System Specification

enocean alliance
Building Smarter Connectivity

Remote Management

V 2.9

San Ramon, CA, USA, 2016

Executive Summary

Ver.	Editor	Change	Date
1.1		Added Smart Acknowledge RPCs	
1.2		Corrected Smart Acknowledge RPCs for simple remote management	
1.3		Migrated to the System Specification Document All the answer commands are defined as UNICAST Corrected EEP definition – there is no such thing as default EEP so EEP mask bits were introduced Set correct Manufacturer ID in answer telegrams Adjusted return codes of answer telegrams, introduced new return codes	
1.4		Corrected EEP mask bit definition, added information about repeating	
1.5		Modified security behavior, if code is set there is no 30min unlock period	
1.6	ASt	Exported from system spec	15.10.2010
1.7	ASt	Moved RPC description to the EEP2.1 specification	14.12.2010
1.8	ASt	Major review, improved text and structure	16.12.2010

1.9	AP	Inserted ERP2 information	16.08.2012
2.0	BE	Manufacturer ID Clarification	06.03.2014
2.5	MH	Review of TWG. Unlock period, attempt period, manager ID storage	06.06.2016
2.6	MH	Feedback from TWG review.	23.08.2016
2.7	MH	Reman and encrypted communication added	14.05.2018
2.8	TM	Updated secure maintenance chaining examples and improved wording	14.11.2018
2.9	AP	Moved RPC description from the EEP2.6.8 specification and added more explanations to some RMCC parameters	13.09.2019

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 noninfringement.

In no event will the EnOcean Alliance be liable for any loss of profits, loss of business, los 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 Remote Management (REMAN) Specification 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. EnOcean Alliance grants no rights to any third party IP, patents or trademarks.

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

EnOcean Alliance Inc.
c/o Global Inventures
5000 Executive Parkway, Suite 302
San Ramon, CA 94583
USA
Graham Martin Chairman & CEO EnOcean Alliance

Table of Contents

1 Introduction

This document describes the functionality of Remote Management. Remote Management allows EnOcean devices to be configured and maintained over the air or serial interface using radio or serial telegrams. Thanks to Remote Management, sensors or switches IDs, for instance, can be stored or deleted from already installed actuators or gateways which are hard to access.

Remote Management also allows querying debug information from the Remote Device and calling some manufacturer implemented functions.

Remote Management is supposed to be used with current and future products, so it has to ensure back compatibility with devices and be extendible for future use. The software of current devices has to be changed in order to support Remote Management. Remote Management is platform independent.

1.1 Definitions & references

1.1.1 Definitions

Sys_ex telegram – is a telegram that is sent through radio or serial interface and is built according to the sys_ex telegram specification.

Message – is information that is sent through radio or serial interface. It consists of one or several sys_ex telegrams.

Command – is a request from Remote Manager to perform a specified reaction.

EEP – EnOcean Equipment Profile

Remote Management – is a network module, which allows Remote Devices to be configured and maintained over the air or via serial interface from a Remote Manager.

Remote Device – is a device that supports Remote Management. In the network model it is the client. The device that handles requests, sends answers and executes functions. It is the managed module.

Remote Manager – is a device that supports Remote Management. In network model it is the manager. The device sends request to execute functions and processes answers. It is the manager module. Usually the actor uses Remote Manager to communicate with remote devices

2 Functional description

Remote Management is performed by the Remote Manager, operated by the actor, on the managed Remote Device (Sensor, Gateway). The management is done through a series of commands and responding answers. Actor sends the commands to the Remote Device. Remote Device sends answers to the actor. The commands indicate the Remote Device what to do. Remote Device answers if requested by the command. The commands belong to one of the main use case categories, which are:

- Security
- Locate / indentify remote device
- Get status
- Extended function.

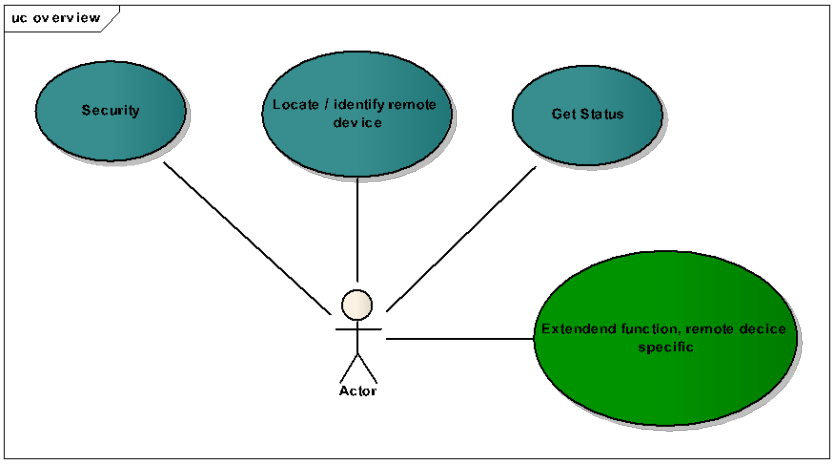


Figure 1 Use case Overview

The management is often done with a group of Remote Devices. Commands are sent as addressed unicast telegrams, usually. In special cases broadcast transmission is also available. To avoid telegram collisions the Remote Devices respond to broadcast commands with a random delay.

The Security, Locate, and Get Status options provide to the actor basic operability of Remote management. Their purpose is to ensure the proper work of Remote Management when operating with several Remote Devices. These functions behave in the same way on every Remote Device. Every product that supports Remote Management provides these options.

Extended functions provide the real benefit of Remote Management. They vary from Remote Device to Remote Device. They depend on how and where the Remote Device is used. Therefore, not every Remote Device provides every extended function. It depends on the programmer / customer what extended functions he wants to add. There is a list of specified commands, but the manufacturer can

also add manufacturer specific extended functions. These functions are identified by the manufacturer ID.

2.1 Security

For security reasons the remote management commands can only be accessed in the unlock period. The period can be entered in:

- Within 5 min after an unlock command with a correct 32bit security code is received

The unlock/lock period can be accessed only with the security code. The security code can be set whenever the Remote Device accepts remote management commands. The state diagram below describes the various states of Remote Management in the view of security.

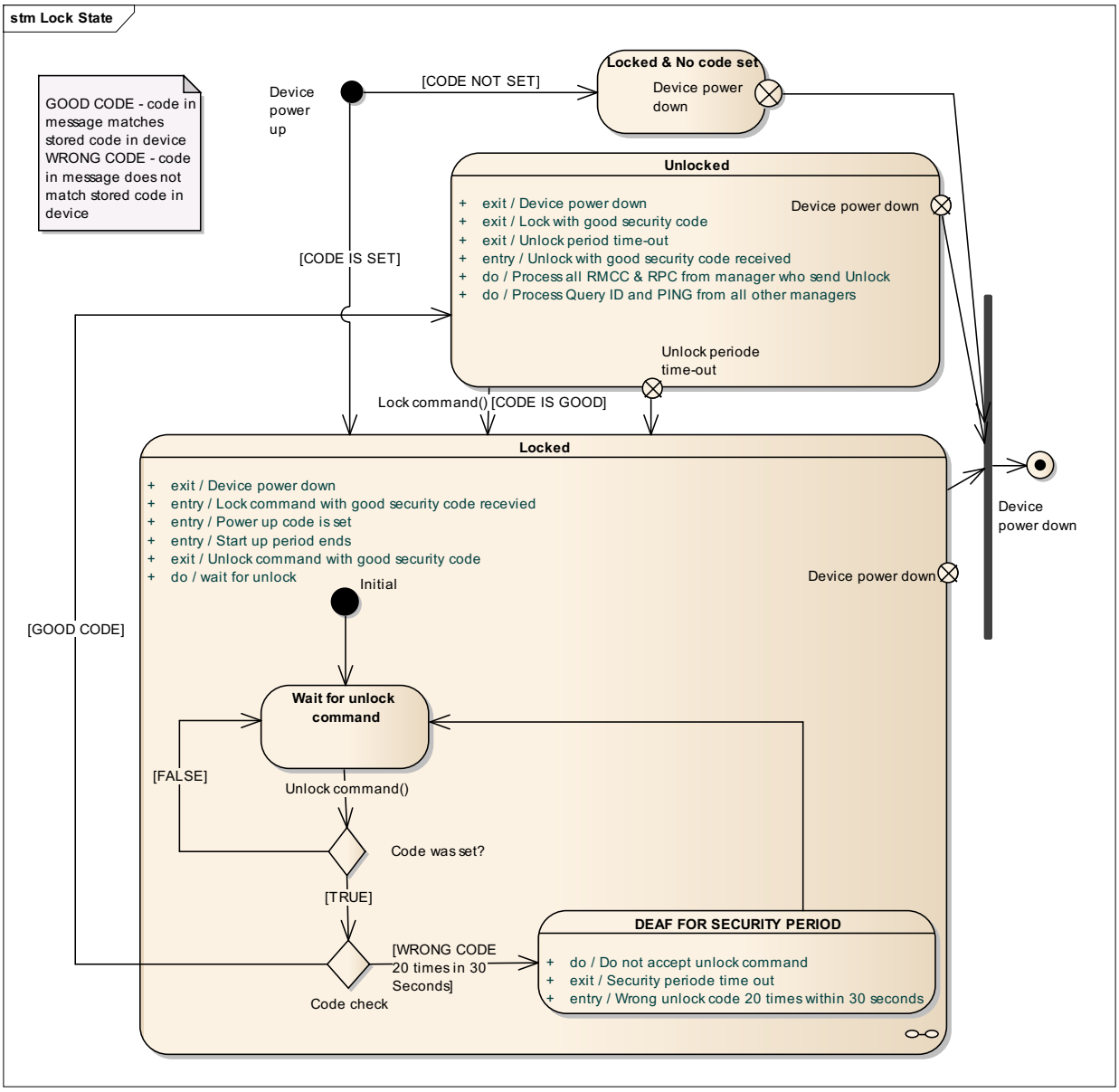


Figure 2 State diagram

When the Remote Device is locked it does not respond to any command, but unlock and ping.

When a wrong security code is received 20 times within the 30 second attempt period the Remote Device does not process unlock commands for a security period of 30 seconds. This limits the risk of finding the security code through random generator. The attempt period is started by first wrong code received. The wrong code counter is incremented by each wrong code received. If the count reaches 20

within the attempt period of 30 second the device lock up for the security period. After the attempt period elapses, again 20 tries are possible.

Once the device is unlocked it processes only commands from the manager that unlocked it. During the following unlock period it will process only RMCCs and RPCs from that manager. This explicitly includes also another Unlock commands. From other managers only Ping commands are processed and Query ID commands in case they are responded with the Query ID Answer Extended. Query ID Answer Extended includes the feature to inform other managers that the device is being locked for an exclusive manager. Authentication of managers is done based on the Sender ID.

