <xs:attribute name="partner_id" type="xs:unsignedInt" use="required"/>
  </xs:complexType>
  <xs:complexType name="partnerInformationGetType">
    <xs:attribute name="partner_id" type="xs:unsignedInt" use="optional"/>
  </xs:complexType>
  <xs:complexType name="partnerInformationReportType">
    <xs:choice>
      <xs:sequence minOccurs="0" maxOccurs="unbounded">
        <xs:element name="partner" type="partnerType"/>
        <xs:element name="group" type="groupType"/>
      </xs:sequence>
    </xs:choice>
  </xs:complexType>
  <xs:complexType name="partnerInformationSetType">
    <xs:choice>
      <xs:sequence minOccurs="1" maxOccurs="unbounded">
        <xs:element name="partner" type="partnerType"/>
        <xs:element name="group" type="groupType"/>
      </xs:sequence>
    </xs:choice>
  </xs:complexType>
  <xs:complexType name="partnerInformationDeleteType">
    <xs:attribute name="partner_id" type="xs:unsignedInt" use="optional"/>
  </xs:complexType>
  <xs:complexType name="partnerInformationMemoryGetType">
  </xs:complexType>
  <xs:complexType name="partnerInformationMemoryReportType">
  </xs:complexType>
</xs:schema>
<xs:complexType name="network">
  <xs:sequence>
    <xs:element name="device" minOccurs="1" maxOccurs="unbounded">
      <xs:complexType>
        <xs:choice minOccurs="0" maxOccurs="unbounded">
          <xs:element name="partner_information_get" type="partnerInformationGetType"/>
          <xs:element name="partner_information_report" type="partnerInformationReportType"/>
          <xs:element name="partner_information_set" type="partnerInformationSetType"/>
          <xs:element name="partner_information_delete" type="partnerInformationDeleteType"/>
          <xs:element name="partner_information_get_memory" type="partnerInformationMemoryGetType"/>
          <xs:element name="partner_information_report_memory" type="partnerInformationMemoryReportType"/>
        </xs:choice>
      </xs:complexType>
    </xs:element>
  </xs:sequence>
</xs:complexType>
</xs:element>
</xs:complexType>
</xs:attribute name="version" type="xs:unsignedInt" use="required"/>
</xs:complexType>
</xs:element>
</xs:sequence>
<xs:attribute name="version" type="xs:unsignedInt" use="required"/>
</xs:complexType>
</xs:element>
</xs:schema>

XML 128: Partner Information XSD 2/2
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
  <xs:complexType name="getType">
    <xs:attribute name="value_id" type="xs:unsignedInt" use="required"/>
    <xs:attribute name="partner_id" type="xs:unsignedInt" use="optional"/>
    <xs:attribute name="transport_mode" type="xs:unsignedInt" use="optional"/>
  </xs:complexType>
  <xs:complexType name="setType">
    <xs:attribute name="value_id" type="xs:unsignedInt" use="required"/>
    <xs:attribute name="number" type="xs:double" use="optional"/>
    <xs:attribute name="string" type="xs:string" use="optional"/>
    <xs:attribute name="hexBinary" type="xs:hexBinary" use="optional"/>
    <xs:attribute name="partner_id" type="xs:unsignedInt" use="optional"/>
    <xs:attribute name="calculation_id" type="xs:unsignedInt" use="optional"/>
    <xs:attribute name="transport_mode" type="xs:unsignedInt" use="optional"/>
  </xs:complexType>
  <xs:complexType name="reportType">
    <xs:attribute name="my_value_id" type="xs:unsignedInt" use="required"/>
    <xs:attribute name="partner_id" type="xs:unsignedInt" use="optional"/>
    <xs:attribute name="transport_mode" type="xs:unsignedInt" use="optional"/>
  </xs:complexType>
  <xs:complexType name="invokeType">
    <xs:attribute name="action_id" type="xs:unsignedInt" use="required"/>
    <xs:attribute name="partner_id" type="xs:unsignedInt" use="required"/>
  </xs:complexType>
  <xs:complexType name="timerType">
    <xs:attribute name="timer_id" type="xs:unsignedInt" use="required"/>
  </xs:complexType>
  <xs:complexType name="actionType">
    <xs:choice minOccurs="1" maxOccurs="unbounded">
      <xs:element name="get" type="getType"/>
      <xs:element name="set" type="setType"/>
      <xs:element name="report" type="reportType"/>
      <xs:element name="invoke" type="invokeType"/>
      <xs:element name="timer_start" type="timerType"/>
      <xs:element name="timer_stop" type="timerType"/>
    </xs:choice>
    <xs:attribute name="action_id" type="xs:unsignedInt" use="optional"/>
  </xs:complexType>
  <xs:complexType name="actionGetType">
    <xs:attribute name="action_id" type="xs:unsignedInt" use="optional"/>
  </xs:complexType>
  <xs:complexType name="actionReportType">
    <xs:sequence minOccurs="0" maxOccurs="unbounded">
      <xs:element name="action" type="actionType"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="actionSetType">
    <xs:sequence minOccurs="1" maxOccurs="unbounded">
      <xs:element name="action" type="actionType"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="actionDeleteType">
    <xs:attribute name="action_id" type="xs:unsignedInt" use="optional"/>
  </xs:complexType>
</xs:schema>
<xs:complexType name="actionInvokeType">
  <xs:attribute name="action_id" type="xs:unsignedInt" use="required"/>
</xs:complexType>

