System Specification

Communication Profiles Certification

V 1.4

Approved for first release:

Approved for second release:

Approved for final release:

San Ramon, CA, USA, 2015

Executive Summary

Ver.	Editor	Change	Date
0.1	DM	Document created, EEP added	Sept 01, 2014
0.2	MHö	Content reviewed, GP, Recom added	Oct 30, 2014
1.0	AP	Content reviewed, applying Alliance template, EEP direction added	Nov 18, 2014
1.1.	MH	Added Testing Strategy chapter.	Dec 12, 2014
1.11	MH	Review by TR	Jan 16, 2015
1.2	TM	Content review, new XML format added, eep and gp chapter redone. Chapter Reman and Recom redone	Feb 17, 2015
1.3	AP	Complete rework after EnOcean GmbH internal review	May 6,2015
1.31	AP	Implement review comments of TWG review	July 22,2015
1.4	LC	Native speaker rework	July 23, 2015

Copyright © EnOcean Alliance Inc. (2017-2019). All rights Reserved.

Disclaimer

This information within this document is the property of the EnOcean Alliance and its use and disclosure are restricted. Elements of the EnOcean Alliance specifications may also be subject to third party intellectual property rights, including without limitation, patent, copyright or trademark rights (such a third party may or may not be a member of the EnOcean Alliance.)

The EnOcean Alliance is not responsible and shall not be held responsible in any manner for identifying or failing to identify any or all such third party intellectual property rights. This document and the information contained herein are provided on an "as is" basis and the EnOcean Alliance disclaims all warranties express or implied, including but not limited to
(1) any warranty that the use of the information herein will not infringe any rights of third parties (including any intellectual property rights, patent, copyright or trademark rights, or (2) any implied warranties of merchantability, fitness for a particular purpose, title or non-infringement.

In no event will the EnOcean Alliance be liable for any loss of profits, loss of business, loss of use of data, interruption of business, or for any other direct, indirect, special or exemplary, incidental, punitive or consequential damages of any kind, in contract or in tort, in connection with this document or the information contained herein, even if advised of the possibility of such loss or damage. All Company, brand and product names may be trademarks that are the sole property of their respective owners.
The above notice and this paragraph must be included on all copies of this document that are made.

The EnOcean Alliance "Communication Profile Certification" is available free of charge to companies, individuals and institutions for all non-commercial purposes (including educational research, technical evaluation and development of non-commercial tools or documentation.)

This specification includes intellectual property („IPR") of the EnOcean Alliance and joint intellectual properties („joint IPR") with contributing member companies.   No part of this specification may be used in development of a product or service for sale without being a participant or promoter member of the EnOcean Alliance and/or joint owner of the appropriate joint IPR.

These errata may not have been subjected to an Intellectual Property review, and as such, may contain undeclared Necessary Claims.

EnOcean Alliance Inc.

5000 Executive Parkway, Suite 302

San Ramon, CA 94583

USA

Graham Martin

Chairman & CEO EnOcean Alliance

Table of Contents

1. Introduction 6

1.1. Definitions & references 6

1.1.1. Definitions 6

1.1.2. References 8

2. Certification strategy for application profiles. 9

2.1. Introduction 9

2.1.1. Scope 9

2.1.2. Summary 9

2.1.3. General Definitions and Remarks 9

2.2. Testing strategy 10

2.2.1. Testing strategy of inbound device – data consumer 10

2.2.2. Testing strategy of outbound test - data emitter 11

2.2.3. Strategy remarks 12

2.3. Dataset composition 12

2.3.1. Telegrams/ Raw Value 12

2.3.2. Real Data 13

2.4. XML Markup 13

2.5. Direction of testing 14

2.6. Interdependency 14

2.7. Test steps for individual telegram fields 15

2.7.1. Enum / flag value 15

2.7.2. Scale measurement 16

2.7.3. Unused bits 17

3. EnOcean Equipment Profiles (EEP) 18

3.1. EEP Description characteristics 18

3.1.1. Enum Range 18

3.1.2. Bi-directional profiles 18

3.1.3. Teach-in Procedures 19

3.1.4. Direction 20

3.2. XML 20

3.2.1. EEP specific certification XML 20

3.2.2. How-to EEP to certification xml 21

4. Generic profiles 28

4.1. Generic profile device 28
4.2. Generic profile controller 28
4.3. XML 29

4.3.1. GP specific certification XML 29
4.3.2. How to: GP to certification xml 31

5. Remote Management & Remote Commissioning 37

5.1. Remote Management Test process 37
5.2. Remote Commissioning Test process 38
5.3. XML file 39

6. Test documentation 41

1. Introduction

1.1. Definitions & references

1.1.1. Definitions

Commissioning - all operations involved in the management of target devices, first time setup, daily maintenance, troubleshooting, changes in field, etc.

Dataset – The set of data used to verify the profile. It is composed of a telegram and the parsed values for this telegram.

DUT – **D**evice **u**nder **t**est

EEP - **E**nOcean **E**quipment **P**rofile; Specification to define structure of over-the-air data bytes for EnOcean transmitters. Also see Generic Profiles.

EURID – A unique identification number hard-coded to every EnOcean transmitting module during the production process

GP – **G**eneric **P**rofile; A specification outlining a communication language which allows devices to describe, at runtime, what data it will transmit.

Inbound – receiving device which receives e.g. a Teach-in request message; Inbound also refers to the incoming communication direction. The inbound device is the device which "consumes" the application profile to check.

ManID – Manufacturer ID. The EnOcean alliance assigns a unique manufacturer ID to its members. This Id is stored inside products and can be used to easily identify the manufacturer.

Outbound – A device which can transmit EnOcean telegrams, e.g. a Teach-in request. Outbound also refers to the outbound communication direction. The outbound device is the device which "produces" the application profile to check.

Product ID – A 6 byte value where the two most significant bytes are the Manufacturer ID and the least four significant bytes are a manufacturer defined product identifier. This is used to link a target device to its device description file.

Remote Management – For content and functionality of EnOcean's Remote Management specification refer to [3]. The remote commissioning specification is built on top of this. Remote Management defines the base for interoperable commissioning.

Remote Commissioning - the application of the EnOcean Remote Management functionality by defining new standardized RPCs. It is the process of commissioning target devices without requiring physical access to the device. This can be accomplished in an interoperable way with the Remote Commissioning interface and processes defined in this document.

Teach-in – A process to connect or logically link two devices to establish uni/bi-directional communication between them. The details of different teach-in processes can be found in related specification documents [1] and [2].

Teach-out – A process to disconnect or unlink two devices. This is the opposite process to Teach-in.

1.1.2. References

[1] EnOcean Equipment Profiles (EEP).
http://www.enocean-alliance.org/eep/

[2] EnOcean Generic Profiles 1.0 (GP).
http://portal.enoceanlliance.org/DesktopModules/Inventures_Document/FileDownload.aspx?ContentID=21174

[3] EnOcean Remote Management Specification.
http://www.enocean.com/fileadmin/redaktion/pdf/tec_docs/RemoteManagement.pdf

[4] EnOcean Wireless Standard
http://www.enocean-alliance.org/en/enocean_standard/

2. Certification strategy for application profiles.

2.1. Introduction

2.1.1. Scope

This document provides the blueprint for creating the test cases needed to certify EnOcean products. Based on the functions that a device is able to perform, this document provides a method to isolate and test individual features of the application protocol as well as the guidelines for embedding these steps within an XML structure.

2.1.2. Summary

For each used protocol or profile, e.g. Generic Profiles, EEP, Remote Management and Remote Commissioning, a profile test is defined. Each profile test can contain different tests and each test for a device is divided into steps. Each step is defined by its inputs and outputs. The DUT is provided with inputs and the outputs are recorded. If the recorded output matches the defined output the test is passed. The outputs are unambiguous. For protocol certification the EnOcean telegrams play the central role, they are used as the inputs and outputs of the certification process.

The number of tests for a given feature is dependent on the number of use cases that it includes. Each feature will be tested individually by forcing the device to perform each function and checking that the correct output is observed. Each defined step shall contain only one dataset.

As a general rule, when a command or feature is dependent on another, the test should implement all the necessary settings and sending of telegrams as steps.