During managing process it is recommended to execute repeated UNLOCK command before transmitting extensive configuration to extend the unlock period and ensure that the device do not lock up during management process or message transmission.

If no security code is set (0xFFFFFFFF, 0x00000000), unlock after the unlock period is not processed. Only ping will be processed. 0xFFFFFFFF, 0x00000000 is reserved and cannot be used as security code, it means not code set.

The following commands belong to the security option:

- Set code command

  The command enables the actor to set the security code.

- Lock Command

  The command explicitly locks the device. With the *lock* command it is mandatory to transfer the appropriate security code.

- Unlock Command

  The command unlocks the device. With the *unlock* command it is mandatory to transfer the appropriate security code.

An overview is below.

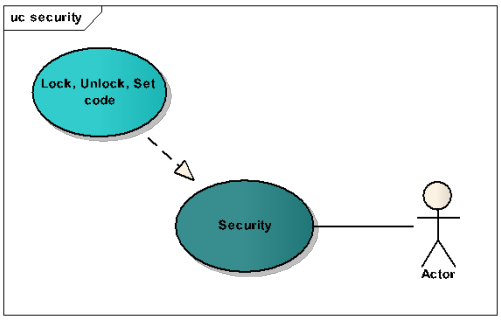


Figure 3 Use case Security

2.2 Locate/identify remote device

This option identifies one desired Remote Device in a group of devices in an unknown environment. To communicate with the Device it is necessary to find out the device ID. For this purpose the following commands are available:

- Query ID Command

  Query ID is sent always as broadcast telegram. All unlocked devices respond to the Query ID with their ID and their EEP.

  The EEP is a 21 bit and it is defined as following: ORG-FUNC-TYPE. For more information about the EEP be sure to read EEP2.1 specification.

  The Query ID command contains an EEP definition and mask bits. When the mask bits are set to 0x01 only Remote Devices with the matching EEP will process the remote command. If the query ID with mask bit 0x00 is transmitted, the EEP bytes in this command will be ignored and every Remote Device will answer to this command. If a Remote Device has no EEP, then it will only respond to the Query ID command where the mask bits are set to 0x00.

  The mask bits in the Query ID answer telegrams are set to 0x00.

  Query ID Answer Extended (0x704) was defined in later reviews and should replace the original Query ID Answer (0x604). The usage of Query ID Answer (0x604) is depreciated. Query ID Answer Extended contains the information about the device being managed by other manager. Locked by other manager (0 – false, 1 - true).

- Action Command

  When this command is received then the addressed device performs an action (audio, visual, etc), depending on the functionality of the device. With this function a remote device with known ID can be can be clearly localized. A detailed description is listed in the use case scenarios.

An overview is below.

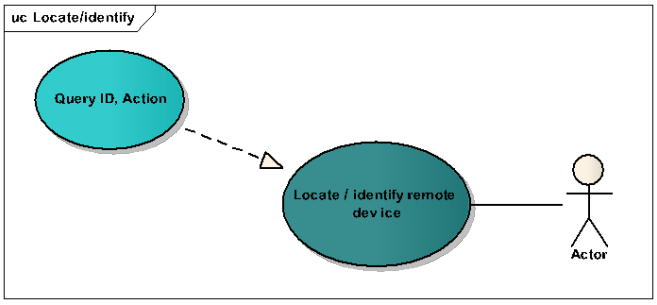


Figure 4 Use Case Locate/identify

2.3 Get status

This option is mainly intended for obtaining debug information from the managed Remote Device. For more information see chapter 4.2.3. The following commands belong to this group:

- Query Status

  With this command the actor directly asks for the status info of the Remote Device. The Remote Device answers with remote management debug data.

  The answer will contain:

  - If a security code is set or not
  - Last remote command function number (RMCC or RPC)
  - Last commands return code (OK, error, etc.)
  - Telegram merge info – last SEQ number and if merge successful or error in receive
- Ping

  The ping command functionality is similar to ping in TCP / IP communication. The actor sends a ping request to see if the Remote Device is alive and communicating. Ping requests are processed also when device is in lock status. Remote device sends the radio signal strength of the received request within the ping response.

An overview is given below.

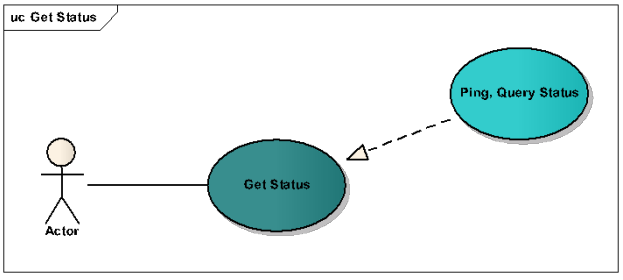


Figure 5 Use Case Get Status

2.4 Extended options

The benefit of extended options in Remote Management is that special and user defined remote device functions can be called remotely. The Remote Management offers ways to call those functions with appropriate commands and parameters. The following actions after the commands are specific for the remote device. The extended functions are specified by their function code and manufacturer ID. Not every remote device supports every extended function.

The following commands belong to this option:

- Call Function Command

  With this command extended functions can be called.
- Query function Command

  With this command the actor requests the supported extended functions list.

It is expected that some functions need to send data back to the actor. The length of the data can vary and exceed the length of the data field in one telegram. For this purpose, the remote manager provides the merge/divide option of data with accurate encapsulation. An overview is given below.

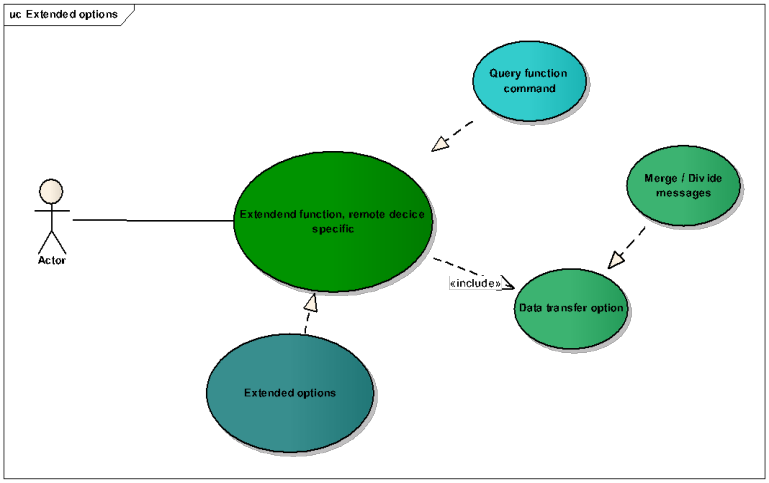


Figure 6 Use Case Extended options

3 Functions and responses

As mentioned the management is done by the commands that directly request an action. For communication reasons they are addressable and support broadcast. The various commands are specific by their function code and remote function they call.

There are two types of remote management commands:

- Remote Management Control Commands - RMCC
- Remote Procedure Calls - RPC

Remote Management Control Commands - RMCCs are available in every product with Remote Management feature. They provide the basic functionality for Remote Management. RMCCs have a common definition. Remote Devices react always in the same way on RMCC.

These commands are:

- Lock
- Unlock
- Set CODE
- Query ID
- Action
- Query status
- Ping
- Query function

RPCs functions strongly depended on the Remote Device. They provide additional functions like remote learn or remote clear of the learned IDs. Not every Remote Device provides the same RPCs. The manufacturer can also determine and implement RPC for his needs. These special RPCs are defined by the function code and Manufacturer Id. The RPC are called with the call function command.

Complete overview of the use case is below.

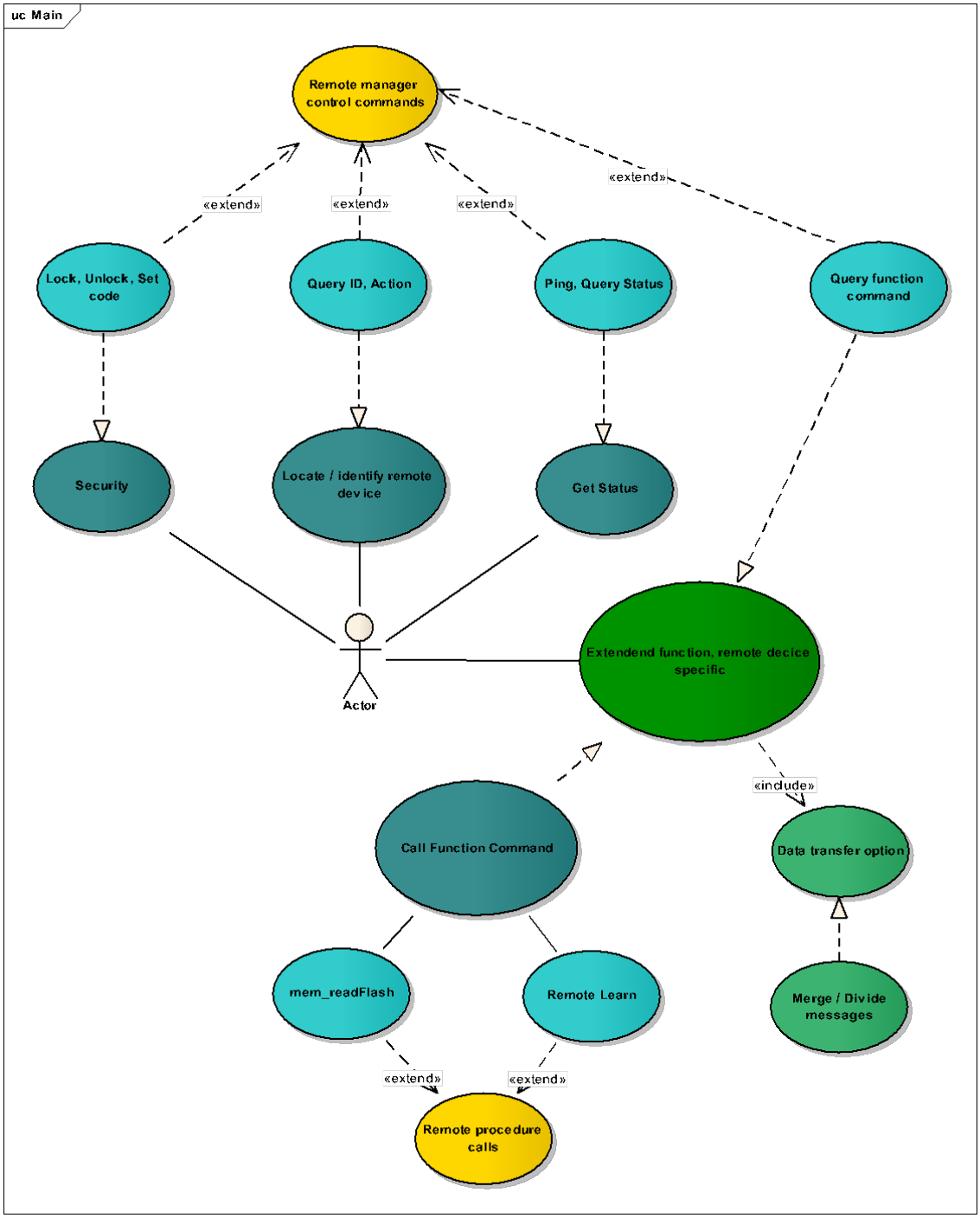


Figure 7 Use Case Main

3.1.1 Query supported function list

The list of supported RPC can be fetched with the query function command. Every Remote Device can support different commands so the Remote Devices have to be queried one by one. In the figure below a scenario of fetching the supported RPC list is shown.

