,

VELUX®
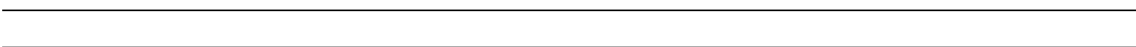
VELUX A/S Accessories
Bækgårdsvej 40, 6900 Skjern

Technical Specification for

KLF 200 API

io-homecontrol® Gateway

History

Version	Changes	Author	Date
3.14	Match KLF 200 firmware version 2.0.0.71	AHM	27-09-2018
3.15	API released for public use.	AHM	01-10-2018
3.16	TimeStamp parameter is uint32_t type in GW_NODE_STATE_POSITION_CHANGED_NTF. Parameter length bug fixed in GW_CS_GENERATE_NEW_KEY_NTF, GW_CS_RECEIVE_KEY_NTF and GW_CS_REPAIR_KEY_NTF. Missing parameter added in GW_GET_NODE_INFORMATION_NTF.	AHM	02-11-2018

Table of Contents

1 Welcome

Welcome to KLF 200 API. With the release of this API VELUX A/S would like to increase the possibility for very advanced users and professionals to control motorized VELUX roof windows, blinds and shutters. We have chosen to release the API with no technical limitation. It is simply open and available. It is free and without registration. Therefore, we also kindly ask for understanding that we cannot offer support with regards to using the API.

KLF 200 and its possibilities are regularly discussed in blogs and forums. It is possible that you can gain knowledge here and maybe find answers to your questions.

Using the API is on your own responsibility and liability. Please read through VELUX discharge of liability below.

2 VELUX liability

VELUX is only liable for connecting and/or integrating KLF 200 API to VELUX programs, products and/or services and not for any third-party programs, products and/or services. VELUX liability is regulated in the VELUX Guarantee, which is available at VELUX national homepages.

VELUX does not assume any liability for connection via the ethernet connetor.

VELUX does not provide any service or support for third parties' use of the API, programming, integration, interface etc. with KLF 200.

It is not possible for VELUX to trace or get an insight into third parties' data or communication with the KLF 200.

VELUX is not liable in case of any interruption of the operation of the KLF 200, including but not limited to solar powered products which are drained for power or other interruption.

The API does not provide burglary security or any security against illegal trespassing. VELUX is not liable for any theft, vandalism or damage which is due to illegal trespassing.

If you sell a solution based on KLF 200 API to a customer you are entirely responsible for the control. VELUX shall not help the customer to improve/change the functionality setup with the API and the VELUX Guarantee does not apply.

3 Introduction

KLF 200 io-homecontrol® gateway with KLF 200 API gives you a way to control io-homecontrol® nodes using a simple protocol over a local Ethernet network.

In addition to the functions to setup io-homecontrol® system offers KLF200 two different operating modes; Command handler and Scene handler.

- Command handler mode makes it possible to control each actuator in the system individually and it is possible to use some of the io-homecontrol® protocol features like priority and command originator. It is much more complex than scene mode and is only recommended if scene mode doesn't fulfil the required behaviour.

- A scene is characterized by that it defines a certain position (or level) of one or more actuators. All kind of actuators can be included and the position can be individual for all the actuators.
  Only one scene is active at a time and last until another scene is activated. Scene mode is by far the easiest way to use KLF200 and doesn't require io-homecontrol® protocol knowledge.

Figure 1 – KLF 200 photo.

4 Gateway interface

4.1 TCP/IP interface

The Ethernet module establishes a TCP/IP server listening at port 51200. Up to two sockets can be established at the same time on wired ethernet.
TCP/IP socket will be closed after 15 min, with no communication. The command GW_GET_STATE_REQ can be used to ping KLF200 from time to time, to keep the socked established.
TLS is used to encrypt communication. For now, the certificate is self-signed.

4.2 Gateway command frame

Command	Data
2 byte	0 to 250 bytes

Table 1 - Prototype of gateway command frame format.

4.2.1 Command parameter

Command parameter is an unsigned 16-bit integer. Network byte order are big endian, means that most significant byte is transferred first.
All the commands and the related numbers can be found in paragraph 15 "Appendix 3: List of Gateway commands" at page 107.

4.2.2 Data field

Command specific data field. Size can vary from 0 to 250 bytes. If the data field contains a 16-bit or 32-bit integer, it must be transferred with big endian byte order, means that most significant byte is transferred first.

4.3 Gateway command frame length

Length	Command	Data
1 byte	2 byte	0 to 250 bytes

Table 2 – Length parameter added to Gateway Command frame.

4.3.1 Length parameter

Length is the total length of frame shown in Table 1.