<xs:complexType name="actionMemoryGetType">
</xs:complexType>

<xs:complexType name="actionMemoryReportType">
  <xs:attribute name="count" type="xs:unsignedInt" use="required"/>
  <xs:attribute name="free_count" type="xs:unsignedInt" use="required"/>
</xs:complexType>

<xs:element name="network">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="device" minOccurs="1" maxOccurs="unbounded">
        <xs:complexType>
          <xs:choice minOccurs="0" maxOccurs="unbounded">
            <xs:element name="action_get" type="actionGetType"/>
            <xs:element name="action_report" type="actionReportType"/>
            <xs:element name="action_set" type="actionSetType"/>
            <xs:element name="action_delete" type="actionDeleteType"/>
            <xs:element name="action_invoke" type="actionInvokeType"/>
            <xs:element name="action_get_memory" type="actionMemoryGetType"/>
            <xs:element name="action_report_memory" type="actionMemoryReportType"/>
          </xs:choice>
          <xs:attribute name="version" type="xs:unsignedInt" use="required"/>
          <xs:attribute name="device_id" type="xs:unsignedInt" use="optional"/>
          <xs:attribute name="go_to_sleep" type="xs:unsignedInt" use="optional"/>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>
</xs:element>

XML 130: Action XSD 2/2
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
  <xs:complexType name="calcSubType">
    <xs:attribute name="value_id" type="xs:unsignedInt" use="optional"/>
    <xs:attribute name="calculation_id" type="xs:unsignedInt" use="optional"/>
    <xs:attribute name="partner_id" type="xs:unsignedInt" use="optional"/>
    <xs:attribute name="statemachine_id" type="xs:unsignedInt" use="optional"/>
    <xs:attribute name="timer_id" type="xs:unsignedInt" use="optional"/>
    <xs:attribute name="calendar_id" type="xs:unsignedInt" use="optional"/>
    <xs:attribute name="is_updated" type="xs:unsignedByte" use="optional"/>
    <xs:attribute name="constant_number" type="xs:double" use="optional"/>
    <xs:attribute name="constant_string" type="xs:string" use="optional"/>
    <xs:attribute name="constant_hexBinary" type="xs:hexBinary" use="optional"/>
  </xs:complexType>
  <xs:complexType name="calculationGetType">
    <xs:attribute name="calculation_id" type="xs:unsignedInt" use="optional"/>
  </xs:complexType>
  <xs:complexType name="calculationReportType">
    <xs:sequence>
      <xs:element name="calculation" type="calculationType" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="calculationSetType">
    <xs:sequence>
      <xs:element name="calculation" type="calculationType" minOccurs="1" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>
  <xs:complexType name="calculationDeleteType">
    <xs:attribute name="calculation_id" type="xs:unsignedInt" use="optional"/>
  </xs:complexType>
  <xs:complexType name="calculationMemoryGetType">
    <xs:attribute name="count" type="xs:unsignedInt" use="required"/>
    <xs:attribute name="free_count" type="xs:unsignedInt" use="required"/>
  </xs:complexType>
  <xs:complexType name="calculationMemoryReportType">
    <xs:attribute name="count" type="xs:unsignedInt" use="required"/>
    <xs:attribute name="free_count" type="xs:unsignedInt" use="required"/>
  </xs:complexType>
  <xs:element name="network">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="device" minOccurs="1" maxOccurs="unbounded">
          <xs:complexType>
            <xs:choice minOccurs="0" maxOccurs="unbounded">
              <xs:element name="calculation_get" type="calculationGetType"/>
            </xs:choice>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
<xs:element name="calculation_set" type="calculationSetType"/>
<xs:element name="calculation_delete" type="calculationDeleteType"/>
<xs:element name="calculation_get_memory" type="calculationMemoryGetType"/>
<xs:element name="calculation_report_memory" type="calculationMemoryReportType"/>
</xs:choice>
<xs:attribute name="version" type="xs:unsignedInt" use="required"/>
<xs:attribute name="device_id" type="xs:unsignedInt" use="optional"/>
<xs:attribute name="go_to_sleep" type="xs:unsignedInt" use="optional"/>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:attribute name="version" type="xs:unsignedInt" use="required"/>
</xs:complexType>
</xs:element>
</xs:schema>
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
  <xs:complexType name="executeType">
    <xs:attribute name="timer_id" type="xs:unsignedInt" use="required"/>
    <xs:attribute name="after" type="xs:unsignedInt" use="required"/>
    <xs:attribute name="calculation_id" type="xs:unsignedInt" use="optional"/>
    <xs:attribute name="action_id" type="xs:unsignedInt" use="optional"/>
  </xs:complexType>