Whenever applicable, all consecutive unused zero bits are grouped together and their encoding is verified in each test step e.g. unused bytes must have the given defined values. The same guidelines are applied when designing the test cases for binding processes e.g. the Teach-in and Teach-out procedures.

Values which are not under test, e.g. the humidity value of one profile while testing the temperature, can be ignored. They will be tested in separate defined test cases.

2.1.3. General Definitions and Remarks

For each tested feature, the expected telegram format and the expected real values are given in a dataset. This dataset can be used to verify either the inbound or the outbound device.

The normal communication flow is that the telegram is sent by the outbound device, and the inbound device parses the telegram and represents the value. For bidirectional communication, where the inbound device may send telegrams to the outbound device, the send direction is explicitly mentioned.

Remote management assumes bidirectional communication. Therefore tests in remote management will be defined on message level. A message flow is defined consisting of a request message, sent by the outbound device, and a response message from the inbound device. This communication flow is verified in the certification process.

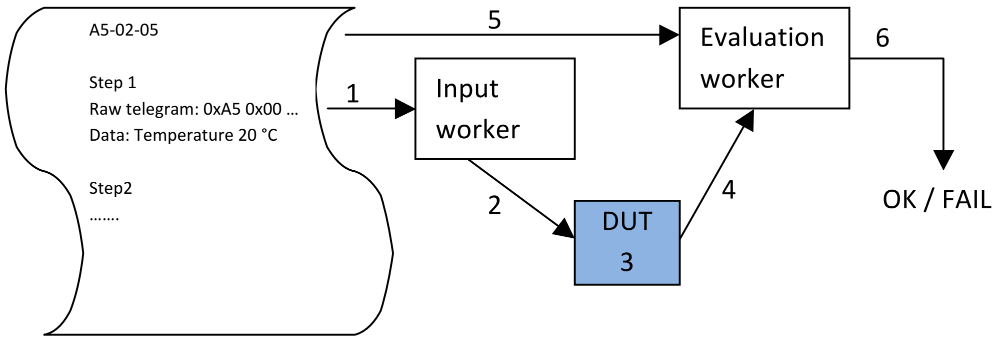
2.2. Testing strategy

The aim of testing is to validate the EnOcean protocol interface of the DUT.

The EnOcean protocol interface is represented by the ability to encode and decode telegrams according the specified protocol rules of EEP, GP, Remote Management and Remote commissioning. We do not focus to test/certify the application and processes which consume/produce the data included in the telegrams.

For explanation purposes the testing strategy is described for EEP – EnOcean equipment profiles. The Same assumption applies for other protocols and the specifics of each Protocol are listed separately.

The testing strategy for inbound and outbound devices is different. The following diagram shows the message flow for both cases described in the next two chapters.



2.2.1. Testing strategy of inbound device – data consumer

The following test scenario applies for testing a Gateway which claims to support a profile, e.g. A5-02-05. The dataset is represented as an XML file specified later in this document.

1. Take a specified, simulated telegram from the dataset with specified values.
2. Provide a raw telegram as input for the input worker.

3. DUT - Gateway decodes data from the raw telegram
4. Take Data from the output
5. Take the specified real data from the dataset.
6. Compare DUT output data with specified real data=> OK / FAIL

The input Worker can be:

- A person that transmits the telegrams. In this case the test uses the actual radio interface.
- A script that tests the given application in pure simulated software environment. In this case the functions to parse the specific EEP are verified. No radio communication is executed.

The Evaluation worker can be a real person that compares the data or a testing framework, operating in a simulated environment.

The evaluation worker reads out the parsed information from the DUT on an interface which should be immediately after the telegram processing function of the DUT. Otherwise, other processes which are not part of the profile certification may affect the result.

2.2.2. Testing strategy of outbound test - data emitter

A similar example is a test of a device such as a sensor which claims to transmit a specific profile e.g. A5-02-05. The dataset defined in the xml for the consumer is reused as it contains real values and the actual telegrams.

In the tests of the outbound device, the simulated and evaluated values are reversed. The real values are simulated and the resulting raw telegrams are evaluated. Please consider following steps:

1. The input worker takes the real value from the dataset.
2. The Input worker triggers the DUT to send a telegram with the specified data content. This can be executed by putting the device into specific conditions where the DUT measures the requested value. E.g. put into a climate chamber at 20 °C. Another approach is to simulate the value on a defined sensor interface, e.g. provide a defined resistance instead of a thermistor. The DUT could also be controlled with a serial interface using debug commands to send specific real values.
   In other words the Input Worker needs to trigger the DUT to transmit the specified telegram
3. The DUT sends the telegram.
4. The evaluation worker receives the telegram.
5. The evaluation worker takes the expected telegrams from the dataset.

6. The received telegram from DUT is compared with the telegram from dataset to decide the result of the test.

The input worker and the output worker tasks can be performed in a simulated software environment. For the certification itself it is 100 % sufficient to test only the routines which are responsible for the EEP interface.
The expected telegrams can contain bits which can be ignored. An ignore mask is defined for the validation of the received telegrams.

2.2.3. Strategy remarks

The certification is only valid for one device with a defined firmware version, which is identified by its unique product-id.

The “EnOcean Communication Certification” certifies the communication interface of a device. To achieve certification, the software functionality can either be unit tested or the device can be black box tested.

If the functionality of firmware parts (or the complete firmware) for the communication is verified using unit tests in a test framework, the manufacturer of the device shall ensure that the device functions the same way in final operating conditions. The manufacturer confirms the functionality with his signature.

If the functionality is tested with a black box test, testing the whole device with the final firmware or a modified firmware with a debug interface, the manufacturer of the device shall ensure that the device functions the same way in final operating conditions. The manufacturer guarantees the functionality with his signature.

If the manufacturer uses a black box test for the final verification of his final product and the specified communication verification is a part of the black box test, he may omit the previous mentioned unit or black box testing with debug interface.

2.3. Dataset composition

Each dataset consists of two parts, the telegram (raw values) and the expected parsed value. This dataset can be used for the inbound and the outbound devices. If no direction is mentioned the outbound device is fed with the real values which it uses to compute and send the telegram. The inbound device is fed with the telegram and computes the real value. The computed data from the DUT is compared with the values in the dataset.

2.3.1. Telegrams/ Raw Value

In the telegram field, each byte is enumerated in sending order. For the expected telegrams (outbound test, data emitter) an ignore mask is created for bits which are not relevant to the test. The bits set in the ignore mask shall by ignored when the telegram is validated. If the telegram is sent by the test worker, the bytes shall be sent as defined.

For Remote Management or Remote commissioning, the telegram field specifies the manufacturerID, functionCode and, if necessary, the bytes to send with that functionCode. I.e. when unlocking the device the unlock code has to be added to the manufacturerID and the functionCode.

For GP, instead of a telegram, a list of GP channels can be given for parsing selective data. It is then the task of test worker to generate (or parse) the selective data telegram. A detailed description of how selective data messages are defined is given in the Generic Profiles specification [2].

2.3.2. Real Data

The real data contains the interpreted real values. Each value has an identifier and the expected value.

For the GP Teach-in the real data is the list of channels with the GP channel parameters according to [2].

For Remote Management or Remote Commissioning no real values are defined, instead a response message in telegram format is used. This response message is the result of the Remote Management or Remote commissioning function call and represents the real data values.

2.4. XML Markup

An XML file is defined for each of the different certification profiles and an XSD is available for each to verify that the format is correct. The different XMLs are explained in detail in the chapters for the different profiles.

The following base structure is valid for each XML:

```
<ProfileCertification> : root tag
 <ProfileTypeToTest attribute=identifier> : profile to test with identifier, e.g. rorg-func-type
  <TestCase> : Test for one feature, e.g. temperature
   <Name/> : name for the feature tests (Temperature tests)
    <Test>: One test, e.g. max temperature test
     <Name/>: Name the test
       <Step>: one step of the test
         <Dataset/>A pair of "telegram" data and real values, differs for profiles.
        </Dataset>
      </Step>
  </Test>
</TestCase>
</ProfileTypeToTest>
</ProfileCertification>
```

There are the following tags to differ the xml files between the profile types:

• Generic Profiles:	*<GP …>*
• EEP:	*<EEP …>*
• Reman:	*<Reman …>*

2.5. Direction of testing

Depending of the direction of the testing (outbound or inbound) the dataset can be defined with the *outboundDeviceAction* tag. This specifies if the outbound device should send or receive the telegram.