Length value	Description
0-2	Not defined
3	No data
4	1 byte of data
…	…
253	250 bytes of data
254-255	Not defined

Figure 2 - Length parameter description.

4.4 Transport layer

ProtocolID are first added in front of frame. And then a checksum is calculated and added to the end of frame.

ProtocolID	Length	Command	Data	Checksum
1 byte	1 byte	2 byte	0 to 250 bytes	1 byte

Table 3 – Transport layer frame format.

4.4.1 ProtocolID parameter

ProtocolID are always set to 0.

4.4.2 Checksum parameter

After last data byte, there is a Checksum byte. The Checksum are made by bitwise XOR all bytes from and including the ProtocolID parameter to last data byte.

4.5 SLIP wrapping

Before the frame can be transmitted, it must be packed into SLIP.
The Slip protocol indicates data start and end.

Indicates	Marker name	Number (dec)	Number (Hex)
Start and end of frame	END	192	0xC0
Byte stuffing	ESC	219	0xDB
ESC ESC_END means END data byte	ESC_END	220	0xDC
ESC ESC_ESC means ESC data byte	ESC_ESC	221	0xDD

Table 4 - Value of SLIP markers.

END	data	data	data	data	data	data	**END**

Table 5 - A frame packed in Slip.

This paragraph is only introduction to the SLIP technique, for a comprehensive description read the document RFC 1055. The document can be found at http://www.rfc-base.org/txt/rfc-1055.txt.
In Table 4 marker values used in this project is given. Table 5 shows a data block packet into SLIP, where the fields named data is a Gateway protocol frame.

Note: The size of a frame packed into SLIP may exceed 255 bytes, if the data contains END or ESC characters.

4.6 Standard communication and frame naming

All frames sent to KLF200 has suffix REQ. REQ is short for request. Each request frame will be acknowledged by a confirm frame. Confirm frames has suffix CFM. See sequence diagram in Figure 3.
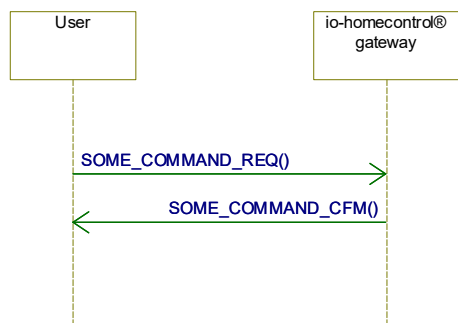
Figure 3 - Sequence diagram showing standard communication with REQ and CFM frames.

If the request involves communication with io-homecontrol® nodes, the REQ frame will first be acknowledged by CFM frame, then after the io-homecontrol® communication one or more NTF frames will also be returned. (NTF is an abbreviation for notify.) Time to NTF

frame can be from a few hundred milliseconds to several seconds. See sequence diagram in Figure 4.
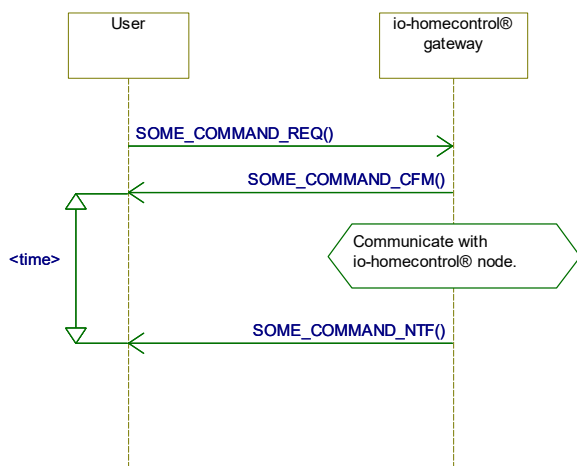
Figure 4 - Sequence diagram showing standard communication with REQ, CFM and NTF frames.

Deviations from the rules above

- If an error occurs, CFM frame, NTF frame or both can be replaced by an Error frame. See sequence diagram in Figure 5.
  Read more about the error frame in paragraph 10.6.1.1 at page 79.
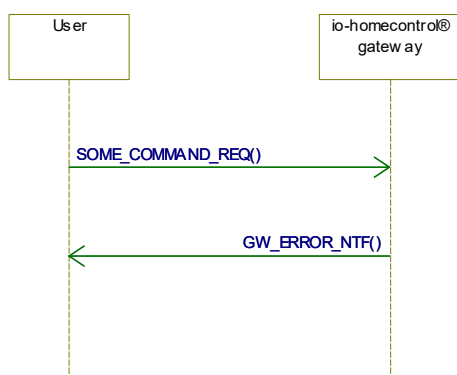
Figure 5 - Sequence diagram showing standard communication with REQ and Error frames.