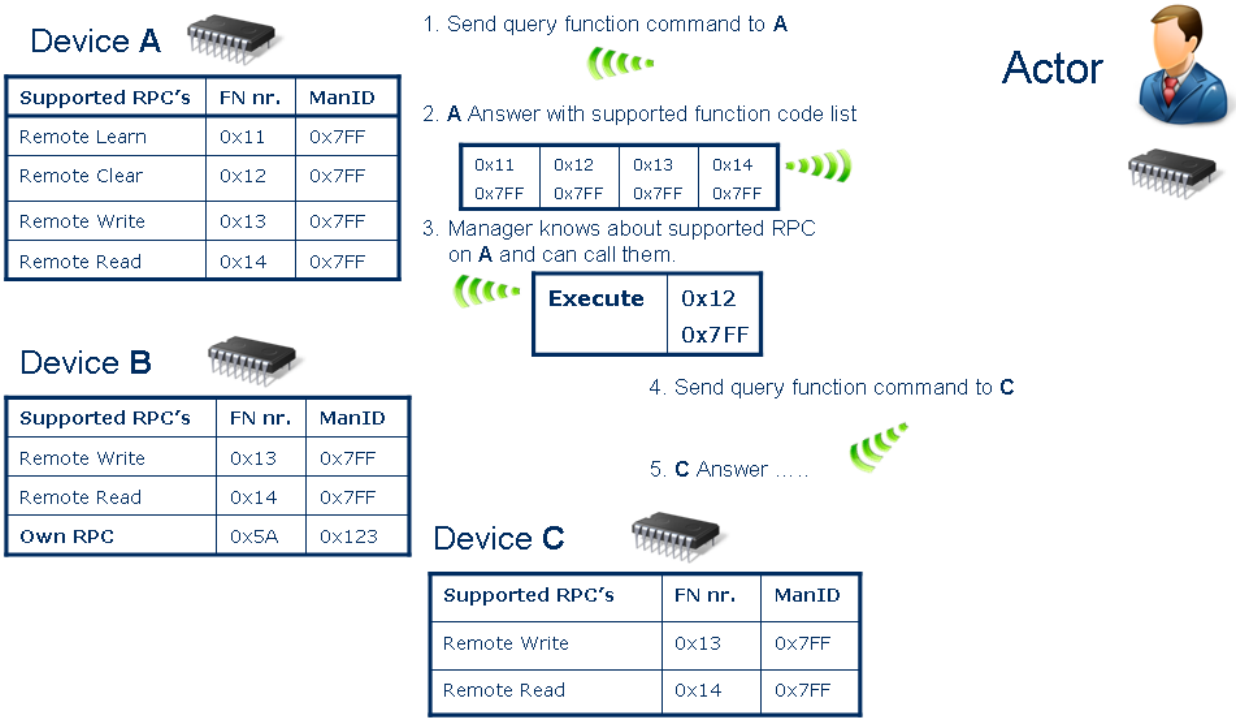


Figure 8 Query supported function list

3.1.2 Remote Device's response to RMCC

In case that a RMCC directly requires to send something back to the actor (query ID, ping, query status, query function command) then this answer will be send immediately with according answer function code. There is strong need to separate the answer function codes from original commands so Remote Device does not process the answers or even react to them. Otherwise that will cause an error state, deadlock or message - answer cycles. In case there is no answer needed, the debug result of the last RMCC will be saved locally and can be fetched remotely with the query status command.

Table 1 RMCC Answers

RMCC	Information contained in answer
Query ID	Remote device's ID and EEP (Function code 0x604). DEPRECIATED. Use 0x704 instead.
Query ID	Remote device's ID and EEP and the locked by another manager information (Function code 0x704).
Ping command	EEP
Query function command	Extended functions list supported by the Remote Device.
Query status	Debug information about the Remote Device.

3.1.3 Remote Device's response to RPC

When a RPC function wants to send an answer with data, then the answer should be send in form of a remote management response. It means that a different function number should be used. Doing so other Remote Devices do not have to process the answers or even react to answers, what can cause an error state. If the amount of data exceeds one telegram, then the response can be separated into more telegrams.

The debug information about the last call is saved locally and can be fetched with query status. It is necessary to separate the answers function codes from original commands.

3.1.4 Broadcast

If the Remote Device recognizes that the received message was a broadcast message and the command requests to send a response, then the answer is send with random delay to avoid collisions. The random delay is between 0 – 2000 ms.

4 Communication Protocol

4.1 Messages structure

The remote management communication happens on the Transport OSI layer and uses Messages as containers. The messages are chained SYS_EX telegrams.

A SYS_EX message may consist of several SYS_EX telegrams. When a message is received the Remote Device merges the SYS_EX telegrams that the message consists of. Then the information is passed as a unit to the application (when RPC).

The length of transferred data is also a part of the protocol and it is used to count the amount of message parts when receiving.

For sending, the SYS_EX message is split into several parts capsulated into SYS_EX telegrams and sent to the receiver module. There the telegrams are merged to a message again. This mechanism is supported by the Remote Manager and also by the Remote Device, so the communication with SYS_EX messages is bidirectional. A status diagram of merge is shown below.

Endianness is Big-endian like in other protocol stacks. But in contrast to other protocol stacks the data field gets filled from left to right. So when three data bytes are sent the three most left bytes get filled and the one most right stays empty.

4.1.1 Message addressing

Each SYS_EX telegram can be addressed. The addressing mechanism is done using ADT encapsulation. For more information please see EnOcean System Specification Address Destination Telegrams. The Remote Device ignores telegrams with other than Remote Device ID or broadcast ID. The broadcast address is 0xFFFFFFFF.

4.1.2 SYS_EX Telegram Structure

For ERP1:

RORG	msg_id (1 b)		data field	sender id	status	crc8
1 byte	**SEQ**	**IDX**	**8 bytes**	4 bytes	1 byte	1 byte
	2 B	**6 B**				

RORG: 0xC5

length: 16 bytes

For ERP2:

Length	Header	Teltype	sender id	msg_id (1 b)		data field	crc8
1 byte	1 byte	1 byte	4 bytes	**SEQ**	**IDX**	**8 bytes**	1 byte
				2 B	**6 B**		

RORG: Extended telegram type 0x00

length: 18 bytes

msg_id:
The telegram identification is a composite of the sequence number, the telegram index and the sender ID.

Sequence number SEQ – the number is for error handling. Telegrams of the message have the same sequence number SEQ. So when a message consists of several telegrams, every telegram of that message has the same SEQ number. The SEQ is random generated by the Remote Management. It ensures that telegrams of several sys_ex messages do not get mixed; we can clearly identify the telegrams by their SEQ numbers.

Telegram index IDX – indicates the order of the telegram in the sys_ex message. This counter starts from 0.

For detailed information please read 4.1.3.

Data field:

Indicates the data transmitted or received. The data field structure is depended on the message.

The first telegram of sys_ex message, with index 0, contains information about the message (Length, Count, Fn_Number, Target_ID). All other telegrams have the same structure.

IDX = 0 (first telegram)

data_length	manufacturer ID	fn_number	payload
9 bits	11 bits	12 bits	32 bits

data_length – total number of data bytes in the message (excluding data_length, manufacturer ID and Fn_Number, may be distributed over several telegrams)

Manufacturer ID – RMCC and EEP defined RPCs will be sent using the multi user (0x7FF) manufacturer ID. Custom RPCs will be sent with the manufacturer ID of the Remote Device. All responses to commands from Remote Devices will use the manufacturer ID of the device responding. The response cannot be multi user.

fn_number – function number to call

payload – custom data, can be interpreted depending on the function

Note: the number of telegrams can be calculated from the data_length

IDX > 0 (all next telegrams)

payload
8 byte

Note: any further telegram / part of the sys_ex message contains only payload

Restrictions:

- The telegrams must use the whole data length, except the last telegram.

- The last telegram is the only one that may have the DATA field partially filled.
- The used bytes in the last telegram can be calculated from the data length field in the first telegram.

4.1.3 Sequence (SEQ) and Index (IDX)

The SEQ, IDX numbers are in every SYS_EX telegram. Telegrams of the same message have the same SEQ number. Every SYS_EX telegram has a specific IDX number. The SEQ = 0 is not allowed. IDX is used to sort and identify message parts. IDX starts at 0 in every following telegram the IDX is incremented. Error handling:

- If parts of a message are not received and the actor sends a new SYS_EX message, the protocol handler recognizes this by the messages SEQ number.
- The telegrams have arrived in a different order as they were sent. In this case, the protocol handler sorts the data with the IDX number

SEQ handling is demonstrated in Figure below.

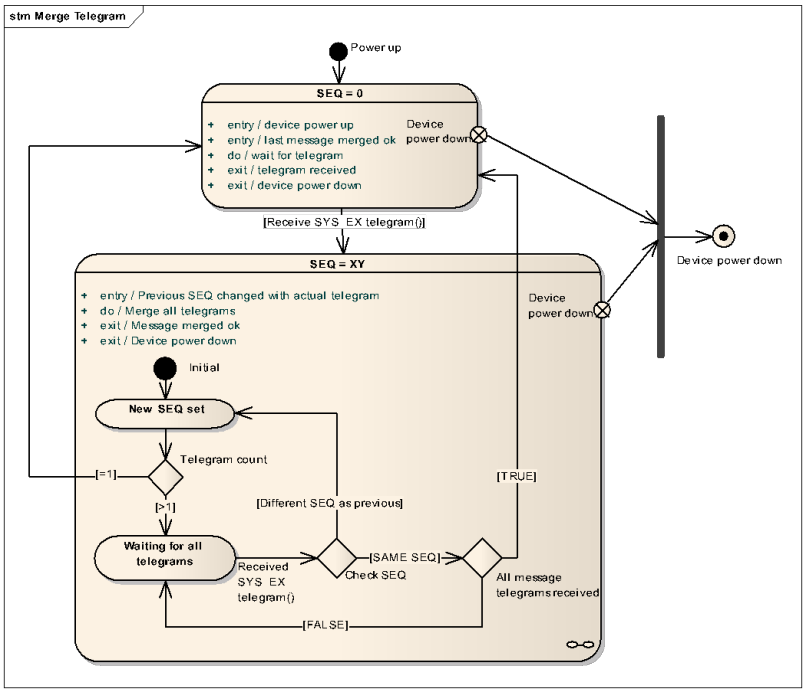


Figure 9 State Diagram Merge Telegram with SEQ

Dividing and merging telegrams into messages enables to transfer bigger amount of data. The dividing and merging process has these characteristics:

- Telegram with IDX 0 has header information about the whole message
- Messages are merged based on same destination Id, source Id, SEQ. IDX orders the telegrams parts.
- In the telegram with IDX = 0 are transmitted 4 bytes of data
- In all next telegrams are transmitted 8 bytes of data

A common message composite of 4 telegrams looks like in the tables bellow. The amount of data is 22 bytes and the function number is 0x210. The telegrams have incremented IDX numbers.