  <xs:complexType name="timerGetType">
    <xs:attribute name="timer_id" type="xs:unsignedInt" use="optional"/>
  </xs:complexType>

  <xs:complexType name="timerReportType">
    <xs:element name="execute" type="executeType" minOccurs="0" maxOccurs="unbounded"/>
  </xs:complexType>

  <xs:complexType name="timerSetType">
    <xs:element name="execute" type="executeType" minOccurs="1" maxOccurs="unbounded"/>
  </xs:complexType>

  <xs:complexType name="timerDeleteType">
    <xs:attribute name="timer_id" type="xs:unsignedInt" use="optional"/>
  </xs:complexType>

  <xs:complexType name="timerMemoryGetType">
    <xs:attribute name="count" type="xs:unsignedInt" use="required"/>
    <xs:attribute name="free_count" type="xs:unsignedInt" use="required"/>
  </xs:complexType>

  <xs:complexType name="timerMemoryReportType">
    <xs:element name="network" minOccurs="1" maxOccurs="unbounded">
      <xs:complexType>
        <xs:choice minOccurs="0" maxOccurs="unbounded">
          <xs:element name="timer_get" type="timerGetType"/>
          <xs:element name="timer_set" type="timerSetType"/>
          <xs:element name="timer_delete" type="timerDeleteType"/>
          <xs:element name="timer_get_memory" type="timerMemoryGetType"/>
          <xs:element name="timer_report_memory" type="timerMemoryReportType"/>
          <xs:attribute name="version" type="xs:unsignedInt" use="required"/>
          <xs:attribute name="device_id" type="xs:unsignedInt" use="optional"/>
          <xs:attribute name="go_to_sleep" type="xs:unsignedInt" use="optional"/>
        </xs:choice>
      </xs:complexType>
    </xs:element>
  </xs:complexType>
</xs:schema>
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
<xs:complexType name="taskType">
  <xs:attribute name="task_id" type="xs:unsignedInt" use="required"/>
  <xs:attribute name="start" type="xs:unsignedLong" use="required"/>
  <xs:attribute name="action_id" type="xs:unsignedInt" use="optional"/>
  <xs:attribute name="end" type="xs:unsignedLong" use="optional"/>
  <xs:attribute name="repeat" type="xs:unsignedInt" use="optional"/>
  <xs:attribute name="weekdays" type="xs:unsignedByte" use="optional"/>
</xs:complexType>
<xs:complexType name="calendarGetType">
  <xs:attribute name="task_id" type="xs:unsignedInt" use="optional"/>
</xs:complexType>
<xs:complexType name="calendarReportType">
  <xs:sequence>
    <xs:element name="task" type="taskType" minOccurs="0" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="calendarSetType">
  <xs:sequence>
    <xs:element name="task" type="taskType" minOccurs="1" maxOccurs="unbounded"/>
  </xs:sequence>
</xs:complexType>
<xs:complexType name="calendarDeleteType">
</xs:complexType>
<xs:complexType name="calendarTimezoneGetType">
</xs:complexType>
<xs:complexType name="calendarTimezoneSetType">
  <xs:attribute name="offset" type="xs:int" use="required"/>
</xs:complexType>
<xs:complexType name="calendarTimezoneReportType">
  <xs:attribute name="offset" type="xs:int" use="required"/>
</xs:complexType>
<xs:complexType name="calendarMemoryGetType">
</xs:complexType>
<xs:complexType name="calendarMemoryReportType">
  <xs:attribute name="count" type="xs:unsignedInt" use="required"/>
  <xs:attribute name="free_count" type="xs:unsignedInt" use="required"/>
</xs:complexType>
<xs:element name="network">
  <xs:complexType>
    <xs:sequence>
      <xs:element name="device" minOccurs="1" maxOccurs="unbounded"/>
      <xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:element name="calendar_get" type="calendarGetType"/>
        <xs:element name="calendar_report" type="calendarReportType"/>
        <xs:element name="calendar_set" type="calendarSetType"/>
      </xs:choice>
    </xs:sequence>
  </xs:complexType>
</xs:element>
</xs:complexType>
</xs:schema>
<xs:element name="calendar_get_timezone" type="calendarTimezoneGetType"/>
<xs:element name="calendar_set_timezone" type="calendarTimezoneSetType"/>
<xs:element name="calendar_report_timezone" type="calendarTimezoneReportType"/>
<xs:element name="calendar_get_memory" type="calendarMemoryGetType"/>
<xs:element name="calendar_report_memory" type="calendarMemoryReportType"/>
</xs:choice>
<xs:attribute name="version" type="xs:unsignedInt" use="required"/>
<xs:attribute name="device_id" type="xs:unsignedInt" use="optional"/>
<xs:attribute name="go_to_sleep" type="xs:unsignedInt" use="optional"/>
</xs:complexType>
</xs:element>
</xs:sequence>
</xs:complexType>
</xs:element>
</xs:schema>