- If a large scene list with more than 7 scenes, will be transferred in several GW_GET_SCENE_LIST_CFM frames. See paragraph 11.7 at page 91.

- If a large system table list with more than 10 nodes, will be transferred in several GW_CS_GET_SYSTEMTABLE_DATA_CFM frames. See paragraph 7.4 at page 24.

- If PGC is used, some status frames will be sent from gateway. See paragraph 7.12.4 at page 37.

- If GW_CS_CONTROLLER_COPY_CANCEL_NTF is used. See paragraph 7.8.4 at page 32.

5 Authentication

Client must authenticate after TLS socket has been established, using a password.
The default password is the same as Wi-Fi password written on back side of KLF200.
The user can change the password. The password is reset to Wi-Fi password if factory reset is requested.
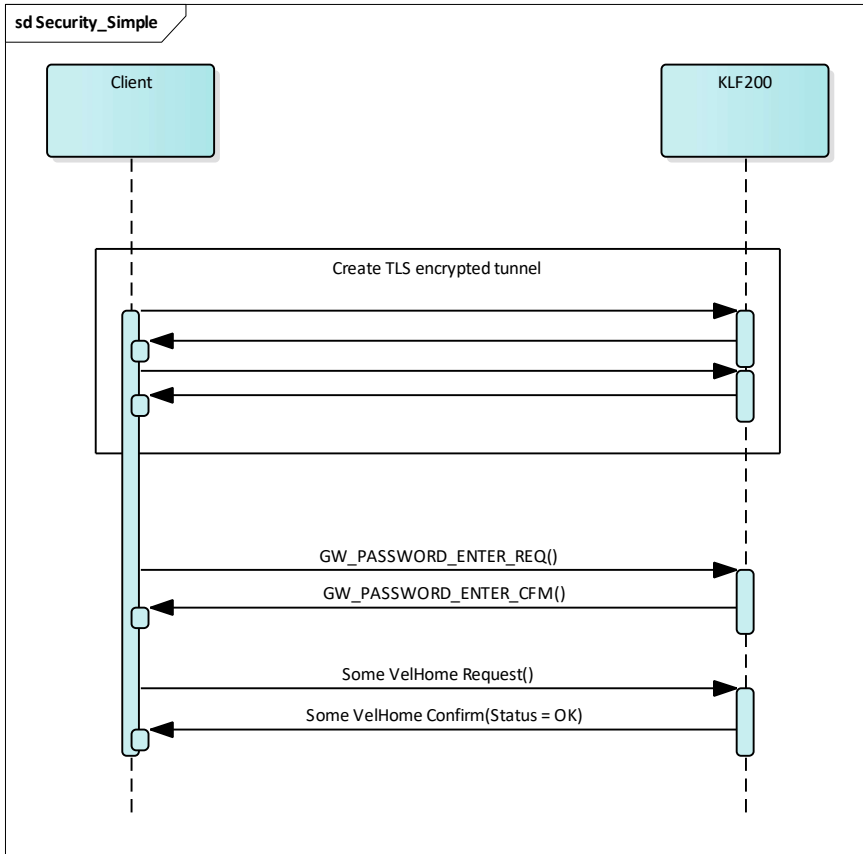
5.1.1 GW_PASSWORD_ENTER_REQ

Command	Data 1-32
GW_PASSWORD_ENTER_REQ	Password

Table 6 - GW_PASSWORD_ENTER_REQ frame format.

5.1.1.1 Password

The password parameter must contain a paraphrase followed by zeros.
Last byte of Password byte array must be null terminated.

5.1.2 GW_PASSWORD_ENTER_CFM

Command	Data 1
GW_PASSWORD_ENTER_CFM	Status

Table 7 - GW_PASSWORD_ENTER_CFM frame format.

5.1.2.1 Status

Status value	Description
0	The request was successful.
1	The request failed.