<Dataset outboundDeviceAction="Receive">
If this attribute is set to "Receive", the outbound DUT is expected to receive the specified telegram, and compute the defined parsed values.
If the *outboundDeviceAction* attribute is not specified, the "*Send*" action is used.

A test case can also only be specified for a specific *DeviceType* with the *onlyValidFor* tag:
<TestCase onlyValidFor="InboundDevice">
Here only an inbound DUT would execute these tests.
If the *onlyValidFor* attribute is not specified, the test case is valid for both devices.

2.6. Interdependency

Interdependency means that the value of a field in a telegram is dependent on another or that after sending a certain telegram, an answer is expected.

For telegrams with interdependency, in the profile, the linked values shall be checked. For example if the temperature is only available if the temperature sensor bit is set, the necessary values for the test step shall be verified.

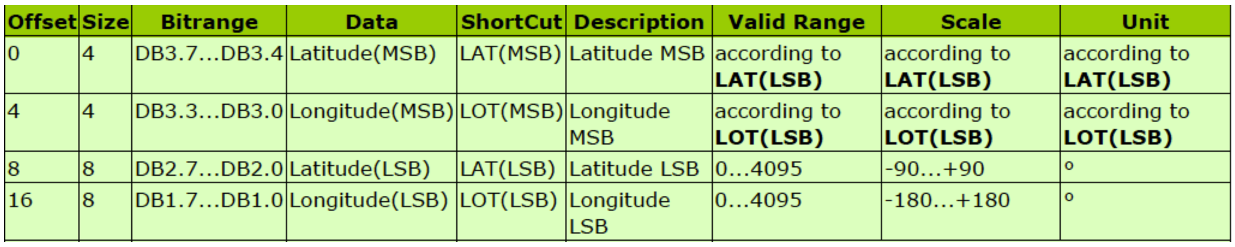
If a certain communication flow is expected, each expected telegram shall be listened as independent test step for example:S

Step 1: Send ReCom Get MetaData Query

Step2: Expect answer with certain values.

Example A5-13-06

Interdependent Measurement Scales



Offset	Size	Bitrange	Data	ShortCut	Description	Valid Range	Scale	Unit
0	4	DB3.7...DB3.4	Latitude(MSB)	LAT(MSB)	Latitude MSB	according to **LAT(LSB)**	according to **LAT(LSB)**	according to **LAT(LSB)**
4	4	DB3.3...DB3.0	Longitude(MSB)	LOT(MSB)	Longitude MSB	according to **LOT(LSB)**	according to **LOT(LSB)**	according to **LOT(LSB)**
8	8	DB2.7...DB2.0	Latitude(LSB)	LAT(LSB)	Latitude LSB	0...4095	-90...+90	°
16	8	DB1.7...DB1.0	Longitude(LSB)	LOT(LSB)	Longitude LSB	0...4095	-180...+180	°

1. Create a first test and select the Latitude (LSB) option
   - Trigger the device to encode a latitude of -90 degrees in the less significant bit in the first test step
   - Verify that the latitude's MSB follows the correct scale by triggering the device to encode -90 degrees in the most significant bit, in the second test step
   - Repeat the two previous test steps by triggering the device to measure two more angle values, notably the latitude's LSB and MSB as 0 degrees, and the latitude's LSB and MSB as 90 degrees.
2. Create a second test and select the Longitude (LSB) option
   - Trigger the device to encode a longitude of -180 degrees in the less significant bit in the first test step
   - Verify that the longitude's MSB follows the correct scale by forcing the device to encode -180 degrees in the most significant bit, in the second test step
   - Repeat the two previous test steps by triggering the device to measure two more angle values, notably the longitude's LSB and MSB as 0 degrees, and the longitude's LSB and MSB as 179 degrees.
3. Create a third test of the device and this time verify both latitude and longitude measurements at the same time
   - Trigger the device to measure 40 degrees in latitude and -40 degrees in longitude and verify that the encodings of LSB and MSB bits in both cases are correct.

2.7. Test steps for individual telegram fields

In following chapter the test steps of individual telegram fields are described. In general all EnOcean profiles, EEP and GP, contain fields with similar characteristics. Following are defined:

- Enumeration/Flags – value out of a list
- Scaling value / Analog value – value represents a generic field of any kind of information
- Unused bits – bits with a defined value

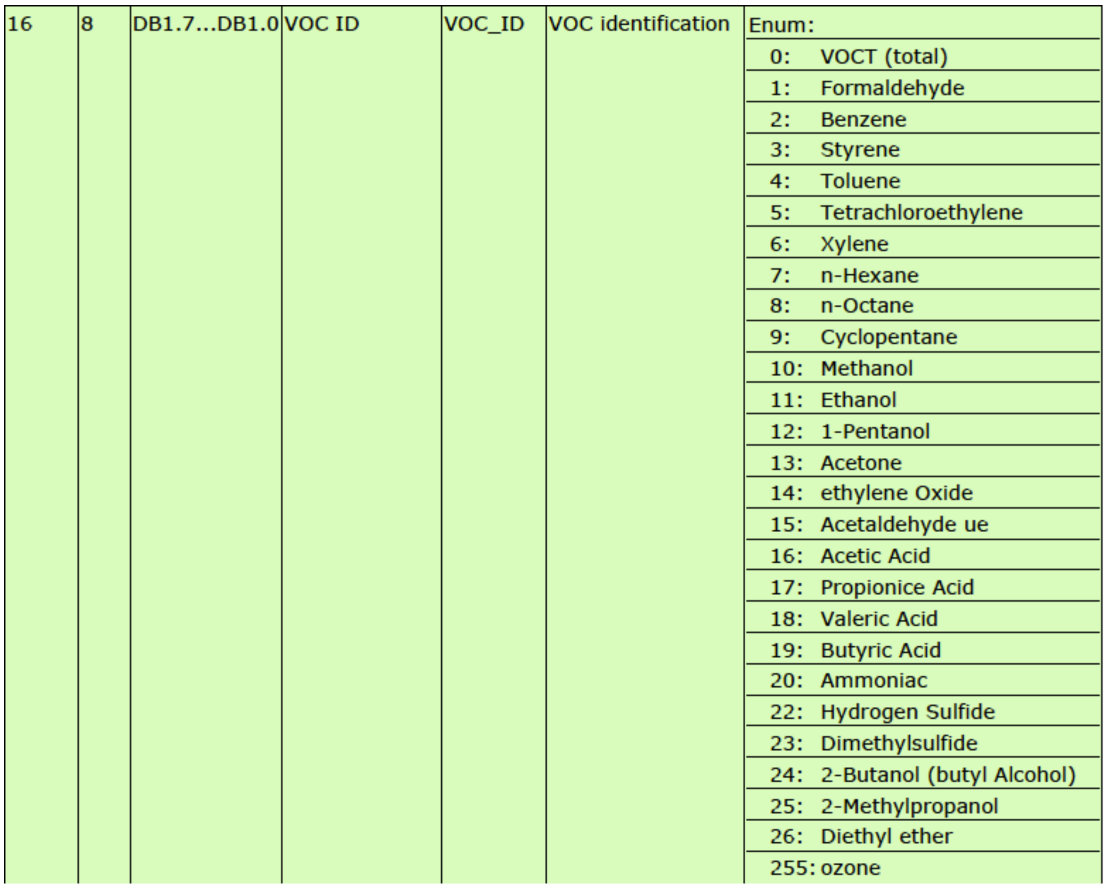
2.7.1. Enum / flag value

Description characteristic: 1 enum, 1 test

Select the user option corresponding to each possible enum value (speed level, rocker action, etc.) and verify the encoding of each of those values in the telegram.

A flag can be seen as special case enumeration with only 2 possible values (0 and 1).