XML 136: Calendar XSD 2/2
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
  <xs:complexType name="transactionType">
    <xs:attribute name="calculation_id" type="xs:unsignedInt" use="optional"/>
    <xs:attribute name="action_id" type="xs:unsignedInt" use="optional"/>
    <xs:attribute name="goto_state_id" type="xs:unsignedInt" use="optional"/>
  </xs:complexType>

  <xs:complexType name="stateType">
    <xs:sequence>
      <xs:element name="transaction" type="transactionType" minOccurs="1" maxOccurs="unbounded"/>
    </xs:sequence>
    <xs:attribute name="state_id" type="xs:unsignedInt" use="required"/>
  </xs:complexType>

  <xs:complexType name="statemachineGetType">
    <xs:attribute name="statemachine_id" type="xs:unsignedInt" use="optional"/>
    <xs:attribute name="state_id" type="xs:unsignedInt" use="optional"/>
  </xs:complexType>

  <xs:complexType name="statemachineType">
    <xs:choice minOccurs="1" maxOccurs="unbounded">
      <xs:element name="state" type="stateType"/>
    </xs:choice>
    <xs:attribute name="statemachine_id" type="xs:unsignedInt" use="required"/>
  </xs:complexType>

  <xs:complexType name="statemachineReportType">
    <xs:sequence>
      <xs:element name="statemachine" type="statemachineType" minOccurs="0" maxOccurs="unbounded"/>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="statemachineSetType">
    <xs:sequence>
      <xs:choice minOccurs="1" maxOccurs="unbounded">
        <xs:element name="statemachine" type="statemachineType" minOccurs="1" maxOccurs="unbounded"/>
      </xs:choice>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="statemachineDeleteType">
    <xs:attribute name="statemachine_id" type="xs:unsignedInt" use="optional"/>
    <xs:attribute name="state_id" type="xs:unsignedInt" use="optional"/>
  </xs:complexType>

  <xs:complexType name="statemachineStateType">
    <xs:simpleContent>
      <xs:extension base="xs:unsignedInt">
        <xs:attribute name="statemachine_id" type="xs:unsignedInt" use="required"/>
      </xs:extension>
    </xs:simpleContent>
  </xs:complexType>
</xs:schema>
<xs:complexType name="statemachineGetStateType">
    <xs:attribute name="statemachine_id" type="xs:unsignedInt" use="optional"/>
</xs:complexType>

<xs:complexType name="statemachineSetStateType">
    <xs:element name="statemachine_state" type="statemachineStateType" minOccurs="1" maxOccurs="unbounded"/>
</xs:complexType>

<xs:complexType name="statemachineReportStateType">
    <xs:element name="statemachine_state" type="statemachineStateType" minOccurs="0" maxOccurs="unbounded"/>
</xs:complexType>

<xs:complexType name="statemachineMemoryGetType">
</xs:complexType>

<xs:complexType name="statemachineMemoryReportType">
    <xs:attribute name="count" type="xs:unsignedInt" use="required"/>
    <xs:attribute name="free_count" type="xs:unsignedInt" use="required"/>
</xs:complexType>