Table 8 - Status parameter
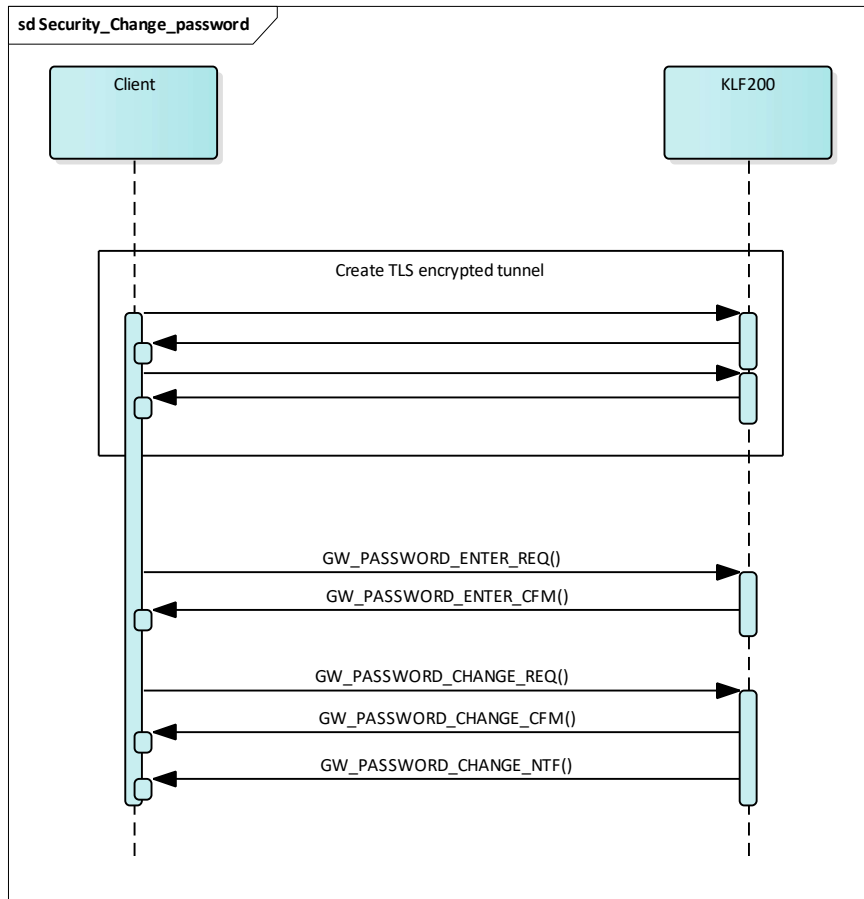
Figure 6 - Sequence diagram, change password.
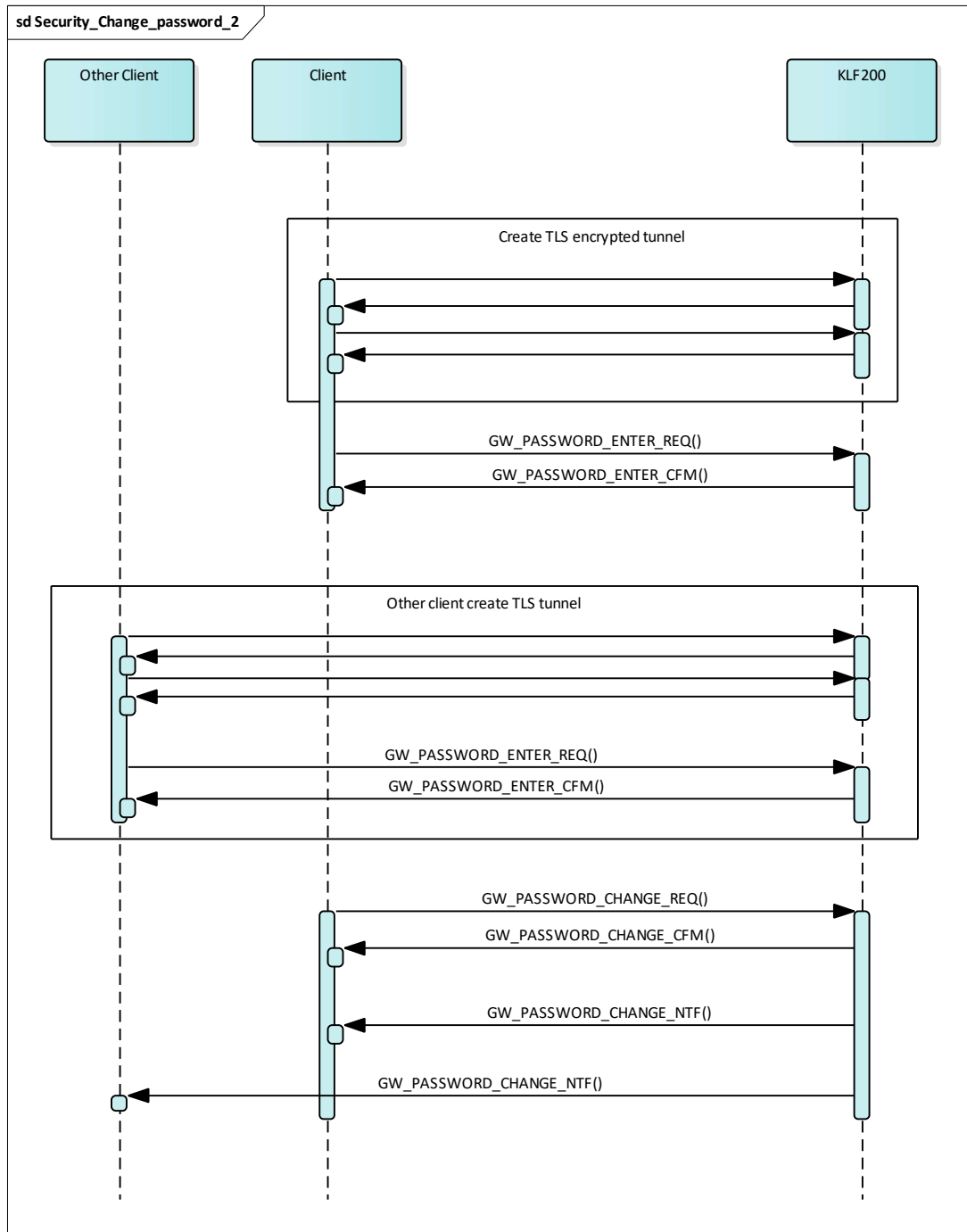
Figure 7 - Sequence diagram, change password and inform other client.