4.2 Error handling in message merge process

Transferring more telegrams chained to a message demands error handling. Individual telegrams of a message can get lost e.g. because of a collision. When working with Remote Management a safer operation must be achieved, even when telegrams get lost or other failures occur. Therefore we declare some error handling and error avoiding mechanisms.

For error handling in Remote Management we declare the following mechanism:

a) INDEX number:

- Order the telegram within a message.
- When telegram with same IDX received twice discard previous message.

b) SEQ number:

- Group telegrams with same SEQ to one message.

c) CHAIN period:

- Telegrams in sequence must be sent within the chain period.
- If a message is not received completely and the chain period expires the message is discarded
- telegrams with other Sender Id are discarded within the chain period

d) Message grouping:

- Messages are differentiated by their Sender ID and SEQ number

Also we declare:

- chained telegrams must be sent in sequence within the chain period
- every telegram can be sent only once (with 3 subtelegrams)

The receive process can work as in Figure 10. On this figure the chain period and the message Sender Id are evaluated. This should happen before the SEQ gets checked.

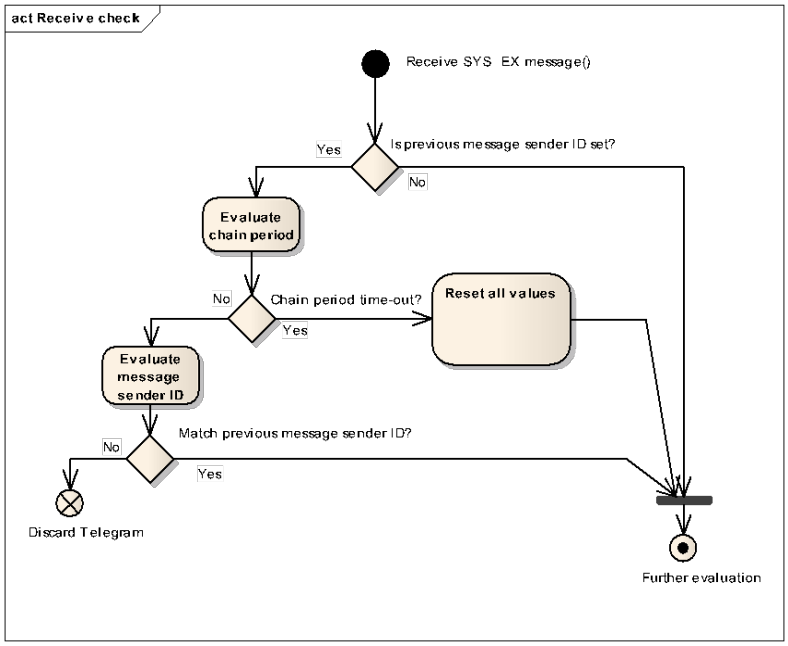


Figure 10 Activity Diagram Check Message ID and Chain period

4.2.1 Timeout

When more managers are operating at the same time telegrams can interfere. Telegrams of one message mix with other telegrams. Telegrams can be sort based on Sender Id, Destination Id and SEQ. To exclude any possible failure we declare a chain period. The chain period determines the maximum time between any two telegrams in a message. When the next telegram is not received within the chain period, the message is declared as corrupted. So if more chained telegram the next must be transferred within the chain period.

Problem Description:

One actor sends a SYS_EX message with 5 telegrams to a desired Remote Device with ID A.

a) 2 of 5 telegrams are received.
b) Between the second and third telegram the second actor sends a broadcast message with the query ID command.
c) The remote device with ID A receives the query ID command.
d) When the remote device has only buffer for one message, the message from the second actor gets processed and the current message from the first actor is corrupt. It cannot be processed, although all telegrams have been received, because the second message has overwritten the first message in buffer.

If we declare, that telegrams with other sender ID do not get processed within the chain period, this failure will not occur. When a remote device can receive more messages at once this will not occur.

Problem Description:

a) The actor sends a SYS_EX message with 5 telegrams to a desired Remote Device with ID A.
b) All telegrams are sent but only 4 are received (with IDX 0,1,2,4).
c) The Remote Device is still waiting for the fifth telegram (with IDX 3).
d) The actor sends a different SYS_EX message with same SEQ and with 4 telegrams.
e) The last telegram (IDX 3) is received as first on the remote device.
f) The Remote Device thinks that the currently received telegram is the last one from the previous message and processes the previous message – error behaviour will occur.

When we discard the message after the chain period expires, it can be clearly declared when a new message is transferred and this issue will not occur. The chain period is 1 second.

4.2.2 Check already received telegram

Every telegram has its IDX number that identifies the position within the whole message. Only if all telegrams are received the message gets processed. Every telegram is transferred only once, so if a telegram with same IDX is within a message received again it indicates that a possible failure occurred.

Problem Description:

The actor sends a SYS_EX message with 5 telegrams to a desired Remote Device with Id A.

a) All telegrams are send but only 4 are received (with IDX 0,1,2,4).
b) The Remote Device is still waiting for the fifth telegram (with IDX 3).
c) The actor sends a different SYS_EX message within the chain period with same SEQ and with 4 telegrams.
d) When the first telegram is received (IDX 0) the remote device can assume that it is dealing with new message. Data will be overwritten and error state can occur.

When we discard a message when a telegram with already received IDX number is been received again, we can avoid this failure. This will work only when we declare that every telegram is transmitted only once.

4.2.3 Query Status

With the query status command the actor can query debug information about the Remote Management on the Remote Device. He can query this information:

- If security code is set or not (0 – false, 1 - true)Last remote command function number (RMCC or RPC)

- The last remote command code is same as the function code of the command.
- Last function return code (OK, error, etc.)
- Last function return code is relevant to function when merge was successful, because only then a function gets executed. When merge failed then last function return code give a clue why the merge failed (0x09, 0x0A, 0x0B, 0x0C).
- Telegram merge info – if merge successful or error in receive
- Telegram merge info is information if the last message was received OK. When the value is 0 then it is OK – the last message was completely received and merged if needed.
- If value is >0 the merge was not successful, the message was not received. In this case the value is the last SEQ that was received by the remote device.
- Only if the last merge was successful we can evaluate the last command function number. If last merge failed we cannot know if the last command function code is up to date.

The last function number, return code and merge info belong to the most recent command and merge process.

The last function return code is listened in table bellow.

Table 2 Function return codes

Status name	Code number
OK	0x00
Wrong target ID	0x01
Wrong unlock code	0x02
Wrong EEP	0x03
Wrong manufacturer ID	0x04
Wrong data size	0x05
No code set	0x06
Not sent	0x07
RPC failed	0x08
Message time out	0x09
Too Long Message	0x0A
Message part already received	0x0B
Message part not received	0x0C
Address out of range	0x0D
Code data size exceeded	0x0E
Wrong data	0x0F

4.3 REMAN concept and repeating

When using REMAN telegrams the telegram repeater status should be considered. When the repeater status of the telegrams is 0 the REMAN telegrams will be repeated by repeaters. It can happen that in this can lead to an unexpected behaviour.

Consider the following scenario:

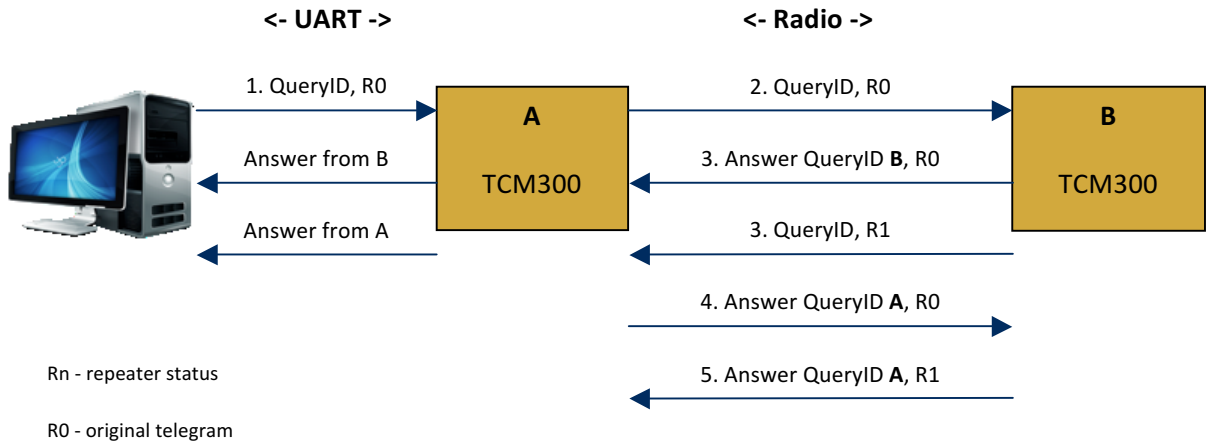
An application sends a QueryID telegram from PC with repeater status set to 0.

Setup:

a) PC – Computer running application like DolphinView
b) Device A – radio to serial gateway
c) Device B – device we want to control per remote management

Step-by-step:

a) DolhinView send QueryID telegram over serial, repeater status is 0.
b) Device A receives serial telegram and forwards it to radio.
c) Device B replays with QueryID Answer telegram.
d) Device B repeats the QueryID, increases repeater status bits to 1 (R1).
e) Device A receives the QueryID telegram.
f) Device A replays witch QueryID Answer telegram.
g) Device A repeats the QueryID Answer, increases repeater status.
h) PC receives QueryID answer from device B and device A.