<xs:element name="network">
    <xs:complexType>
        <xs:sequence>
            <xs:element name="device" minOccurs="1" maxOccurs="unbounded">
                <xs:choice minOccurs="0" maxOccurs="unbounded">
                    <xs:element name="statemachine_get" type="statemachineGetType"/>
                    <xs:element name="statemachine_report" type="statemachineReportType"/>
                    <xs:element name="statemachine_set" type="statemachineSetType"/>
                    <xs:element name="statemachine_delete" type="statemachineDeleteType"/>
                    <xs:element name="statemachine_get_state" type="statemachineGetStateType"/>
                    <xs:element name="statemachine_report_state" type="statemachineReportStateType"/>
                    <xs:element name="statemachine_set_state" type="statemachineSetStateType"/>
                    <xs:element name="statemachine_get_memory" type="statemachineMemoryGetType"/>
                    <xs:element name="statemachine_report_memory" type="statemachineMemoryReportType"/>
                    <xs:element name="statemachine_report_state_memory" type="statemachineMemoryReportType"/>
                </xs:choice>
            </xs:element>
        </xs:sequence>
    </xs:complexType>
</xs:element>
</xs:complexType>

XML 138: State Machine XSD 2/2
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
  <xs:complexType name="firmwareInitType">
    <xs:attribute name="size" type="xs:unsignedInt" use="required"/>
    <xs:attribute name="checksum" type="xs:hexBinary" use="required"/>
    <xs:attribute name="firmware_id" type="xs:unsignedInt" use="required"/>
  </xs:complexType>
  <xs:complexType name="firmwareDataType">
    <xs:sequence>
      <xs:element name="chunk" type="xs:hexBinary" minOccurs="1" maxOccurs="1"/>
    </xs:sequence>
    <xs:attribute name="offset" type="xs:unsignedInt" use="required"/>
  </xs:complexType>
  <xs:complexType name="firmwareUpdateStartType"/>
  <xs:complexType name="firmwareReportType">
    <xs:attribute name="status" type="xs:unsignedInt" use="required"/>
    <xs:attribute name="expected_offset" type="xs:unsignedInt" use="optional"/>
  </xs:complexType>
  <xs:complexType name="firmwareInformationGetType">
  </xs:complexType>
  <xs:complexType name="firmwareInformationReportType">
    <xs:attribute name="size" type="xs:unsignedInt" use="required"/>
    <xs:attribute name="firmware_id" type="xs:unsignedInt" use="required"/>
    <xs:attribute name="received_size" type="xs:unsignedInt" use="required"/>
    <xs:attribute name="chunk_size" type="xs:unsignedInt" use="required"/>
  </xs:complexType>
  <xs:element name="network"/>
</xs:schema>

XML 139: Firmware Update XSD 1/2

XML 140: Firmware Update XSD 2/2
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
  <xs:complexType name="configStatusGetType">
  </xs:complexType>
  <xs:complexType name="configStatusReportType">
    <xs:attribute name="status" type="xs:unsignedInt" use="required"/>
  </xs:complexType>
  <xs:complexType name="configModeSetType">
    <xs:attribute name="mode" type="xs:unsignedInt" use="required"/>
  </xs:complexType>
  <xs:element name="network">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="device" minOccurs="1" maxOccurs="unbounded">
          <xs:complexType>
            <xs:choice minOccurs="0" maxOccurs="unbounded">
              <xs:element name="config_status_report" type="configStatusReportType"/>
              <xs:element name="config_status_get" type="configStatusGetType"/>
              <xs:element name="config_mode_set" type="configModeSetType"/>
            </xs:choice>
            <xs:attribute name="version" type="xs:unsignedInt" use="required"/>
            <xs:attribute name="device_id" type="xs:unsignedInt" use="optional"/>
            <xs:attribute name="go_to_sleep" type="xs:unsignedInt" use="optional"/>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
<?xml version="1.0" encoding="UTF-8"?>
<xs:schema xmlns:xs="http://www.w3.org/2001/XMLSchema" elementFormDefault="qualified">
  <xs:complexType name="statusReportType">
    <xs:attribute name="type_id" type="xs:unsignedInt" use="required"/>
    <xs:attribute name="code" type="xs:unsignedInt" use="required"/>
    <xs:attribute name="level" type="xs:unsignedInt" use="required"/>
    <xs:attribute name="data" type="xs:hexBinary" use="optional"/>
  </xs:complexType>
  <xs:complexType name="statusGetLevelType">
  </xs:complexType>
  <xs:complexType name="statusSetLevelType">
    <xs:attribute name="level" type="xs:unsignedInt" use="required"/>
  </xs:complexType>
  <xs:complexType name="statusReportLevelType">
    <xs:attribute name="level" type="xs:unsignedInt" use="required"/>
  </xs:complexType>
  <!-- Main Element -->
  <xs:element name="network">
    <xs:complexType>
      <xs:sequence>
        <xs:element name="device" minOccurs="1" maxOccurs="unbounded">
          <xs:complexType>
            <xs:choice minOccurs="0" maxOccurs="unbounded">
              <xs:element name="status_report" type="statusReportType"/>
              <xs:element name="status_get_level" type="statusGetLevelType"/>
              <xs:element name="status_set_level" type="statusSetLevelType"/>
              <xs:element name="status_report_level" type="statusReportLevelType"/>
            </xs:choice>
            <xs:attribute name="version" type="xs:unsignedInt" use="required"/>
            <xs:attribute name="device_id" type="xs:unsignedInt" use="optional"/>
            <xs:attribute name="go_to_sleep" type="xs:unsignedInt" use="optional"/>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
      <xs:attribute name="version" type="xs:unsignedInt" use="required"/>
    </xs:complexType>
  </xs:element>
</xs:schema>
# List of Tables