<u>**Example EEP A5-09-05**</u>



16	8	DB1.7...DB1.0	VOC ID	VOC_ID	VOC identification	Enum:
						0: VOCT (total)
						1: Formaldehyde
						2: Benzene
						3: Styrene
						4: Toluene
						5: Tetrachloroethylene
						6: Xylene
						7: n-Hexane
						8: n-Octane
						9: Cyclopentane
						10: Methanol
						11: Ethanol
						12: 1-Pentanol
						13: Acetone
						14: ethylene Oxide
						15: Acetaldehyde ue
						16: Acetic Acid
						17: Propionice Acid
						18: Valeric Acid
						19: Butyric Acid
						20: Ammoniac
						22: Hydrogen Sulfide
						23: Dimethylsulfide
						24: 2-Butanol (butyl Alcohol)
						25: 2-Methylpropanol
						26: Diethyl ether
						255: ozone

Strategy: Verify the encoding of each possible enum. Thus, this EEP table would be tested with 28 tests.

2.7.2. Scale measurement

Description characteristic: Measurement range

The test design should use 3 different values within the defined range, usually the two extremes and the mid-point (*)(**), regardless of the number of bits used by the feature : resolution is not used because the valid range can almost always be encoded in one byte – 0 to (less than or equal to) 255. The goal is to verify that the scale to valid range conversion was done correctly and that the resulting valid range number was correctly encoded in the telegram.

(*) If either the extreme or the midpoint measured values cannot be converted to an integer in the valid range of the sensor, then a different test value that can be encoded without truncation should be chosen.

(**) If either of the two extreme values of a sensor scale is too difficult to physically attain with the available test facilities, other more reasonable values should be chosen.

Example A5- 02-0B

16	8	DB1.7...DB1.0	Temperature	TMP	Temperature (linear)	255...0	+60...+100	°C

Strategy: Trigger the device to measure three different temperature values and verify if the measurement value is encoded correctly. Usually the two extremes and mid-point of the range should be tested, unless these do not yield exact values and require truncation or approximation. If this is the case, other temperature values that yield an exact conversion should be used. The EEP table above requires 3 tests. The recommended temperature test values in this case would be 60°C (0xFF in the telegram), 84°C (0x66) and 100 °C (0 x00)).

2.7.3. Unused bits

Characteristic Description: Unused 0 bits in a telegram

All unused bits or set of bits will be included in the dataset and verified along with feature encodings.

Example A5-02-05

Offset	Size	Bitrange	Data	ShortCut	Description	Valid Range	Scale	Unit
0	8	DB3.7...DB3.0	Actual valve	AV	Actual valve	0...100	0...100	%
8	8	DB2.7...DB2.0	Not Used (= 0)					
16	8	DB1.7...DB1.0	Temperature	TMP	Temperature (linear)	0...255	0...+40	°C
24	4	DB0.7...DB0.4	Not Used (= 0)					
28	1	DB0.3	LRN Bit	LRNB	LRN Bit	Enum:		
						0:	Teach-in telegram	
						1:	Data telegram	
29	3	DB0.2...DB0.0	Not Used (= 0)					

Strategy: To test the above table, the 0 values are set to 0 in the telegram of the dataset.

3. EnOcean Equipment Profiles (EEP)

This chapter includes the strategy how to certify and test EEPs.

Each telegram is identified by an RORG, a FUNC and a TYPE number. The RORG specifies the EnOcean radio protocol type, the FUNC specifies the basic functionality of the data type and the TYPE specifies the individual characteristics of any particular device. Each EEP should have an individual and unique RORG-FUNC-TYPE number.

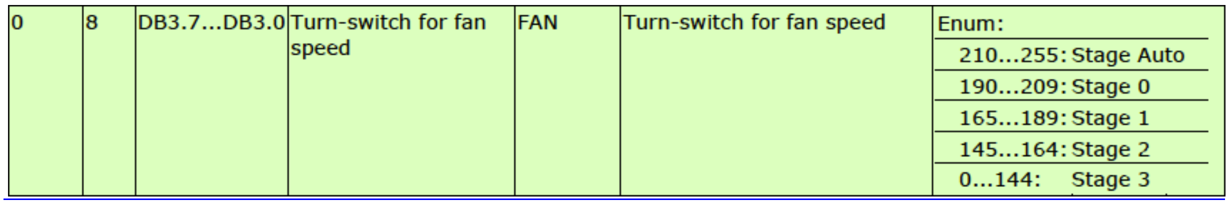
3.1. EEP Description characteristics

3.1.1. Enum Range

Description characteristic: Enum range (x enum, 1 feature)

Select the user option corresponding to every enum range and verify that the encoded value is situated in the desired range.

Example A5-10-01



0	8	DB3.7...DB3.0	Turn-switch for fan speed	FAN	Turn-switch for fan speed	Enum: 210...255: Stage Auto 190...209: Stage 0 165...189: Stage 1 145...164: Stage 2 0...144: Stage 3

Strategy: Place the device in each one of the 5 stages and verify that the encoded enum is in the correct range. In this example the enum table requires one <*test*> and five <*step*> tags. Each step represents one stage and the expected result is the range of the corresponding stage enum.

3.1.2. Bi-directional profiles

Description Characteristic: Devices allowing bi-directional communication (4BS and VLD)

All profiles can be seen to be used in two directions, inbound and outbound. Each profile will define a certification process for the inbound device, which received the Teach-in telegram, the outbound device, which generated the Teach-in telegram.

All bidirectional VLD EEPs will be tested similar to the 4BS EEPs.

3.1.3. Teach-in Procedures

For Teach-in test purposes, it is assumed that the device under test is executing an outbound Teach-in.

If the profile supports more than one Teach-in method (i.e. 4BS and UTE) both methods have to be defined in the xml and the method(s) supported by the device shall be tested.

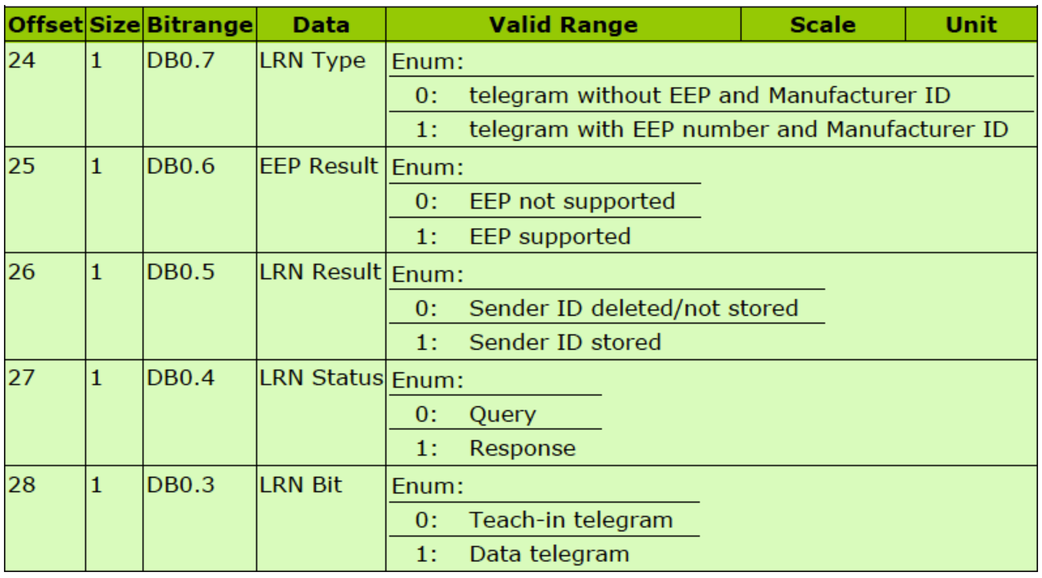
4BS Learn Telegram Teach-in (Unidirectional)

This case concerns devices having one data bit reserved for the Teach-in process. A test for this kind of Teach-in can be designed using one test.

Test: Verify the Teach-in routine

Test step: Trigger the device to send a Teach-in telegram. Here the correct EEP and manufacturer ID will be verified.