5.1.3 GW_PASSWORD_CHANGE_REQ

Command	Data 1-32	Data 33-64
GW_PASSWORD_CHANGE_REQ	CurrentPassword	NewPassword

Table 9 - GW_PASSWORD_CHANGE_REQ frame format.

5.1.3.1 CurrentPassword and NewPassword

See paragraph 5.1.1.1 for description.

5.1.4 GW_PASSWORD_CHANGE_CFM

Command	Data 1
GW_PASSWORD_CHANGE_CFM	Status

Table 10 - GW_PASSWORD_CHANGE_CFM frame format.

5.1.4.1 Status

Status value	Description
0	The request was successful.
1	The request failed.

Table 11 - Status parameter

5.1.5 GW_PASSWORD_CHANGE_NTF

Command	Data 1-32
GW_PASSWORD_CHANGE_NTF	NewPassword

Table 12 - GW_PASSWORD_CHANGE_NTF frame format.

5.1.5.1 NewPassword

See paragraph 5.1.1.1 for description.

6 General device commands

6.1 Version information commands

Use GW_GET_VERSION_REQ to get information about current KLF200 firmware version. A GW_GET_VERSION_CFM frame will be returned.

Use GW_GET_PROTOCOL_VERSION_REQ to get information of the current protocol ID used by gateway and what version of this specification the firmware matches.

6.1.1 GW_GET_VERSION_REQ

Command
GW_GET_VERSION_REQ

Table 13 - GW_GET_VERSION_REQ frame format.

6.1.2 GW_GET_VERSION_CFM

Command	Data 1 - 6	Data 7	Data 8	Data 9
GW_GET_VERSION_CFM	SoftwareVersion	HardwareVersion	ProductGroup	ProductType

Table 14 - GW_GET_VERSION_CFM frame format.

6.1.2.1 SoftwareVersion parameter

SoftwareVersion is a six byte long byte array, containing the current firmware version of KLF200.

SoftwareVersion parameter	Description
Data 1	Command Version Number
Data 2	Version Whole Number
Data 3	Version Sub Number
Data 4	Branch ID
Data 5	Build Number
Data 6	Micro Build

Table 15 - SoftwareVersion description

6.1.2.2 HardwareVersion parameter

HardwareVersion is a single byte, containing the current hardware version of KLF200.

6.1.2.3 ProductGroup parameter

ProductGroup is a single byte, containing the product group number for the gateway, this can be used to identify the gateway. KLF200 is members of remote control product group, therefore ProductGroup is always 14.

6.1.2.4 ProductType parameter

ProductType is a single byte, containing the product type number for the gateway, this can be used to identify the gateway. ProductType is 3 for KLF200.

6.1.3 GW_GET_PROTOCOL_VERSION_REQ

Command
GW_GET_PROTOCOL_VERSION_REQ

Table 16 - GW_GET_PROTOCOL_VERSION_REQ frame format.

6.1.4 GW_GET_PROTOCOL_VERSION_CFM

Command	Data 1-2	Data 3-4
GW_GET_PROTOCOL_VERSION_CFM	MajorVersion	MinorVersion

Table 17 - GW_GET_PROTOCOL_VERSION_CFM frame format.

6.1.4.1 MajorVersion parameter

MajorVersion is 2 bytes, containing the major version of protocol which the gateway SW is compatible with.

6.1.4.2 MinorVersion parameter

MinorVersion is 2 bytes, containing the minor version of protocol which the gateway SW is compatible with.

6.2 Gateway state

The user can get the state of the gateway, during an ongoing operation, using GW_GET_STATE_REQ/CFM command set. This command set can also be as a kind of ping method.

6.2.1 GW_GET_STATE_REQ

Command
GW_GET_STATE_REQ

Table 18 - GW_GET_STATE_REQ frame format.