<table>
<thead>
<tr>
<th>Section</th>
<th>Title</th>
<th>Page</th>
</tr>
</thead>
<tbody>
<tr>
<td>2.1</td>
<td>NTP Timestamp</td>
<td>7</td>
</tr>
<tr>
<td>2.2</td>
<td>NTP Frame</td>
<td>7</td>
</tr>
<tr>
<td>2.3</td>
<td>List of port numbers</td>
<td>10</td>
</tr>
<tr>
<td>2.4</td>
<td>The format of the inclusion message</td>
<td>10</td>
</tr>
<tr>
<td>2.5</td>
<td>Special Virtual Values</td>
<td>12</td>
</tr>
<tr>
<td>2.6</td>
<td>Network and device tags</td>
<td>16</td>
</tr>
<tr>
<td>2.7</td>
<td>Network management tags</td>
<td>17</td>
</tr>
<tr>
<td>2.8</td>
<td>The values used in the example</td>
<td>17</td>
</tr>
<tr>
<td>2.9</td>
<td>Public key tags</td>
<td>19</td>
</tr>
<tr>
<td>2.10</td>
<td>Public key types</td>
<td>19</td>
</tr>
<tr>
<td>2.11</td>
<td>Service description tags</td>
<td>21</td>
</tr>
<tr>
<td>2.12</td>
<td>Service description message child tags</td>
<td>21</td>
</tr>
<tr>
<td>2.13</td>
<td>Service Description Types</td>
<td>21</td>
</tr>
<tr>
<td>2.14</td>
<td>Memory description tags</td>
<td>23</td>
</tr>
<tr>
<td>2.15</td>
<td>Memory_information_report child tags</td>
<td>23</td>
</tr>
<tr>
<td>2.16</td>
<td>Memory Types</td>
<td>23</td>
</tr>
<tr>
<td>2.17</td>
<td>Device Description tags</td>
<td>25</td>
</tr>
<tr>
<td>2.18</td>
<td>Device Description child tags</td>
<td>25</td>
</tr>
<tr>
<td>2.19</td>
<td>Device Description Types. Types in <strong>bold</strong> are settable</td>
<td>25</td>
</tr>
<tr>
<td>2.20</td>
<td>Protocols</td>
<td>26</td>
</tr>
<tr>
<td>2.21</td>
<td>Radio Mode</td>
<td>26</td>
</tr>
<tr>
<td>2.22</td>
<td>Manufactures</td>
<td>26</td>
</tr>
<tr>
<td>2.23</td>
<td>SGTIN-96</td>
<td>26</td>
</tr>
<tr>
<td>2.24</td>
<td>SGTIN-96 Partition Values</td>
<td>27</td>
</tr>
<tr>
<td>2.25</td>
<td>Value Description tags</td>
<td>30</td>
</tr>
<tr>
<td>2.26</td>
<td>Value_description_report message child tags</td>
<td>30</td>
</tr>
<tr>
<td>2.27</td>
<td>Value description child tags</td>
<td>31</td>
</tr>
<tr>
<td>2.28</td>
<td>Value description String format child tags</td>
<td>31</td>
</tr>
<tr>
<td>2.29</td>
<td>Value description types</td>
<td>32</td>
</tr>
<tr>
<td>2.30</td>
<td>Value tags</td>
<td>36</td>
</tr>
<tr>
<td>2.31</td>
<td>Partner tags</td>
<td>40</td>
</tr>
<tr>
<td>2.32</td>
<td>Partner information child tags</td>
<td>40</td>
</tr>
<tr>
<td>2.33</td>
<td>Partner Information partner child tags</td>
<td>41</td>
</tr>
<tr>
<td>2.34</td>
<td>Device Description Types. Types in <strong>bold</strong> are settable</td>
<td>41</td>
</tr>
<tr>
<td>2.35</td>
<td>Group child tags</td>
<td>41</td>
</tr>
<tr>
<td>2.36</td>
<td>Action tags</td>
<td>45</td>
</tr>
<tr>
<td>2.37</td>
<td>Action_report and action_set child tags</td>
<td>45</td>
</tr>
<tr>
<td>2.38</td>
<td>Action child tags</td>
<td>46</td>
</tr>
<tr>
<td>2.39</td>
<td>Transport mode</td>
<td>46</td>
</tr>
<tr>
<td>2.40</td>
<td>Calculation tags</td>
<td>50</td>
</tr>
<tr>
<td>2.41</td>
<td>Calculation message child tags</td>
<td>50</td>
</tr>
<tr>
<td>2.42</td>
<td>Calculation child tags</td>
<td>51</td>
</tr>
<tr>
<td>2.43</td>
<td>Method types</td>
<td>51</td>
</tr>
<tr>
<td>2.44</td>
<td>Timer tags</td>
<td>55</td>
</tr>
<tr>
<td>2.45</td>
<td>Timer message child tags</td>
<td>55</td>
</tr>
<tr>
<td>2.46</td>
<td>Calendar tags</td>
<td>59</td>
</tr>
</tbody>
</table>
2.47 Calendar child tags ................................................................. 59
2.48 State machine tags ................................................................. 64
2.49 State machine message child tags ........................................... 65
2.50 Statemachine child tags .......................................................... 65
2.51 State child tags ................................................................. 65
2.52 Firmware update tags ............................................................. 71
2.53 Firmware_data message child tags .......................................... 71
2.54 Firmware Update Status Type ................................................ 71
2.55 Status tags ................................................................. 74
2.56 Status type ................................................................. 74
2.57 Status level ................................................................. 74
2.58 Status device description code ............................................ 75
2.59 Status value description code ................................................ 75
2.60 Status value code ............................................................. 75
2.61 Status partner information code .............................................. 75
2.62 Status action code ............................................................. 75
2.63 Status calculation code .......................................................... 76
2.64 Status timer code .............................................................. 76
2.65 Status calendar code ............................................................ 76
2.66 Status state machine code .......................................................... 76
2.67 Status firmware update code .................................................. 76
2.68 Status configuration code .......................................................... 76
2.69 Status system code .............................................................. 77
2.70 Configuration tags ............................................................. 80
2.71 Configuration Status ........................................................... 80
2.72 Configuration Mode ............................................................... 80
List of XMLs