4BS Learn telegram Teach-in (Bidirectional)



Offset	Size	Bitrange	Data	Valid Range	Scale	Unit
24	1	DB0.7	LRN Type	Enum: 0: telegram without EEP and Manufacturer ID 1: telegram with EEP number and Manufacturer ID		
25	1	DB0.6	EEP Result	Enum: 0: EEP not supported 1: EEP supported		
26	1	DB0.5	LRN Result	Enum: 0: Sender ID deleted/not stored 1: Sender ID stored		
27	1	DB0.4	LRN Status	Enum: 0: Query 1: Response		
28	1	DB0.3	LRN Bit	Enum: 0: Teach-in telegram 1: Data telegram		

Test: Verify the Teach-in routine

Test Step: Place the inbound device in learn mode

Test Step: Trigger the outbound device to send a Teach-in query telegram

Test Step: Verify the encoding of the Teach-in response received

Universal Teach-in

Remark: The RORG number used for this Teach-in procedure is D4

Test: Verify the universal Teach-in routine for the outbound device

Test Step: Set the inbound and outbound devices in learn mode

Test Step: Trigger the outbound device to send an EEP Teach-in Query message and consider the time when this message is sent (T0).

Test Step: Record the time when the response from the inbound device is received (T1).

Test Step: Verify that T1 - T0 $\leq$ 700ms to validate the test (this calculation is needed for bidirectional devices only)

3.1.4. Direction

We define these aspects of DUT:

- Outbound device – device which transmits the Teach-in telegram
- Inbound device – device which is able to decode EEP Teach-in and data telegrams. If a profile is bi directional it can answer accordingly.

3.2. XML

3.2.1. EEP specific certification XML

The schema of the XML is specified in the Certification_EEP.xsd file. This chapter gives a short overview over the tags and attributes which differ from the main XML format. The XML file eep_example.xml uses all available tags with a short comment.

*The <EEP rorg="0x**" func="0x**" type="0x**">* tag defines that the XML specifies the tests for an EEP.

The available tags for the dataset are:

- *<Telegram rorg=”0x**” length=”*”>*, this defines the telegram generated by the worker or validated by the worker.
- *<RealData>*, to feed the DUT or to validate the telegram data.

The *<Telegram>* tag consists of the length (decimal value) and *rorg*(hex number) attribute. The tag <Byte> is a child of the <Telegram> tab and is used to define 1 byte of the telegram. There should be one <Byte> tag for each byte of the telegram. The attributes of the *<Byte>* tag are *order* and the *ignoremask*(hexnumber).

```
<Telegram rorg="0xA5" length="4">
  <Byte order="0">0x00</Byte>
  <Byte order="1" ignoremask="0xFF">0x34</Byte>
```

```
    <Byte order="2" ignoremask="0xFF">0x00</Byte>
    <Byte order="3">0x08</Byte>
   </Telegram>
```

The *rorg* defines the RORG of the telegram, e.g. A5 (4BS) and the *length* defines the payload length of the telegram. The bytes are enumerated in order and their values are expressed as hex values. The *ignoremask* specifies which bytes can be ignored for validating the data. (RealTelegram.Byte[X]&(~ignoremask[X])==Expected Telegram.Byte[X]&(~ignoremask[X])).

The *<RealData>* defines the parsed values for the dataset, e.g.

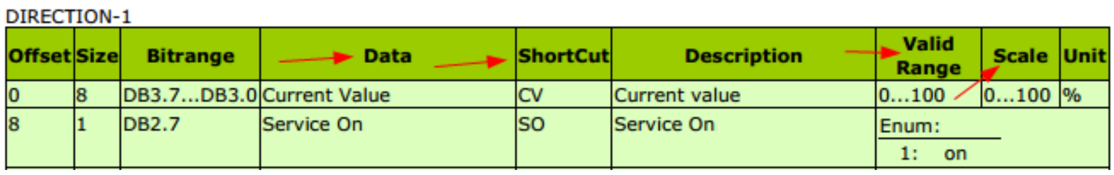
```
<RealData>
   <Data>
    <Shortcut>TSN</Shortcut>
    <Type>T-Sensor</Type>
    <Value>0</Value><!--decimal value -->
   </Data>
   <Data>
    <Type>LRN Bit</Type>
    <Value>1</Value>
   </Data>
 </RealData>
```

The *Shortcut* tag is the shortcut from the EEP spec, the *type* is the value from the EEP table and the *value* is the parsed expected value expressed in decimal format.

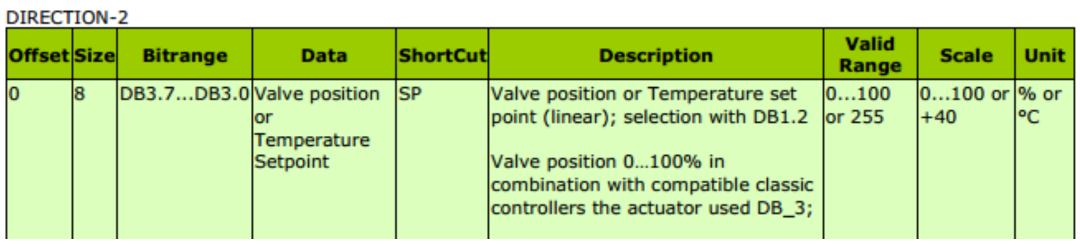
3.2.2. How-to EEP to certification xml

For this example parts of the A5-20-01 EEP are specified for a certification test. The profile is a bidirectional profile.

The tests for the *CurrentValue* and *ServiceOn* are shown as example for the direction 1, and for direction 2 the test for "Valve position or Temperature Setpoint" are shown. The Teach-in test is also shown.

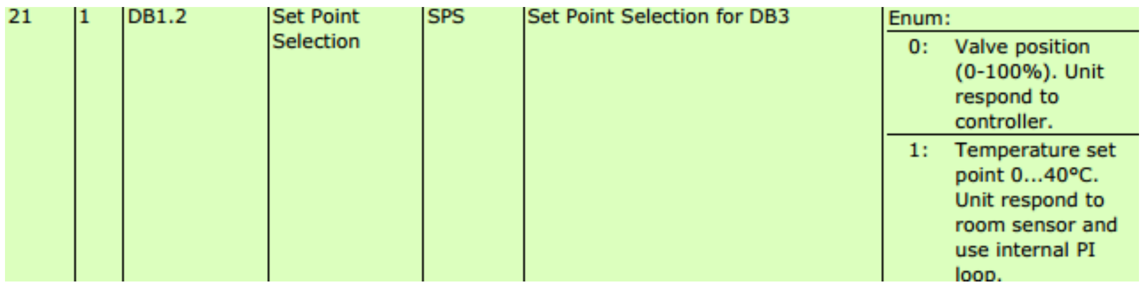

DIRECTION-1

Offset	Size	Bitrange	Data	ShortCut	Description	Valid Range	Scale	Unit
0	8	DB3.7...DB3.0	Current Value	CV	Current value	0...100	0...100	%
8	1	DB2.7	Service On	SO	Service On	Enum: 1: on		



DIRECTION-2

Offset	Size	Bitrange	Data	ShortCut	Description	Valid Range	Scale	Unit
0	8	DB3.7...DB3.0	Valve position or Temperature Setpoint	SP	Valve position or Temperature set point (linear); selection with DB1.2 Valve position 0...100% in combination with compatible classic controllers the actuator used DB_3;	0...100 or 255	0...100 or +40	% or °C



21	1	DB1.2	Set Point Selection	SPS	Set Point Selection for DB3	Enum: 0: Valve position (0-100%). Unit respond to controller. 1: Temperature set point 0...40°C. Unit respond to room sensor and use internal PI loop.