The problem with this scenario is that we actually queried also our gateway. There are several solutions for this problem:

- When sending remote management telegrams from an application set the status byte to 0xF – Telegram must not be repeated. This way the repeaters will not repeat the telegrams again. In some scenarios this also reduces the radio traffic.
- Use gateway software without remote management possibilities
- Filter the remote management answers with gateway ID on application side.
- Recommended is always use the first solution and set the status of REMAN telegrams to 0xF.

5 RMCC and RPC structure definitions

5.1 Remote Management Commands (RMCC)

Note that all the telegrams in this chapter are represented in ADT encapsulated form. ADT encapsulation is used to makes possible transmit telegrams with UNICAST i.e. with a certain destination ID (see chapter XY). To transmit telegrams as a broadcast either the ADT with the broadcast destination ID 0xFFFFFFFF or the SYS_EX telegrams without ADT encapsulation can be sent.

Function code	RMCC – Remote Management Control Commands
0x000	RESERVED
0x001	UNLOCK
0x002	LOCK
0x003	Set CODE
0x004	Query ID
0x005	Action command
0x006	Ping command
0x007	Query function command
0x008	Query status

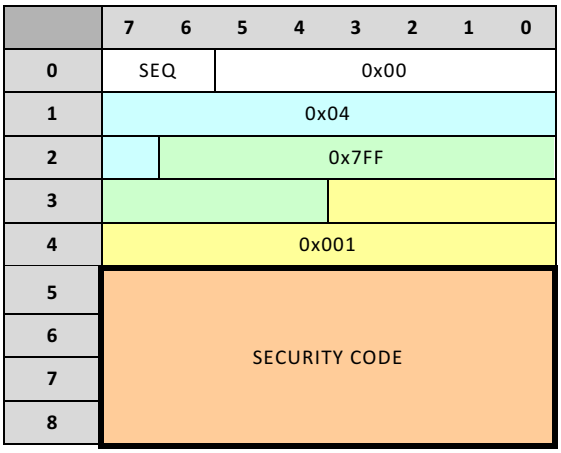
Table 3 RMCC function codes

In the following description only the payload is shown independent of the radio or serial protocol.

5.1.1 Unlock

UNLOCK		
Function code		0x001
Manufacturer Id		0x7FF
Data length		4 bytes
Data content	Security code 4 bytes	
Unicast		yes
Broadcast		yes
Device responses to command		no
Status return code		
OK	0x00	
Wrong target ID	0x01	
Wrong unlock code	0x02	
Wrong manufacturer ID	0x04	
Wrong data size	0x05	
No code set	0x06	

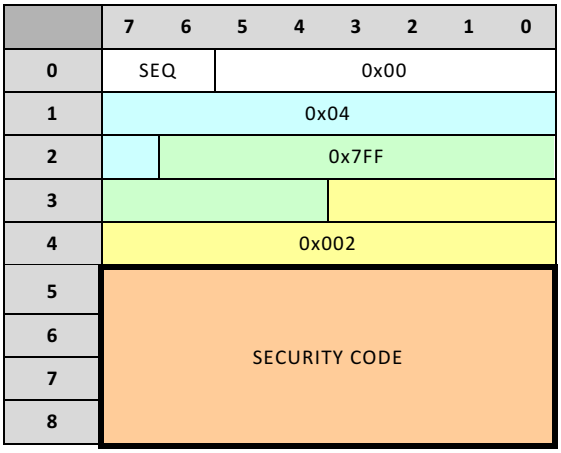
Table 4 Unlock telegram



5.1.2 Lock

LOCK		
Function code		0x002
Manufacturer Id		0x7FF
Data length		4 bytes
Data content	Security code 4 bytes	
Unicast		yes
Broadcast		yes
Device responses to command		no
Status return code		
OK	0x00	
Wrong target ID	0x01	
Wrong unlock code	0x02	
Wrong manufacturer ID	0x04	
Wrong data size	0x05	
No code set	0x06	

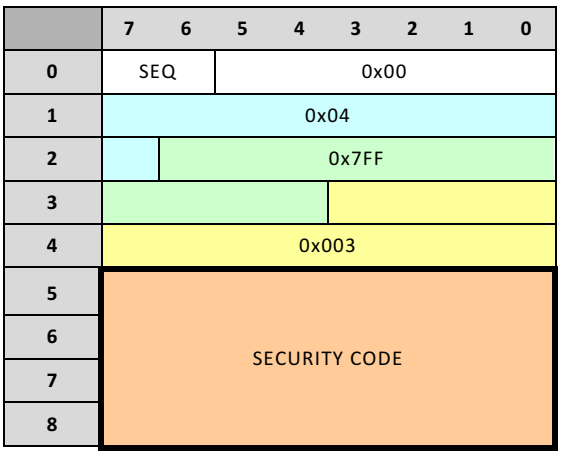
Table 5 Lock telegram



5.1.3 Set code

SET CODE		
Function code		0x003
Manufacturer Id		0x7FF
Data length		4 bytes
Data content	Security code	4 bytes
Unicast		yes
Broadcast		yes
Device responses to command		no
Status return code		
OK	0x00	
Wrong target ID	0x01	
Wrong manufacturer ID	0x04	
Wrong data size	0x05	

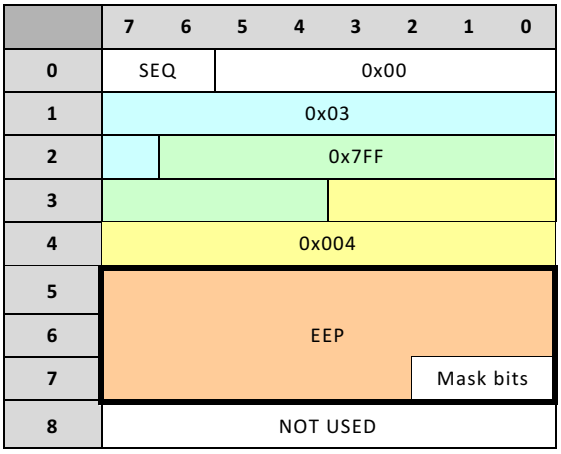
Table 6 Set code telegram



5.1.4 Query ID

QUERY ID		
Function code		0x004
Manufacturer Id		0x7FF
Data length		3 bytes
Data content	Desired EEP	21 bits
	Mask bits (see chapter 2.2)	3 bits
Unicast		no
Broadcast		yes
Device responses to command		yes
Status return code		
OK	0x00	
Wrong EEP	0x03	
Wrong manufacturer ID	0x04	
Wrong data size	0x05	
Not sent	0x07	
Wrong data	0x0F	

Table 7 Query Id telegram



Note:

From EEP 3.0 the FUNC and TYPE have the length of 8 bits. For EEP's with FUNC > 0x3F, or TYPE > 0x7F, this RMCC will not work. It is recommended to use RECOM RPC GetProductID instead of Query ID.

5.1.4.1 Query ID Answer

QUERY ID ANSWER		
Function code		0x604
Manufacturer Id		Device Manufacturer ID
Data length		3 bytes
Data content	Desired EEP	21 bit
	Mask bits = 0x00	3 bits
Unicast		yes
Broadcast		no

Table 8 Query ID Answer telegram

	7	6	5	4	3	2	1	0
0	SEQ				0x00			
1	0x03							
2		Device Manufacturer ID						
3								
4	0x604							
5	EEP							
6								
7							0x00	
8	NOT USED							

5.1.4.2 Query ID Answer Extended

QUERY ID ANSWER EXTENDED		
Function code		0x704
Manufacturer Id		Device Manufacturer ID
Data length		4 bytes
Data content	Desired EEP	21 bit
	Mask bits = 0x00	3 bits
	Locked by other manager	1 bit
Unicast		yes
Broadcast		no

Table 9 Query ID Answer telegram Extended

	7	6	5	4	3	2	1	0
0	SEQ				0x00			
1	0x03							
2		Device Manufacturer ID						
3								
4	0x704							
5	EEP							
6								
7							0x00	
8	*	NOT USED						

- * Locked by other manager (0 – false, 1 - true)

5.1.5 Action

ACTION	
Function code	0x005
Manufacturer Id	0x7FF
Data length	0 bytes
Unicast	yes
Broadcast	yes
Device responses to command	no
Status return code	
OK	0x00
Wrong target Id	0x01
Wrong manufacturer Id	0x04

Table 10 Action telegram

	7	6	5	4	3	2	1	0
0	SEQ				0x00			
1	0x00							
2		0x7FF						
3								
4	0x005							
5	NOT USED							
6								
7								
8								

5.1.6 Ping

PING	
Function code	0x006
Manufacturer Id	0x7FF
Data length	0 bytes
Unicast	yes
Broadcast	no
Device responses to command	yes
Status return code	
OK	0x00
Wrong target Id	0x01
Wrong manufacturer Id	0x04
Not sent	0x07

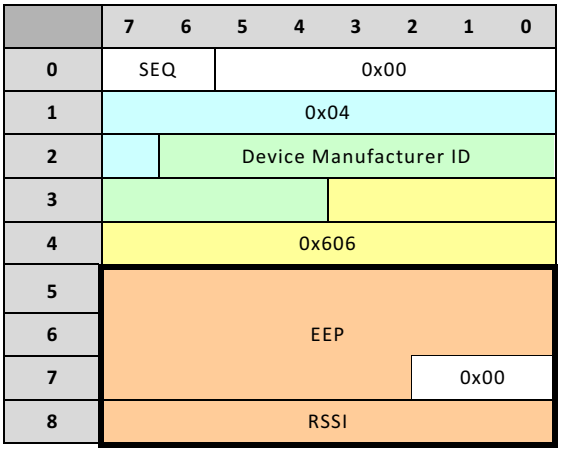
Table 11 Ping telegram

	7	6	5	4	3	2	1	0
0	SEQ		0x00					
1	0x00							
2		0x7FF						
3								
4	0x006							
5	NOT USED							
6								
7								
8								

5.1.6.1 Ping answer

PING ANSWER		
Function code		0x606
Manufacturer Id		Device Manufacturer ID
Data length		4 bytes
Data content	The EEP of the remote device	21 bits
	Mask Bits = 0x00	3 bits
	RSSI of received telegram by device	1 byte
Unicast		yes
Broadcast		no

Table 12 Ping answer telegram



Note:

From EEP 3.0 the FUNC and TYPE have the length of 8 bits. For EEP's with FUNC > 0x3F, or TYPE > 0x7F, this RMCC will not work. In this case, the ping answer will not contain an EEP.