2.1 XML to EXI compression XML version .......................................................... 9
2.2 XML to EXI compression EXI version (values in hexadecimal notation) ................. 9
2.3 Encapsulation using network and device tags .................................................. 16
2.4 Network controller key ...................................................................................... 18
2.5 Get a public key .................................................................................................. 20
2.6 Report a public key .............................................................................................. 20
2.7 Get a service description .................................................................................... 22
2.8 Report a service description ................................................................................. 22
2.9 Get the memory information .............................................................................. 24
2.10 Report the memory information ........................................................................ 24
2.11 Get the device description .................................................................................. 28
2.12 Report of the device description ........................................................................ 28
2.13 Set the device description property ..................................................................... 29
2.14 Get the value description with value description ID 1 ........................................ 33
2.15 Get all value descriptions ................................................................................... 33
2.16 Report of the value description for number format ............................................. 33
2.17 Report of the value description for string format ............................................... 34
2.18 Add virtual value description .............................................................................. 34
2.19 Delete virtual value description .......................................................................... 34
2.20 Delete virtual value description .......................................................................... 35
2.21 Get the value with ID 1 ....................................................................................... 37
2.22 Get all values ...................................................................................................... 37
2.23 Report of the value with ID 1 has the value 55 ................................................... 37
2.24 Report of the value with ID 2 has the value ON ................................................ 38
2.25 Report of all the values ....................................................................................... 38
2.26 Set the value with ID 1 to 55 .............................................................................. 38
2.27 Set the value with ID 2 to ON ............................................................................ 38
2.28 Set the value with ID 1 to 55 and the value with ID 2 to ON ............................. 39
2.29 Get the log for the value with ID 1 ..................................................................... 39
2.30 Report all the log values for the value with ID 1 and 2 ........................................ 39
2.31 Get the partner with the ID 1 ............................................................................. 42
2.32 Get all partners .................................................................................................. 42
2.33 Report for the partner with ID 1 ......................................................................... 42
2.34 Report all partners .............................................................................................. 43
2.35 Set the partner with ID 1 ..................................................................................... 43
2.36 Set multiple partners .......................................................................................... 44
2.37 Delete the partners with ID 1 ............................................................................. 44
2.38 Delete all partners .............................................................................................. 44
2.39 Get the action with ID 1 ..................................................................................... 47
2.40 Get all actions .................................................................................................... 47
2.41 Report the set action with ID 2 .......................................................................... 47
2.42 Report all actions ............................................................................................... 48
2.43 Set the action with ID 1 ..................................................................................... 48
2.44 Set multiple actions ............................................................................................ 49
2.45 Delete the action with ID 5 ................................................................................. 49
2.46 Delete all actions ............................................................................................... 49
2.47 Get the calculation with ID 1 ............................................................................. 52