The xml starts with the Teach-in test, and the *EEP* tag contains the rorg, func and type.

```
<?xml version="1.0" encoding="utf-8"?>
<ProfileCertification xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="./Certification_EEP.xsd" schemaVersion=”0.9”>
 <EEP rorg="0xA5" func="0x20" type="0x01">
  <TestCase>
  <Name>
  Teach In Tests
  </Name>
<Test>
   <Name>Unidirectional Tech In Test</Name>
   <Step>
    <Dataset>
     <Telegram rorg="0xA5" length="4">
       <Byte order="0">0x80</Byte>
       <Byte order="1">0x0F</Byte>
       <Byte order="2">0xFF</Byte>
       <Byte order="3" ignoremask="0x7F">0x80</Byte>
     </Telegram>
     <RealData>
      <Data>
       <Type>LRN Bit</Type>
       <Value>0</Value>
      </Data>
      <Data>
       <Type>RORG</Type>
       <Value>165</Value>
      </Data>
      <Data>
       <Type>FUNC</Type>
       <Value>32</Value>
      </Data>
      <Data>
       <Type>TYPE</Type>
       <Value>1</Value>
      </Data>
     </RealData>
    </Dataset>
   </Step>
   </Test>
  </TestCase>
….
```

The current value is a scaled value and three tests are defined for this case: min, max and medium value. Using the offset, size, valid range and scale the raw values in the telegram can be calculated as specified in chapter 1.8 of the EEP specification [1]. The first byte represents the current value and for the dataset the following pairs can be calculated:

- 0% : 0x00
- 50% :0x32
- 100% : 0x64

For the Current Value tests, most of the other bytes can be ignored, only the not used bits (DB0.7-DB0.4 + DB0.2-DB0.0) and the LRN Bit(DB0.3) needs to be checked.

```
…
<TestCase>
 <Name>
 Current Value Tests
 </Name>
```

```
  <Test>
   <Name>Min  Current Value Test</Name>
   <Step>
    <Dataset>
     <Telegram rorg="0xA5" length="4">
       <Byte order="0">0x00</Byte>
       <Byte order="1" ignoremask="0xFF">0x00</Byte>
       <Byte order="2" ignoremask="0xFF” >0x00</Byte>
       <Byte order="3">0x08</Byte>
     </Telegram>
     <RealData>
      <Data>
       <Shortcut>CV</Shortcut>
       <Type>Current  Value</Type>
       <Value>0</Value>
      </Data>
      <Data>
       <Type>LRN Bit</Type>
       <Value>1</Value>
      </Data>
     </RealData>
    </Dataset>
   </Step>
   </Test>
…
</TestCase>
```

The test for the medium and the max value are similar.

The test for the *ServiceOn* = true would be:

```
…
<TestCase>
  <Name>
  ServiceOn Enum Test
  </Name>
   <Test>
    <Name> ServiceOn on</Name>
    <Step>
     <Dataset>
      <Telegram rorg="0xA5" length="4">
      <Byte order="0" ignoremask="0xFF">>0x00</Byte>
      <Byte order="1" ignoremask="0x7F">0x80</Byte>
      <Byte order="2" ignoremask="0xFF" >0x00</Byte>
      <Byte order="3">0x08</Byte>
     </Telegram>
     <RealData>
      <Data>
      <Shortcut>SO</Shortcut>
      <Type>Service On</Type>
      <Value>1</Value>
     </Data>
      <Data>
      <Type>LRN Bit</Type>
      <Value>1</Value>
      </Data>
     </RealData>
    </Dataset>
   </Step>
   </Test>
…
</TestCase>
```

AS an example of packets which are sent by the outbound device, the direction 2 telegram with the “valve position or temperature set point” tests are shown. As this byte depends on the DB1.2, it is included in the dataset.
For the maximal valve position the test would have the following structure.

```
…
<TestCase>
  <Name>
  Valve position Tests
  </Name>
…
   <Test>
    <Name> Max Valve position </Name>
    <Step  outboundDeviceAction="Receive">
     <Dataset >
     <Telegram rorg="0xA5" length="4">
      <Byte order="0">0x64</Byte>
      <Byte order="1" ignoremask="0xFF">0x00</Byte>
      <Byte order="2" ignoremask="0xFB” >0x00</Byte>
      <Byte order="3">0x08</Byte>
    </Telegram>
    <RealData>
     <Data>
      <Shortcut>SPS</Shortcut>
      <Type>Set Point Selection</Type>
      <Value>0</Value>
    </Data>
     <Data>
      <Shortcut>SP</Shortcut>
      <Type>Valve position</Type>
      <Value>100</Value>
    </Data>
     <Data>
      <Type>LRN Bit</Type>
      <Value>1</Value>
     </Data>
    </RealData>
   </Dataset>
  </Step>
  </Test>
…
</TestCase>
```

And for the maximal Temperature setpoint:

```
…
<TestCase>
  <Name>
  Temperature setpoint Tests
  </Name>
…
   <Test>
    <Name> Max Temperature setpoint </Name>
    <Step  outboundDeviceAction="Receive">
     <Dataset >
      <Telegram rorg="0xA5" length="4">
        <Byte order="0">0xFF</Byte>
        <Byte order="1" ignoremask="0xFF">0x00</Byte>
        <Byte order="2" ignoremask="0xFB" >0x04</Byte>
        <Byte order="3">0x08</Byte>
      </Telegram>
      <RealData>
        <Data>
        <Shortcut>SPS</Shortcut>
        <Type>Set Point Selection</Type>
        <Value>1</Value>
      </Data>
        <Data>
        <Shortcut>SP</Shortcut>
        <Type> Temperature Setpoint </Type>
        <Value>40</Value>
      </Data>
      </RealData>
     </Dataset>
    </Step>
    </Test>
…
</TestCase>
```

4. Generic profiles

Generic profiles test process philosophy is same as for EEP. The difference is how XMLs will be created due to the generic nature of different profiles.

We define these aspects of DUT:

- Generic Profile outbound device – device which has a defined GP profile and transmits a Teach-in telegram
- Generic Profiles inbound device – device which understands Generic Profiles as language (i.e. supports all defined features or a subset of Generic Profiles)

4.1. Generic profile device

Since GP allows a wide spectrum of communication profiles we use a similar approach of templates as defined in Remote Commissioning. DUT manufacturer will create an XML for certification tests with DUT Specific data. For creating and validating this XML an XSD is available and different example XML files.

<u>EXAMPLE</u>

A GP device is created with 3 different Signal types, 1x Enum and 1x Flag type. The DUT Manufacturer creates an XML file with the following features:

- *A specific Teach-in telegram is tested in one test case*
- *six test cases are created to test all 6 values in a complete message– if available*
- *six test cases are created to test selective data message – if available*
- *Additional test steps will be added if DUT is bidirectional*
- *At least one generic profile message type must be implemented and tested.*

4.2. Generic profile controller

To test a generic profile controller, XML files with test cases considering a set of Teach-in messages and data messages will be defined. The test cases will aim to test several combinations of data types and configurations. The generic language allows huge variations so not all aspects can be considered. The controller test will be test if the device can support the Generic Profile language – not a specific profile.

4.3. XML

4.3.1. GP specific certification XML

*<GP learnTelegram="0x**">* tag defines that the XML specifies the tests for a GP. The *learnTelegram* is the GP Teach-in telegram for the profile and is used for the worker.

The available tags for the dataset are:

- *<Telegram rorg="0x**" length="*">*, this defines the telegram generated by the worker or to be validated by the worker.
- *<SelectiveData>* instead of the telegram tag, used to define selective data messages.
- *<RealData>*, to feed the DUT or to validate the telegram data.
- *< GPTeachIn>* instead of the *RealData* tag to define the GP channel definitions.

The *<Telegram>* tag consists of the length (decimal value) and *rorg*(hex number) attributes. The <Byte> tag is a child of the <Telegram> tag which defines 1 Byte of the telegram. The attributes of the *<Byte>* tag are *order* and the *ignoremask*(hexnumber).

```
<Telegram rorg="0xB2" length="2">
 <Byte order="0">0x00</Byte>
 <Byte order="1" ignoremask="0x38">0x00</Byte>
</Telegram>
```

The *rorg* defines the RORG of the telegram, e.g. 0xB2(GP_CD) and the *length* defines the payload length of the telegram. The bytes are enumerated in order and their values are expressed as hex numbers. The *ignoremask* specifies which bytes can be ignored when validating the data.
(RealTelegram.Byte[X]&ignoremask[X]==Expected Telegram.Byte[X]&ignoremask[X]).

In a Generic Profile full data message, it has to be checked if channels are transmitted which are not defined in the Teach-In telegram.