6.2.2 GW_GET_STATE_CFM

Command	Data 1	Data 2	Data 3 – 6
GW_GET_STATE_CFM	GatewayState	SubState	StateData

Table 19 - GW_GET_STATE_CFM frame format.

6.2.2.1 GatewayState

GatewayState value	Description
0	Test mode.
1	Gateway mode, no actuator nodes in the system table.
2	Gateway mode, with one or more actuator nodes in the system table.
3	Beacon mode, not configured by a remote controller.
4	Beacon mode, has been configured by a remote controller.
5 - 255	Reserved.

Table 20 - GatewayState value Description

6.2.2.2 SubState

SubState is defined when GatewayState is 1 or 2.

SubState value, when GatewayState is 1 or 2	Description
0x00	Idle state.
0x01	Performing task in Configuration Service handler
0x02	Performing Scene Configuration
0x03	Performing Information Service Configuration.
0x04	Performing Contact input Configuration.
	In Contact input Learn state. ???
0x80	Performing task in Command Handler
0x81	Performing task in Activate Group Handler
0x82	Performing task in Activate Scene Handler
Other values	Reserved.

Table 21 - Value description for SubState, when GatewayState is 1 or 2.

6.2.2.3 StateData

StateData parameter is reserved for future use.

6.3 Leave learn state

If the gateway has been put into learn state by press learn button, then GW_LEAVE_LEARN_STATE_REQ can be sent, for the gateway to leave learn state.

6.3.1 GW_LEAVE_LEARN_STATE_REQ

Command
GW_LEAVE_LEARN_STATE_REQ

Table 22 - GW_LEAVE_LEARN_STATE_REQ frame format.

6.3.2 GW_LEAVE_LEARN_STATE_CFM

Command	Data 1
GW_LEAVE_LEARN_STATE_CFM	Status

Table 23 - GW_LEAVE_LEARN_STATE_CFM frame format.

6.3.2.1 Status

Status value	Description
0	The request failed.
1	The request was successful.

Table 24 - Status parameter

6.4 Real Time Clock

The gateway has a real-time clock running at UTC. The client can set a local time zone and daylight savings rules.
The UTC time must be set every time the gateway is powered on.
UTC time can be set with GW_SET_UTC_REQ.

6.4.1 GW_SET_UTC_REQ

Command	Data 1 – 4
GW_SET_UTC_REQ	utcTimeStamp

Table 25- GW_SET_UTC_REQ frame format.

6.4.1.1 Parameter utcTimeStamp

utcTimeStamp is a 32-bit unsigned integer, representing the number of seconds elapsed since 00:00 hours, Jan 1, 1970 UTC (i.e., a *unix timestamp*).

6.4.2 GW_SET_UTC_CFM

Command
GW_SET_UTC_CFM

Table 26 - GW_SET_UTC_CFM frame format.

6.4.3 GW_RTC_SET_TIME_ZONE_REQ

Command	Data 1 - 64
GW_RTC_SET_TIME_ZONE_REQ	TimeZoneString

Table 27 - GW_RTC_SET_TIME_ZONE_REQ frame format.

6.4.3.1 TimeZoneString parameter

TimeZoneString is a 64-byte long string, formatted as UTF-8.

The string should be on the following form:
:[XXX[:YYY[:NNN[:DST[:DST ...]]]]]

Where XXX is the standard time-zone name, YYY is the daylight savings time-zone name, NNN is the time zone offset, and the DSTs are the daylight savings time rules. Daylight savings time will add one hour to the normal time. (The names are only used in the 'Z' formatter in the strftime library function.)

The time zone offset NNN is specified as a number relative to UTC, possibly negative (east is positive), on the format HHMM, where HH is hours and MM is minutes.

The DSTs specifes a set of rules for how daylight savings time is applied. The rules must be sorted in increasing date order starting from the earliest date. The first rule for a specific year will enable DST, the next will disable it, and so on. Each rule is on the following form:

  [(YYYY)]MMDD[HH][-W|+W]

- (YYYY) is the first year the daylight savings rule was applied. It is optional. If not specified it will default to the same year as the previous rule or zero if no previous rule.
- MM is the month number (1-12).
- DD is the day of the month (1-31).
- HH is the hour number in a 24-hour day (optional, defaults to 0).
- +/-W specifies the day of the week the rule takes effect (where Sunday = 0, Monday = 1, etc). +W means that the rule applies to the first such day on or after the specified date and -W strictly before the date. If this is not specified, the rule will take effect on the exact date, regardless of the day of the week.