2.48 Get all calculations ................................. 52
2.49 Report the calculation multiply-method ................................. 52
2.50 Report all calculations .................................. 53
2.51 Set the calculation add-method ................................. 53
2.52 Set multiple calculations .................................. 54
2.53 Delete the calculation with ID 1 ................................. 54
2.54 Delete all calculations........................................ 54
2.55 Get the timer with ID 1 .................................. 56
2.56 Get all the timers .................................. 56
2.57 Report for the timer with ID 1 that will execute action 2 after 2000 ms ................................. 56
2.58 Report for the timer with ID 2 that will execute after 4000 ms .................................. 57
2.59 Report all timers .................................. 57
2.60 Set the timer with ID 2 to execute action 3 after 2000 ms .................................. 57
2.61 Set the timers with ID 1 and 2 .................................. 58
2.62 Delete the timer with ID 1 .................................. 58
2.63 Delete all the timers .................................. 58
2.64 Get the calendar task with task ID 1 .................................. 60
2.65 Get all the calendar tasks .................................. 60
2.66 Report the calendar task with ID 1 .................................. 60
2.67 Report all calendar tasks .................................. 61
2.68 Set the calendar task with ID 1 .................................. 61
2.69 Set multiple calendar tasks .................................. 61
2.70 Delete the calendar task with ID 1 .................................. 62
2.71 Delete all calendar tasks .................................. 62
2.72 Get the state machine with ID 1 and its state with ID 2 .................................. 66
2.73 Get the complete state machine with ID 1 .................................. 66
2.74 Get all state machines .................................. 66
2.75 Report the state machine with ID 1 and its state with ID 2 .................................. 67
2.76 Report the state machines with ID 1 and all its states .................................. 67
2.77 Report the state machines with ID 1 and the state machine with ID 2 and all there states .................................. 67
2.78 Set the state machine with ID 1 state 1 .................................. 68
2.79 Set multiple states in the state machine with ID 1 .................................. 68
2.80 Set multiple states in the state machine with ID 1 and 2 .................................. 69
2.81 Delete the state with ID 2 from the state machine with ID 1 .................................. 69
2.82 Delete the state machines with ID 1 .................................. 69
2.83 Delete all state machines .................................. 69
2.84 Get the current state of the state machine .................................. 70
2.85 Report the current state of the state machine .................................. 70
2.86 Initialize the firmware .................................. 72
2.87 A chunk of the firmware image .................................. 72
2.88 Start firmware update .................................. 72
2.89 Firmware report .................................. 73
2.90 Get the firmware information .................................. 73
2.91 Firmware information report .................................. 73
2.92 Status report .................................. 78
2.93 Status get level .................................. 78
2.94 Status report level .................................. 78
2.95 Set status level .................................. 79
2.96 Get configuration status .................................. 81
2.97 Report configuration status .................................. 81
2.98 Set configuration mode .................................. 81
2.99 Calculation for User Story 1 .................................. 83
2.100 Action for User Story 1 .................................. 83
2.101 State Machine for User Story 1 .................................. 83
2.102 Action for User Story 2 .................................. 85
2.103 Calendar for User Story 2 .................................. 85
2.104 Calculation for User Story 3 .................................. 87
2.105 Action for User Story 3 .................................. 87
List of Figures

2.1 Inclusion ................................................................. 11
2.2 Exclusion ................................................................. 11
2.3 Action enqueue ......................................................... 13
2.4 State machine executing ............................................. 14
2.5 Remote control with light and TV .................................. 82
2.6 Temperature control with calendar tasks .......................... 84
2.7 Temperature control with window and temperature sensor ...... 86
2.8 Light control with movement and luminance sensor ............ 89
2.9 Washing machine control .............................................. 92
2.10 Electricity pricing ....................................................... 94
List of Corrections

Note: Add example for service get memory .......................................................... 24
Note: Add is_updated example ........................................................................... 54