Instead of *<Telegram>* the *<GPSelectiveData>* tag can be used to check generic profile selective data messages. For the *GPSelectiveData* all channels which should be checked shall be defined using the *<Channel>* tag. The index attribute defines the GP Channel index, and the value of the channel is the raw value expressed as a hex number. To calculate the raw value the measurement value quantization equations from chapter (3.4) of [2] shall be used. The GP Selective Data Message (RORG=0xB3) is defined in chapter 5.2.2. and the worker needs to be able to parse this message into the different channels.

```
<GPSelectiveData>
 <Channel index="0"  bitResolution="10">0x00</Channel>
</GPSelectiveData>
```

The *real data* defines the parsed values for the dataset, e.g.

```
<RealData>
<Data channelIndex="0">
    <Value>40</Value>
   </Data>
   </RealData>
```

Each data tag uses the *channelIndex* attribute to identify the GP Channel. The *channelIndex* is the channel number according to [2].

If the parsed telegram is the GP Teach-in Telegram, the *<GPTeachIn>* tag shall be used.

```
<GPTeachIn>
   <Direction>1</Direction>
   <Purpose>0x00</Purpose>
   <ManufacturerID>0x7FF</ManufacturerID>
   <GPChannel index="0" type="0x01">
    <ValueType>0x00</ValueType>
    <SignalType>0x18</SignalType>
    <Resolution>0x07</Resolution>
    <EngMaximum>0x28</EngMaximum>
    <ScalingMaximum>0x01</ScalingMaximum>
    <EngMinimum>0x00</EngMinimum>
    <ScalingMinimum>0x00</ScalingMinimum>
   </GPChannel>
   <GPChannel index="1" type="0x04">
    <ValueType>0x00</ValueType>
    <SignalType>0x01</SignalType>
    <Resolution>0x01</Resolution>
   </GPChannel>
   <GPChannel index="2" type="0x03" direction="inbound">
    <ValueType>0x00</ValueType>
    <SignalType>0x01</SignalType>
   </GPChannel>
   </GPTeachIn>
```

Each channel shall be identified with a *<GpChannel>* tag and the index and type shall be identified with a tag and index attribute. The *<Direction> <Purpose> <ManufacturerID>* tags represent the special fields of the GP Teach-in message according to [2]. The *<ProductID>* tag can be used if a productID Channel TeachInfo message is sent. All *<GpChannel>* tags use the *<ValueType>* and *<SignalType>* tag. For type 0x01 (Signal) channels, additionally the *<Resolution>* , *<EngMaximum>, <EngMinimum>* and *<ScalingMinimum>* tags are used. For type 0x04 (enum) the *<Resolution>* tag is needed. Type 0x03 (flag) only uses the global tags *<ValueType>* and *<SignalType>*.

4.3.2. How to: GP to certification xml

For creating the specification xml the channel definition list and the Teach-in is needed. Every used channel needs to be tested as well as the supported Teach-in functionality.

The start of the xml and Teach-in test has the following structure:

```
<?xml version="1.0" encoding="utf-8"?>
<ProfileCertification xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="./Certification_GP.xsd" version="0.9">
```

The *GP* type is opened with the *GP* tag containing the learn Telegram. The first test is the GP Teach-in test

```
 <GP learnTelegram="0x01700081048D159E5185C004A07014609401038050">
  <TestCase>
  <Name>
  Teach In Tests
  </Name>
   <Test>
    <Name>Gp Teach In Telegram</Name>
    <Step>
     <Dataset>
      <Telegram rorg="0xB0" length="21">
        <Byte order="0">0x01</Byte>
        <Byte order="1">0x70</Byte>
        <Byte order="2">0x00</Byte>
        <Byte order="3">0x81</Byte>
        <Byte order="4">0x04</Byte>
        <Byte order="5">0x8D</Byte>
        <Byte order="6">0x15</Byte>
        <Byte order="7">0x9E</Byte>
        <Byte order="8">0x51</Byte>
        <Byte order="9">0x85</Byte>
        <Byte order="10">0xC0</Byte>
        <Byte order="11">0x04</Byte>
        <Byte order="12">0xA0</Byte>
        <Byte order="13">0x70</Byte>
        <Byte order="14">0x14</Byte>
        <Byte order="15">0x60</Byte>
        <Byte order="16">0x94</Byte>
        <Byte order="17">0x01</Byte>
        <Byte order="18">0x03</Byte>
        <Byte order="19">0x80</Byte>
        <Byte order="20">0x50</Byte>
      </Telegram>
```

All expected Channels are defined in the teach-in process and can be identified with the unique index attribute.

```
<GPTeachIn>
<Direction>1</Direction>
<Purpose>0x00</Purpose>
<ManufacturerID>0x7FF</ManufacturerID>
```

This is a data channel

```
<GPChannel index="0" channelType ="0x01">
  <ValueType>0x00</ValueType>
  <SignalType>0x18</SignalType>
  <Resolution>0x07</Resolution>
  <EngMaximum>0x28</EngMaximum>
  <ScalingMaximum>0x01</ScalingMaximum>
  <EngMinimum>0x00</EngMinimum>
  <ScalingMinimum>0x00</ScalingMinimum>
</GPChannel>
```

This is an enum channel with 2-bit resolution

```
<GPChannel index="1" channelType ="0x04">
  <ValueType>0x00</ValueType>
  <SignalType>0x01</SignalType>
```

```
    <Resolution>0x01</Resolution>
  </GPChannel>
```

A flag channel

```
  <GPChannel index="2" channelType ="0x03">
    <ValueType>0x00</ValueType>
    <SignalType>0x09</SignalType>
  </GPChannel>
```

An inbound flag channel.

```
<GPChannel index="3" channelType ="0x03" direction="inbound">
    <ValueType>0x00</ValueType>
    <SignalType>0x01</SignalType>
  </GPChannel>
  </GPTeachIn>
 </Dataset>
 </Step>
 </Test>
</TestCase>
```

The whole GP Teach-in telegram is transmitted over the air. The worker listens to the telegram and compares it with an XML file, in which all channels of the DUT are defined.

For each defined channel, tests cases shall be defined. For this example there are outbound test cases for channel indexes 0, 1 and 2, and an inbound test for channel index 3.
Each *gpchannel* type(data, flag and enum) is supported and uses a substructure to define the channel.

For channel index 0, signal type temperature, min, medium and max tests need to be defined. This can either be via the *<Telegram>* tag or using the *<GPSelectiveData>*

```
<TestCase>
  <Name>
  Temperature Tests Complete Data
  </Name>
   <Test>
    <Name>Min Temperature Test</Name>
    <Step>
    <Dataset>
     <Telegram rorg="0xB2" length="2">
      <Byte order="0">0x00</Byte>
      <Byte order="1" ignoremask="0x38">0x00</Byte>
     </Telegram>
     <RealData>
      <Data channelIndex="0">
       <Value>0</Value>
      </Data>
     </RealData>
    </Dataset>
   </Step>
   </Test>
…
</TestCase>
```

Or with selective data:

```
<TestCase>
  <Name>
  Temperature Tests Selective Data
  </Name>
   <Test>
   <Name>Min Temperature Test</Name>
   <Step>
    <Dataset>
     <GPSelectiveData>
      <Channel index="0"  bitResolution="10">0x0FF</Channel>
     </GPSelectiveData>
     <RealData>
      <Data channelIndex="0">
       <Value>0</Value>
      </Data>
     </RealData>
    </Dataset>
   </Step>
   </Test>
```

The test cases for the other channels are similar, for the enum channel each possible enum needs to be checked and for the flag channel the “0” or “1” value shall be verified.

For the inbound channel tests, the *<Dataset> outboundDeviceAction* needs to be set accordingly:

```
…
<TestCase>
 <Name>
 Inbound Flag Channel Test
 </Name>
  <Test>
   <Name>False</Name>
   <Step>
    <Dataset outboundDeviceAction="Receive">
     <Telegram rorg="0xB2" length="1">
       <Byte order="0">0x00</Byte>
     </Telegram>
     <RealData>
      <Data channelIndex="3">
       <Value>0</Value>
      </Data>
     </RealData>
    </Dataset>
   </Step>
   </Test>
 <Test>
   <Name>True</Name>
   <Step>
    <Dataset outboundDeviceAction="Receive">
     <Telegram rorg="0xB2" length="1">
       <Byte order="0">0x10</Byte>
     </Telegram>
     <RealData>
      <Data channelIndex="3">
       <Value>1</Value>
      </Data>
     </RealData>
    </Dataset>
   </Step>
   </Test>
```

```
</TestCase>
```

5. Remote Management & Remote Commissioning

The DUT in remote commissioning certification process will be tested to verify the operation of the communication interface.