5.1.7 Query function

QUERY FUNCTION	
Function code	0x007
Manufacturer Id	0x7FF
Data length	0 bytes
Unicast	yes
Broadcast	no
Device responses to command	yes
Status return code	
OK	0x00
Wrong target Id	0x01
Wrong manufacturer Id	0x04
Not sent	0x07

Table 13 Query function telegram

	7	6	5	4	3	2	1	0
0	SEQ				0x00			
1	0x00							
2		0x7FF						
3								
4	0x007							
5	NOT USED							
6								
7								
8								

5.1.7.1 Query function answer

QUERY FUNCTION ANSWER	
Function code	0x607
Manufacturer Id	Device Manufacturer ID
Data length	n * 4 bytes
Data content	Supported extended function list with manufacturer ID
One list entry	Function number 2 bytes Manufacturer Id 2 bytes
Unicast	yes
Broadcast	no

n is the entry count of the RPC list

Table 14 Query function answer telegram IDX = 0

	7	6	5	4	3	2	1	0
0	SEQ				0x00			
1	0x04 * n							
2		Device Manufacturer ID						
3								
4	0x607							
5	NOT USED							
6	FUNCTION NUMBER							
7	NOT USED							
8	MANUFACTURER ID							

n is the entry count of the RPC list

5.1.7.2 Query function answer telegram IDX = 1

	7	6	5	4	3	2	1	0
0	SEQ				0x01			
1	NOT USED							
2	FUNCTION NUMBER							
3	NOT USED							
4	MANUFACTURER ID							
5	NOT USED							
6	FUNCTION NUMBER							
7	NOT USED							
8	MANUFACTURER ID							

NOTE: Query function answer length depends on the RPC list. All next list entries are transmitted in following telegrams and merged at target.

5.1.8 Query status

QUERY STATUS	
Function code	0x008
Manufacturer Id	0x7FF
Data length	0 bytes
Unicast	yes
Broadcast	yes
Device responses to command	yes
Status return code	
OK	0x00
Wrong target Id	0x01
Wrong manufacturer Id	0x04
Not sent	0x07

Table 15 Query status telegram

	7	6	5	4	3	2	1	0
0	SEQ		0x00					
1	0x00							
2		0x7FF						
3								
4	0x008							
5	NOT USED							
6								
7								
8								

5.1.8.1 Query status answer

QUERY STATUS ANSWER		
Function code		
Manufacturer Id		Device Manufa
Data length		4 bytes
Data content	Code set flag Last SEQ Last function code Last function return code Not used	1 bit 2 bits 12 bits 8 bits 9 bits
Unicast		
Broadcast		

Table 16 Query status answer description

	7	6	5	4	3	2	1	0
0	SEQ				0x00			
1	0x04							
2		Device Manufacturer ID						
3								
4	0x608							
5	*	NOT USED						LAST SEQ
6	NOT USED							
7	LAST FUNCTION NUMBER							
8	LAST FUNCTION RETURN CODE							

* - Code set flag (0 – false, 1 - true)

5.2 Remote Procedure Calls (RPC)

5.2.1 RPC Remote learn

Using this command the learn mode of a device can be started or stopped.

RPC - Remote learn	
Function code	0x201
Manufacturer Id	0x7FF
Data length	4 bytes
Data content	see below
Unicast	yes
Broadcast	yes
Device responses to command	no
Status return code	
OK	0x00
Wrong data size	0x05
RPC failed	0x08

	7	6	5	4	3	2	1	0
0	SEQ				0x00			
1	0x04							
2		Device Manufacturer ID						
3								
4	0x201							
5	EEP							
6								
7							Mask bits	
8	Flag							

Note:

From EEP 3.0 the FUNC and TYPE have the length of 8 bits. For EEP's with FUNC > 0x3F, or TYPE > 0x7F, this RPC will not work. It is recommended to set the Mask bits to 0x00 and not providing an EEP.

Offset	Size	Data	Description	Valid Range	Scale	Unit
0	21	EEP (R-ORG-FUNC-TYPE)	Determines the device type to learn in, all other devices learn telegrams are ignored. To ignore EEP control the mask bits has to be set to 0	…	…	
21	3	Mask bits		Enum: 0b000 – Ignore EEP.		
24	8	Flag	learn flag, determines different behavior of the learn procedure	Enum:		
				0x00: RESERVED		
				0x01: Start learn		
				0x02: Next channel		
				0x03: Stop learn		
				0x04: SMART ACK – Start simple learn mode		
				0x05: SMART ACK – Start advanced learn mode		
				0x06: SMART ACK – Stop learn		

5.2.2 RPC Remote flash write

Using this command, the flash of a device can be written.

RPC - Remote flash write	
Function code	0x203
Manufacturer Id	0x7FF
Data length	4+N bytes
Data content	see below
Unicast	yes
Broadcast	yes
Device responses to command	no
Status return code	
OK	0x00
Wrong data size	0x05
RPC failed	0x08
Code address out of range	0x0D
Data size exceeded	0x0E

	7	6	5	4	3	2	1	0
0	SEQ				0x00			
1	0x04 + N							
2		Device Manufacturer ID						
3								
4	0x203							
5	Flash Memory Address							
6								
7	Number N of bytes							
8								

Offset	Size	Data	Description
0	16	Flash Memory Address	Destination address where the data shall be written
16	16	Number of bytes	Number N of bytes to be transferred and written to the flash
32	N*8	Data	Data to be transferred and written to the flash

5.2.3 RPC Remote flash read

Using this command the flash can be read from the application. The data requested is transmitted in several RPC telegrams with function code 0x804.

RPC - Remote flash read	
Function code	0x204
Manufacturer Id	0x7FF
Data length	4 bytes
Data content	see below
Unicast	yes
Broadcast	no
Device responses to command	yes
Status return code	
OK	0x00
Wrong data size	0x05
RPC failed	0x08
Code address out of range	0x0D
Data size exceeded	0x0E

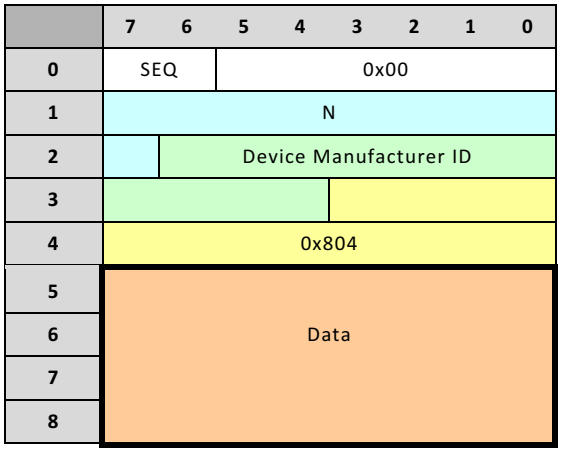
	7	6	5	4	3	2	1	0
0	SEQ				0x00			
1	0x04							
2		Device Manufacturer ID						
3								
4	0x204							
5	Flash Memory Address							
6								
7	Number N of bytes							
8								

Offset	Size	Data	Description
0	16	Flash Memory Address	Start address where the data shall be read from
16	16	Number of bytes	Number of bytes to be transferred and read from the flash

5.2.3.1 RPC Remote flash read answer

Using this command the flash can be read from the application. The data requested is transmitted in several RPC telegrams with function code 0x804.

RPC - Remote flash read answer	
Function code	0x804
Manufacturer Id	0x7FF
Data length Data content	N bytes see below
Unicast	yes
Broadcast	no



Offset	Size	Data	Description
0	N*8	Data	Data read from flash

5.2.4 RPC SMART ACK read settings

Using this command the SMART ACK settings and learn tables can be read from the device. The Setting type filled determines what type of data is requested. The data requested are transmitted in several RPC telegrams with function code 0x805 or 0x806.

RPC - SMART ACK read settings	
Function code	0x205
Manufacturer Id	0x7FF
Data length	1 byte
Data content	see below
Unicast	yes
Broadcast	no
Device responses to command	yes
Status return code	
OK	0x00
Wrong data size	0x05
RPC failed	0x08
Data size exceeded	0x0E

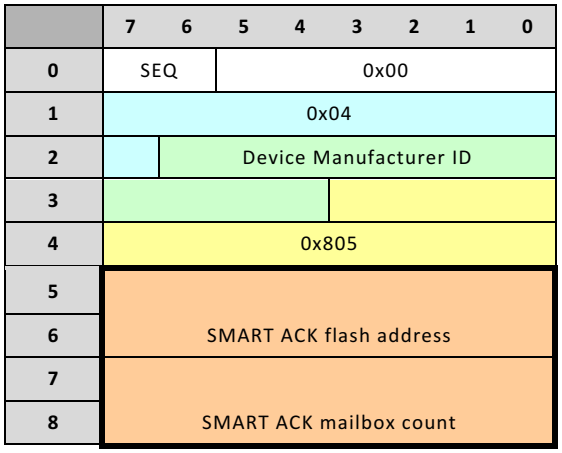
	7	6	5	4	3	2	1	0
0	SEQ				0x00			
1	0x01							
2		Device Manufacturer ID						
3								
4	0x205							
5	Setting type							
6	Not used							
7								
8								

Offset	Size	Data	Description	Valid Range	Scale	Unit
0	8	Setting type	Type of settings to read	Enum:		
				0x00: RESERVED		
				0x01: Mailbox settings		
				0x02: Learned sensor – read the ID table of sensors in the controller		

5.2.4.1 RPC SMART ACK read settings – Mailbox settings answer

This is the answer of the RPC 0x205 with settings type 0x01.

RPC - SMART ACK read settings – Mailbox settings answer	
Function code	0x805
Manufacturer Id	0x7FF
Data length	4 bytes
Data content	see below
Unicast	yes
Broadcast	no



Offset	Size	Data	Description
0	16	SMART ACK flash address	Address where the SMART ACK settings are stored
16	16	SMART ACK mailbox count	Number of mailboxes stored in flash

5.2.4.2 RPC SMART ACK read settings – Learned sensor answer

This is the answer of the RPC 0x205 with settings type 0x02.