On the northern hemisphere, the DST rules normally comes in pairs, a start, Aprilish, and an end, Octoberish. On the southern hemisphere one normally has to use three rules: enabling DST from start of year, disabling it in Aprilish, and then enabling it again in Octoberish.

Examples:
  :GMT:GMT+1:0060:(1990)040102-0:100102-0
Here, the time zone is GMT and under daylight savings time the time zone is named GMT+1. The time zone offset is 0060, i.e. 60 minutes from UTC. As of the year 1990, daylight savings time started on the Sunday before (but not on) 1:st of April at 2am and ends on the first Sunday before (but not on) the first of October.

  :GMT+10:GMT+11:0900:(1990)010100-0:040102-0:100102-0
Tasmania is on UTC+10 hours, with daylight savings time from first Sunday in October until first Sunday in April. Note, the first DST rule is for enabling from start of the year.

6.4.4 GW_RTC_SET_TIME_ZONE_CFM

Command	Data 1
GW_RTC_SET_TIME_ZONE_CFM	Status

Table 28 - GW_RTC_SET_TIME_ZONE_CFM frame format.

6.4.4.1 Status parameter

Status value	Description
0	The request failed.

1	The request was successful.

Table 29 - Status parameter

6.4.5 GW_GET_LOCAL_TIME_REQ command

Command
GW_GET_LOCAL_TIME_REQ

Table 30 - GW_GET_LOCAL_TIME_REQ frame format.

6.4.6 GW_GET_LOCAL_TIME_CFM command

Command	Data 1 - 4	Data 5	Data 6	Data 7
GW_GET_LOCAL_TIME_CFM	UtcTime	Second	Minute	Hour

Data 8	Data 9	Data 10 - 11	Data 12	Data 13 - 14	Data 15
DayOfMonth	Month	Year	WeekDay	DayOfYear	DaylightSavingFlag

Table 31 – GW_GET_LOCAL_TIME_CFM frame format.

6.4.6.1 UtcTime parameter

Current UNIX time stamp.

6.4.6.2 Second parameter

Seconds after the minute (local time), range 0-61

6.4.6.3 Minute parameter

Minutes after the hour (local time), range 0-59

6.4.6.4 Hour parameter

Hours since midnight (local time), range 0-23

6.4.6.5 DayOfMonth parameter

Day of the month, range 1-31

6.4.6.6 Month parameter

Months since January, range 0-11

6.4.6.7 Year parameter

Years since 1900

6.4.6.8 WeekDay parameter

Days since Sunday, range 0-6

6.4.6.9 DayOfYear parameter

Days since January 1, range 0-365

6.4.6.10   DaylightSavingFlag parameter

Value	Description
-1	DST information not available
0	DST is NOT in effect
1	DST is in effect

Table 32 - DaylightSavingFlag parameter description.

6.5 Reboot command set

Use GW_REBOOT_REQ to let KLF200 reboot. KLF200 acknowledge with GW_REBOOT_CFM before reboot. After reboot, Ethernet socket must be reconnected.

6.5.1 GW_REBOOT_REQ

Command
GW_REBOOT_REQ

Table 33 - GW_REBOOT_REQ frame format.

6.5.2 GW_REBOOT_CFM

Command
GW_REBOOT_CFM

Table 34 – GW_REBOOT_CFM frame format.

6.6 Factory default command set

By GW_SET_FACTORY_DEFAULTS_REQ you let KLF200 clear system table, scene table and set Ethernet settings to factory default. Doing that KLF200 will reboot. After 30 seconds a new Ethernet socket can be established.

6.6.1 GW_SET_FACTORY_DEFAULT_REQ

Command
GW_SET_FACTORY_DEFAULT_REQ

Table 35 - GW_SET_FACTORY_DEFAULT_REQ frame format.

6.6.2 GW_SET_FACTORY_DEFAULT_CFM

Command
GW_SET_FACTORY_DEFAULT_CFM

Table 36 - GW_SET_FACTORY_DEFAULT_CFM frame format.

6.7 Network setup

6.8 Get network setup command set

When the gateway receives a GW_GET_NETWORK_SETUP_REQ frame it will return a EV_CMP_GET_NETWORK_SETUP_CFM frame with the actual network settings.

6.8.1 GW_GET_NETWORK_SETUP_REQ

Command
GW_GET_NETWORK_SETUP_REQ

Table 37 - GW_GET_NETWORK_SETUP_REQ frame format.

6.8.2 GW_GET_NETWORK_SETUP_CFM

Command	Data 1 - 4	Data 5 - 8	Data 9 - 12	Data 13
GW_GET_NETWORK_SETUP_CFM	IpAddress	Mask	DefGW	DHCP

Table 38 - GW_GET_NETWORK_SETUP_CFM frame format.