We define one aspect:

The DUT is the remote device which is the one being managed e.g. HVAC controller

A test specification for a generic remote management and remote commissioning device, which is the "manager", is not provided. Such a test is reasonable, but out of the scope of this working package.

The DUT will not be certified for behavior, only the communication-interface message content will be considered during testing.

The evaluation and test process of Remote Management & Remote Commissioning is based on the same principle as for EEPs – a message will be sent and the response will be checked for the expected content. The test is only passed if the defined content rules were respected.

The tests are described in an XML file.

One of the main differences between the EEP and GP tests is that all tests need to be executed in order.

As the interface for the validation of the different profiles, based on remote management**,** only the communication interface is used. After sending remote management telegrams (SYS_EX rorg) certain answers are expected, the details for the different profiles (RMCC, RPC and RECOM) are mentioned in the sub chapters.

5.1. Remote Management Test process

In [3] all messages and responses are defined. Each test step will be focused to test one specific use case or operation of an RMCC. Every remote management device shall support all RMCCs.

Features:

For each use case defined in [3] (e.g. SECURITY feature, Location Feature etc.) tests are defined.

In these tests all defined RMCC and RPC (not relevant to Remote Commissioning) will be checked.

If the tested command uses an answer telegram, the answer is evaluated.

If a response is not available other ways of confirmation will be defined – e.g. Query Status.

Test cases can consist of a defined message flow – several request and response pairs which need to be executed in a defined order.

The test cases are defined in an XML file and can be copied and pasted depending on the supported features of the device.

It is in the responsibility of the manufacturer of the DUT to guarantee that the device supports the Remote management message flow and the defined tests are valid Remote management messages and communication flow processes.

The assumption is that the device is locked with either the standard lock code or a specific one. The first test step should be to unlock the device.

5.2. Remote Commissioning Test process

In remote commissioning the DUT will be specifically tested against its Device Description file (DDF). If a device supports any features defined in the remote commissioning specification, the respected device needs to have a DDF. For RPCs and use cases which are not dependent on the DDF, test process will be same as for Remote Management defined in Chapter 5.1.

The Remote Commissioning Mandatory Commands (Apply Changes, Get Product ID Query and Response, and Remote Commissioning Acknowledge) are always tested and are not specified in the DDF as available features.

DUT features that are defined by a DDF (e.g. parameter list, type of parameters) will be tested with specific adjusted test cases. The DDF specifies exactly which Remote Commissioning features are available. An example with tests is defined which can be used by the manufacturer of the DUT to define his own tests.

For each feature which can be defined with the DDF a corresponding XML template test exists. The different XML template tests can be used as base for writing a Remote Commissioning test. All test cases are bundled into one XML file.

To validate the Remote Commissioning communication, either the explicit response is evaluated or if an explicit response is not sent, the Remote Commissioning Acknowledge is expected and parsed. In addition, If a parameter is modified, it must be read back and verified.

It is in the responsibility of the manufacturer of the DUT to guarantee that the device supports the Remote Commissioning message flow as defined in the specification.

5.3. XML file

For the Remote Management and Remote Commissioning tests the tag *<RemanTelegram fnCode=”0x***” manufacturerID=”0x***>* is specified to validate and generate the Remote Management telegrams.

Each *<Step>* only contains in the *<Dataset>* a *<RemanTelegram>*, *<Manual Step>* or *<TimeoutExpected>*. *<ManualStep>* indicates that the device status needs to be modified with commands which are not available as an RPC, e.g. rebooting. *<TimeoutExpected>* defines that after sending a Remote Management telegram, we expect no answer at all, e.g. locking the device and then using the query status telegram, for which no answer should be given.

For example using [3] and the “lock” command with the security code 0x01020304 the Remote Management telegram in the XML has the following structure:

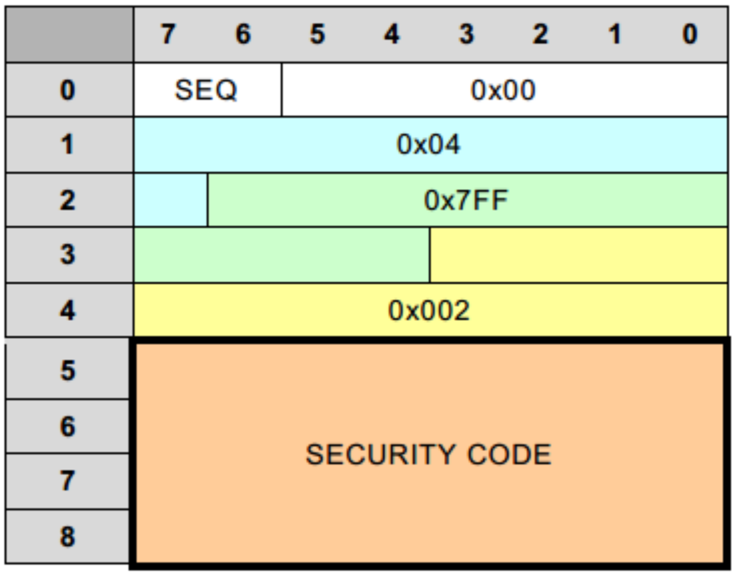


Figure 1 Lock Telegram from [3]

<RemanTelegram fnCode=”0x002” manufacturerID=”0x7FF” length=”4”>
    <Byte order=”0”>0x01</Byte>
    <Byte order=”1”>0x02</Byte>
    <Byte order=”2”>0x03</Byte>
    <Byte order=”3”>0x04</Byte>
</RemanTelegram>

A fully working example for testing the Unlocking of a device with the code 0x01020304, the device was previously locked and the security code set to 0x01020304:
<?xml version="1.0" encoding="utf-8"?>

```
<!-- this is an example for a test for a reman type certification, this can be every sort, rpc,recom
... -->
<ProfileCertification xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
  xsi:noNamespaceSchemaLocation="./Certification_Reman.xsd">
  <!-- the Reman tags open the test for RPC/RMCC/RECOM….-->
 <Reman>
   <!-- Test for unlocking the device -->
    <Test>
     <Name>Unlock the Device</Name>
      <Step>
      <RemanTelegram outboundDeviceAction="Receive" fnCode="0x001"
manufacturerID="0x7FF" length="4">
          <Byte order="0">0x01</Byte>
          <Byte order="1">0x02</Byte>
         <Byte order="2">0x03</Byte>
         <Byte order="3">0x04</Byte>
        </RemanTelegram>
       </Step>
<!-- query status-->
       <Step>
         <RemanTelegram outboundDeviceAction="Receive" fnCode="0x008"
manufacturerID="0x7FF" length="0"/>
      </Step>
 <!--  Get answer -->
     <Step timeoutInMs="500">
      <RemanTelegram fnCode="0x608" manufacturerID="0x00B" length="4">
          <Byte order="0" ignoremask="0x03">0x80</Byte>
         <Byte order="1">0x00</Byte>
         <Byte order="2">0x01</Byte>
         <Byte order="3">0x00</Byte>
       </RemanTelegram>
      </Step>
  </Test>
  </Reman>
</ProfileCertification>
```

6. Test documentation

The results of the tests are to be documented per device, tested using a template provided by the EnOcean Alliance.

The template looks like this:

Location of the certification test:

(Name/*Name*) EnOcean GmbH

(Strasse/*Street*) Kolpingring 18a

(PLZ, Ort/*Location*) (Land/*Country*) 82041 Oberhaching Deutschland

(Datum /Date of test) *2015-04-01*

Device under test:

(Bezeichnung/*Model name*) LEDR

(Produkt/*Product*) LED-Controller with relay

(Serien-Nummer/*Serial no.)* 12345678

(Frequenz / Operating frequency) 868.3MHz

(Antennentyp und Länge / Antenna type and length): Wire antenna / external antenna ; 10 cm

(Foto / Photo)

Photos:

Tested profiles:

(Profil/*profile*) A5-08-15

Results:

Description	*Applied* *yes*	*no*	*Pass* *yes*	*no*
Testcase 1				
Teststep 1				
Teststep 2				
Teststep 3				
Testcase 2				
Teststep 1				
Teststep 2				