N is the number of entries: {SensorID, ControllerID, Mailbox index}

RPC - SMART ACK read settings – Learned sensor answer	
Function code	0x806
Manufacturer Id	0x7FF
Data length Data content	N*9 bytes see below
Unicast	yes
Broadcast	no

	7	6	5	4	3	2	1	0
0	SEQ			0x00				
1	0x09*N							
2		Device Manufacturer ID						
3								
4	0x806							
5	N*9 bytes data as described below							
6								
7								
8								

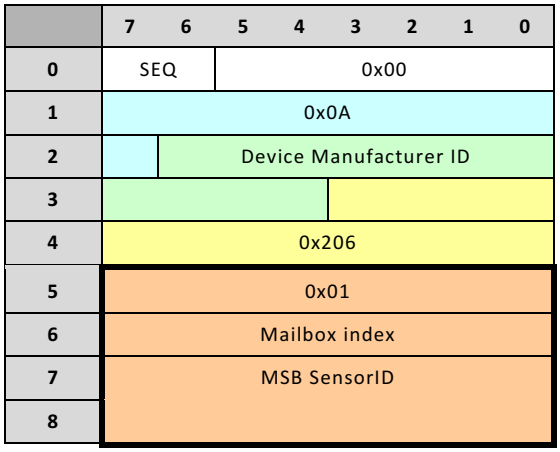
Offset	Size	Data	Description
N*0	32	SensorID	EURID of sensor
N*32	32	ControllerID	EURID of controller
N*64	8	Mailbox index	Index of mailbox

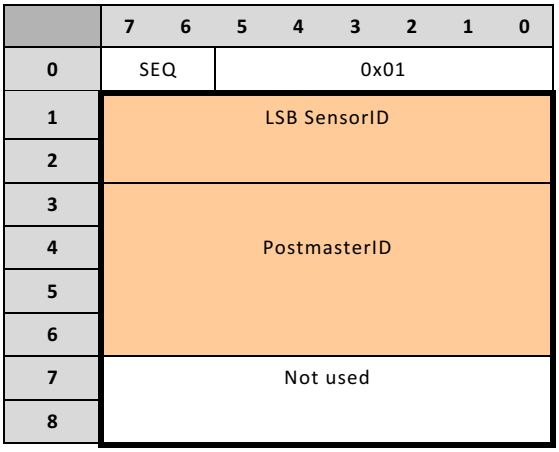
5.2.5 RPC SMART ACK write settings

Using this command different type of data can be transmitted to the SMART ACK devices. This command is useful when the SMART ACK device has to be configured remotely. The structure of the data transmitted is depends on the Operation Type field.

RPC - SMART ACK write settings	
Function code	0x206
Manufacturer Id	0x7FF
Data length	10 bytes
Data content	see below
Unicast	yes
Broadcast	no
Device responses to command	no
Status return code	
OK	0x00
Wrong data size	0x05
RPC failed	0x08

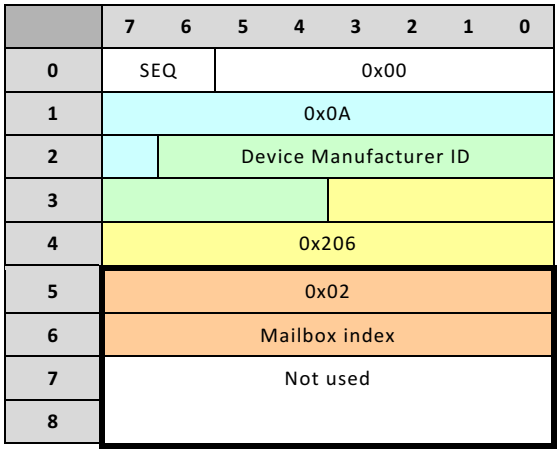
5.2.5.1 Operation Type = 0x01: Add mailbox (only controller)

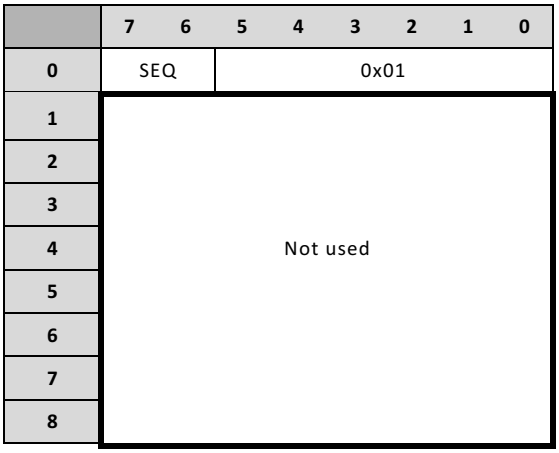




Offset	Size	Data	Value	Description
0	8	Operation Type	0x01	Add mailbox (only controller)
8	8	Mailbox index		Index of mailbox
16	32	SensorID		EURID of sensor
48	32	PostmasterID		EURID of postmaster

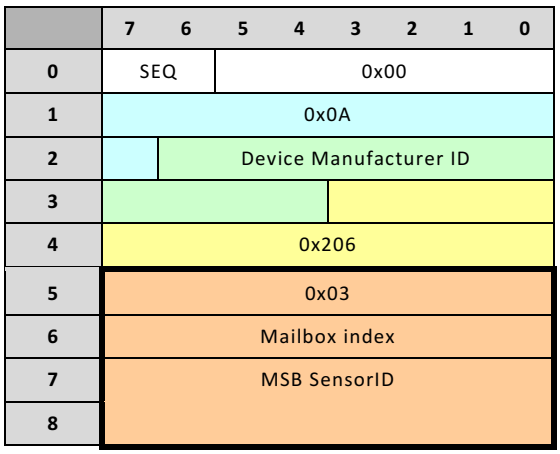
5.2.5.2 Operation Type = 0x02: Delete mailbox

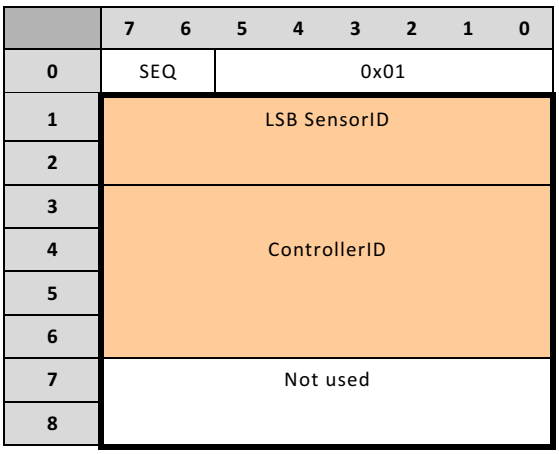




Offset	Size	Data	Value	Description
0	8	Operation Type	0x02	Delete mailbox
8	8	Mailbox index		Index of mailbox
16	64	Not Used (=0)		

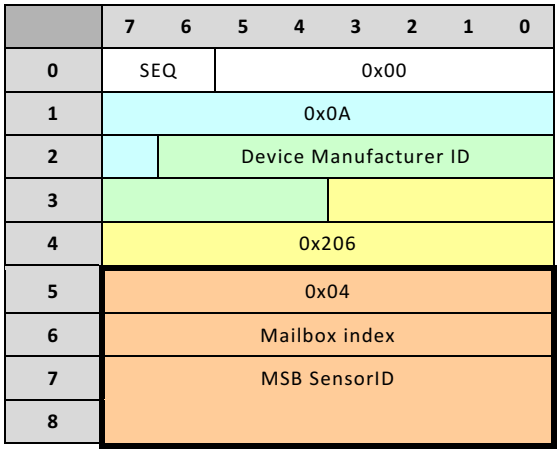
5.2.5.3 Operation Type = 0x03: LearnIn (only controller)

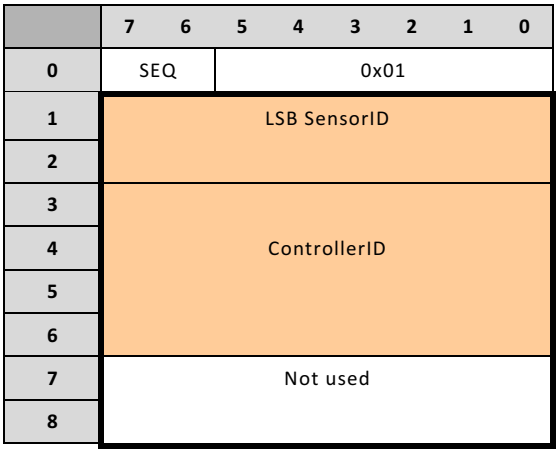




Offset	Size	Data	Value	Description
0	8	Operation Type	0x03	LearnIn (only controller)
8	8	Mailbox index		Index of mailbox
16	32	SensorID		EURID of sensor
48	32	ControllerID		EURID of controller

5.2.5.4 Operation Type = 0x04: LearnOut mailbox (only controller)





Offset	Size	Data	Value	Description
0	8	Operation Type	0x04	LearnOut (only controller)
8	8	Mailbox index		Index of mailbox
16	32	SensorID		EURID of sensor
48	32	ControllerID		EURID of controller

6 Important parameters

Table 17 Data length

Maximum number of SYS_EX message parts	64
Maximum length of transferred data	508 bytes

Table 18 Function Numbers

Available function numbers	(0x000 – 0xFFF)	4096
Reserved	(0x000)	1
Commands RMCC	(0x001 – 0x1FF)	511
Commands RPC	(0x200 – 0x5FF)	1024
Answers	(0x600 – 0xFFF)	2560

Table 19 Security code

Available security codes	(0x00000000 – 0xFFFFFFFF)	2^{32}
Reserved	(0x00000000), (0xFFFFFFFF)	2
Free	(0x00000001 – 0xFFFFFFFE)	2^{32} minus 2

Table 20 Periods

Chain period	1 s
Broadcast delay interval	0 s – 2 s
Power up unlock period	5 min
Unlock period	5 min
Security period	30 s
Attempt period	30 s

7 Reman & Encrypted communication

In this chapter is described how the Remote commissioning communication between a manager and managed device is encrypted. For complete understanding and usage the Security Specification shall be considered[1].

7.1 Operation and Maintenance Keys

The operation key is the security Key, SLF and RLC, used for normal operation telegrams and messages (E.g. the transmitted EEP, GP and Signal telegrams).

The maintenance keys are used for SEC_MAN telegrams and messages. A device using any sort of secure maintenance operation (e.g. recom), shall support at least one maintenance key. A device can use up to 15 maintenance key, for allowing a higher granularity of access rights. It would need to check after receiving a full SEC_MAN messages that the requested operation is valid for the selected key.

Informing the application that request was not permitted or the access rights are not enough is not part of this document. Please refer to the corresponding product documentations.

7.2 SEC_MAN 0x34 -Maintenance Security

Maintenance Security messages have the main purpose to send secured ReMan/Recom messages and other maintenance messages. The maintenance telegram allow a higher granularity of access rights, as different maintenance keys can be used for different access rights.

All transmitted telegrams are using the following SLF:

- 24 bit RLC, transmitted over air (to allow multiple controllers and keep message overhead short)
- 3 Byte CMAC
- VAES

The following ReMan changes are necessary for using SEC_MAN instead of normal ReMan/ ReCom

- Lock/Unlock and set code are not used. Authentication happens via Session start command.
- Adding a „Session start close“ command
- Response Type additions:

[1] See: https://www.enocean-alliance.org/specifications/

- Season is already opened by another controller
- Selected maintenance key does not have the sufficient access rights.

7.2.1 Telegram structure

The SEC_MAN telegrams have the structure described in the following table and should be send with destination ID.

SEC_MAN	Key	Type	data field	EURID	status	Check sum
1 byte	4 bit	4 bit	N bytes.	4 bytes	1 byte	1 byte
	1 byte					

Table 21 SEC_MAN Telegram Structure

Key Value	Meaning
0x0	Reserved
0x1	Maintenance Key 1 (Standard)
…	
0xF	Maintenance Key 15

Table 22 SEC_MAN Key value

Type Value:

- 0x00 =Single Data
  Single secure data, with payload less than is required for chaning
- 0x01 = SEC_MAN_CHAINED
  Same chaining secured chained telegrams. See security specification.
- 0x02 = SEC_SYS_EX
  SYS_EX chaining and SYS_EX messages are encapsulated. The “payload” decreases 1 byte for each telegram per message and the last telegram needs to contain the RLC and the CMAC. Data length, manufacturer id and function are transmitted in clear text, to allow a fast discard, e.g. the RPC is not supported. As most devices are embedded devices with limited memory capacity, it is highly recommended that only one chained message is send at a time (To or from a device).

- 0x3-0xF Reserved

7.2.2 Examples

All examples are based on the limited payload length using ERP1 with destination ID. For ERP2 the maximal payload length for each telegram can be longer.

7.2.2.1 Type 0x00 Single data message

Expected decrypted data:
0x54

Used encryption KEY:
0x45 0x4f 0x54 0x45 0x53 0x54 0x4b 0x45 0x59 0x59 0x45 0x41 0x48 0x21 0x5c 0x30

RLC: **0x01 0x02 0x03**

Calculated data:
0x81

CMAC:
0x23 0xCD 0x25

RORG	Key	Type	Payload	RLC	CMAC
SEC_ MAN	**4 bit**	**4 bit**	**1 byte**	**3 byte**	**3 byte**
34	**1**	**1**	**81**	**01 02 03**	**23 CD 25**

7.2.2.2 Type 0x01 Chained message

Expected decrypted data:
0x01 0x02 0x3 0x04 0x05 0x06 0x07 0x08 0x09 0x0A 0x0B 0x0C 0x0D 0x0E 0x0F 0x10 0x11

Used encryption KEY:
0x45 0x4f 0x54 0x45 0x53 0x54 0x4b 0x45 0x59 0x59 0x45 0x41 0x48 0x21 0x5c 0x30

RLC: **0xAA 0xBB 0xCC**

Calculated data:
0x5D 0xA0 0xD6 0xDB 0x23 0x29 0x4B 0xFC 0xCD 0x0A 0x2F 0xD4 0x2D 0xBF 0x26 0xAD 0xF8

CMAC:
0xE5 0xD9 0xFA

IDX = 0

RORG	Key	Type	SEQ	IDX	Payload Date length	Payload
SEC_ MAN	4 bit	4 bit	2 bit	6 bit	2byte	5 byte
34	1	1	1	0	00 11	5D A0 D6 DB 23

IDX = 1

RORG	Key	Type	SEQ	IDX	Payload
SEC_ MAN	4 bit	4 bit	2 bit	6 bit	7 byte
34	1	1	1	1	29 4B FC CD 0A 2F D4

IDX = 2

RORG	Key		SEQ	IDX	Payload	RLC
SEC_MAN	4 bit	4bit	2 bit	6 bit	5 Byte	2 byte (of 3)
34	1	1	1	2	2D BF 26 AD F8	AA BB

IDX = 3

RORG	Key		SEQ	IDX	RLC	CMAC
SEC_MAN	4 bit	4bit	2 bit	6 bit	1 byte(of 3)	3 Byte
34	1	1	1	3	CC	E5 D9 FA

7.2.2.3 ***Type 0x02 Sys-EX examples:***

Example for ERP1 and sending ReMan Message "Query ID":

Manufacturer ID :0x7FF
FnCode: 0x004

Expected decrypted data:
0x00 0x00 0x00

Used encryption KEY:
0x45 0x4f 0x54 0x45 0x53 0x54 0x4b 0x45 0x59 0x59 0x45 0x41 0x48 0x21 0x5c 0x30

RLC: **0x46 0x43 0x4B**

Calculated data:
0xC2 0xBD 0x6F

CMAC:
0x9B 0x71 0xE7

IDX = 0

RORG	Key		SEQ	IDX	Date length	MAN_ID	Fn	EEP+mask bits
SEC_MAN	4 bit	4 bit	2 bit	6 bit	9bit	11bit	12bit	3 byte
34	1	2	2	0	03	7FF	04	C2 BD 6F

IDX = 1

RORG	Key		SEQ	IDX	RLC	CMAC
SEC_MAN	4 bit	4bit	2 bit	6 bit	3 byte	3 Byte
34	1	2	2	1	46 43 4B	9B 71 E7

Example 2 for ERP1 and sending Recom Message "Get Link Table Meta DataResponse":

Manufacturer ID :0x7FF
FnCode: 0x804

Expected decrypted data:
0xF0 0x05 0x01 0x10 0x05

Used encryption KEY:
0x45 0x4f 0x54 0x45 0x53 0x54 0x4b 0x45 0x59 0x59 0x45 0x41 0x48 0x21 0x5c 0x30

RLC: **0x4d 0x45 0x49**

Calculated data:
0x2C 0xC4 0xB6 0xD3 0x2E

CMAC:
0x7A 0xBB 0x51

IDX = 0

RORG	Key		SEQ	IDX	Date length	MAN_ID	Fn	Payload
SEC_MAN	4 bit	4 bit	2 bit	6 bit	9bit	11bit	12bit	3 byte
34	1	2	2	0	03	7FF	810	2C C4 B6

IDX = 1

RORG	Key		SEQ	IDX	Payload	RLC	CMAC
SEC_MAN	4 bit	4bit	2 bit	6 bit	2 byte	3 byte	2 Byte
34	1	2	3	1	D3 2E	4D 45 49	7A BB

IDX = 2

RORG	Key		SEQ	IDX	CMAC
SEC_MAN	4 bit	4bit	2 bit	6 bit	1 Byte
34	1	2	3	2	51

7.3 ReMan alterations

The ReMan “Lock”/”Unlock” and “Set Code” mechanism of the ReMan are obsolete for secure ReMan and shall not be used anymore. In Secure ReMan a session management is introduced, the “Start Session” and “Close Session” commands are added. “Lock”/”Unlock” and “Code” commands shall not be used in Secure ReMan. The non-secured SYS_EX are not useable if Secure ReMan messages are available.

The whole session management is simplified to the states “OPEN” or “CLOSED”.

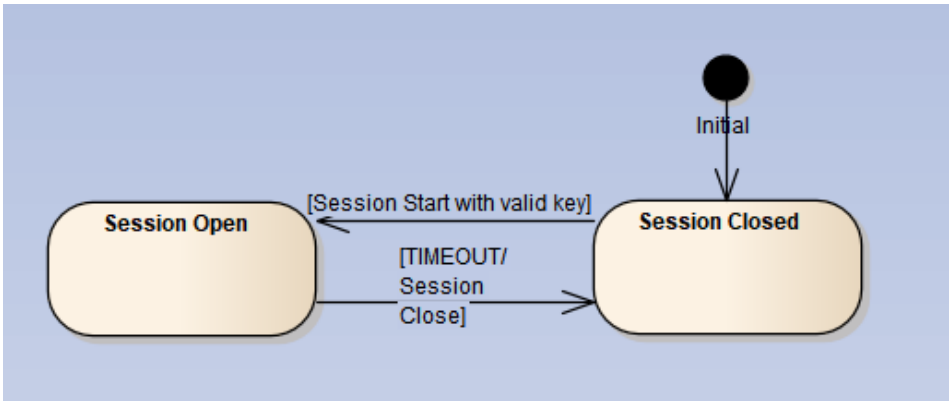


Figure 11 Session management

If the session is closed, a controller can start a session with the “Start Session” command.

The session stays open until either the session timeout is reached or the session has been closed by a “Close session command”.
Each valid SEC_MAN sys ex type telegram from the controller which started the session resets the session timer to 0. The session timer is increased every second and if the session timer reaches 60 seconds, the session is automatically closed.

If a second controller sends a session open command and a session is already active, the session open reply should contain “session already” open. The session stays valid for the first controller and the second controller needs to wait until the session is closed to send ReMan/ ReCom commands.

Both session control commands shall be only accepted if transmitted encrypted within a SEC_MAN message.

7.3.1 Mandatory RMCCs for secure ReMan

Function code	RMCC – Remote Management Control Commands
0x009	Start Session Command
0x00A	Close Session Command
0x004	Query ID
0x005	Action command
0x006	Ping command
0x007	Query function command
0x008	Query status

Table 23 RMCC function codes for Secure ReMan

7.3.2 Start Session Command

The Start Session command is used by a controller which tries to start a ReMan session with a ReMan device.

Start Session Command	
Function code	0x009
Manufacturer Id	0x7FF
Data length	0
Encrypted only	yes
Data content	-
Addressed	Yes
Broadcast	No
Command has paired response	Yes

Table 24 Start Session Command

7.3.3 Start Session Command Reply

Start Session Command Reply		
Function code		0x609
Manufacturer Id		0x7FF
Data length		1
Encrypted only		yes
Data content		
	Response Code	1 Byte
Addressed		Yes
Broadcast		No

Table 25 Start Session Command Reply

Response Code (1 Byte):

- 0x00 Okay: the controller which requested the session controls the session
- 0x01 Another controller has currently a open Session

Note:

- If another controller has currently an open Session, it is recommended to try it again later.

Data structure:

	7	6	5	4	3	2	1	0
0	Response Code							

Table 26 Start Session Command Reply data structure

7.3.4 Close Session Command

When controller has finished configuring a device using ReMan/Recom commands the session should closed again with the close session command.

Close Session Command	
Function code	0x00A
Manufacturer Id	0x7FF
Data length	0
Encrypted only	yes
Data content	-
Addressed	Yes
Broadcast	No
Command has paired response	Yes

Table 27 Close Session Command

7.4 Status code extension

The new session management and possible right management need the addition of 2 new status responses:

Session is closed – 0x10: Currently there is no open session with this controller

Insufficient rights – 0x11: The used maintenance key does not have the right to execute the last command

Status name	Code number (Query Status response)
OK	0x00
Wrong target ID	0x01
Wrong unlock code	0x02
Wrong EEP	0x03
Wrong manufacturer ID	0x04
Wrong data size	0x05
No code set	0x06
Not send	0x07
RPC failed	0x08
Message time out	0x09
Too Long Message	0x0A
Message part already received	0x0B
Message part not received	0x0C
Address out of range	0x0D
Code data size exceeded	0x0E
Wrong data	0x0F
Session is closed	0x10
Insufficient rights	0x11

Table 28 Query Status Return codes

Parameter	Meaning
Session Timeout	60 seconds
SEC_MAN	0x34

Table 29 Defined constants