6.8.2.1 IpAddress parameter

IpAddress is an array of four bytes. IpAddress hold an IPv4 IP address.

6.8.2.2 Mask parameter

Mask is an array of four bytes. Mask is used to setup the network mask.

6.8.2.3 DefGW parameter

DefGW is a abbreviation for default gateway. DefGW is an array of four bytes. If you don't know what to use it for, just put zeroes in.

6.8.2.4 DHCP parameter

The gateway can ether use IpAddress, Mask and DefGW to setup Ethernet interface or get the network settings from a DHCP server on the network.
DHCP is set to 1 as default.

DHCP value	Description
0	Disable DHCP. Use IpAddress, Mask and DefGW to setup Ethernet interface.
1	Enable DHCP. IpAddress, Mask and DefGW are not used to setup Ethernet interface.

Table 39 - DHCP parameter description.

6.9 Set network setup command set

When gateway receives GW_SET_NETWORK_SETUP_REQ it will store new network settings in EEPROM, send GW_SET_NETWORK_SETUP_CFM and reboot.

6.9.1 GW_SET_NETWORK_SETUP_REQ

Command	Data 1 - 4	Data 5 - 8	Data 9 - 12	Data 13
GW_SET_NETWORK_SETUP_REQ	IpAddress	Mask	DefGW	DHCP

Table 40 - GW_SET_NETWORK_SETUP_REQ frame format.

The parameters are described in paragraph 6.8.2.1, 6.8.2.2, 6.8.2.3 and 6.8.2.4.

6.9.2 GW_SET_NETWORK_SETUP_CFM

Command
GW_SET_NETWORK_SETUP_CFM

Table 41 - GW_SET_NETWORK_SETUP_CFM frame format.

6.10 GW_ERROR_NTF

If an error arises, a GW_ERROR_NTF command is send from the gateway to the user. The PC shall always be ready for an error command, no matter which command the PC else expect. The GW_ERROR_NTF commands frame format is shown in Table 42.

Command	Data 1
GW_ERROR_NTF	ErrorNumber

Table 42 - GW_ERROR_NTF command frame format.

The parameter ErrorNumber tells the error type. See list of error types in Table 43.

ErrorNumber value	Description
0	Not further defined error.
1	Unknown Command or command is not accepted at this state.
2	ERROR on Frame Structure.
7	Busy. Try again later.
8	Bad system table index.
12	Not authenticated.

Table 43 - Error types.

7 Configuration service

Configuration service is a number of events used to setup and maintain a system of io-homecontrol® nodes. Information about these nodes is stored in the gateway in a system table.

When configuration services are in use, KLF200 can't be interrupted, by scene or command handler commands. Instead a GW_ERROR_NTF(BUSY) command will be returned, and KLF200 continue configuration service execution.

7.1 System table

The system table in the gateway can store up to 200 actuators and up to three Beacons (RF repeaters).

The system table is divided in three sections.

Index 0 to 199 can contain actuators, index 201 to 203 can contain beacons and index 200 are not used in this version.

7.2 GW_CS_GET_SYSTEMTABLE_DATA_REQ

The command GW_CS_GET_SYSTEMTABLE_DATA_REQ is used to get a copy of the systemtable. Its frame format is shown in Table 44.

Command
GW_CS_GET_SYSTEMTABLE_DATA_REQ

Table 44 - GW_CS_GET_SYSTEMTABLE_DATA_REQ frame format.

As acknowledge to GW_CS_GET_SYSTEMTABLE_DATA_REQ, the gateway send GW_CS_GET_SYSTEMTABLE_DATA_CFM and one or more GW_CS_GET_SYSTEMTABLE_DATA_NTF frames. See its frame format Table 46.

7.3 GW_CS_GET_SYSTEMTABLE_DATA_CFM

Command
GW_CS_GET_SYSTEMTABLE_DATA_CFM

Table 45 - GW_CS_GET_SYSTEMTABLE_DATA_CFM frame format.

7.4 GW_CS_GET_SYSTEMTABLE_DATA_NTF

Command	Data 1	Data 2 – (n+1)	Data (n+2)
GW_CS_GET_SYSTEMTABLE_DATA_NTF	NumberOfEntry	SystemTableObjects	RemainingNumberOfEntry

Table 46 - GW_CS_GET_SYSTEMTABLE_DATA_NTF frame format. Note n ∈ {11; 22; … ; 110}.

7.4.1.1 NumberOfEntry parameter

Tells how many system table objects there are transferred in this particular frame. NumberOfEntry is a number from 0 to 10.

7.4.1.2 SystemTableObjects parameter

The SystemTableObjects parameter is a list of System table objects.

See SystemTableObjects parameter framing in Table 47 and the data format for one system table node object in Table 48.

Data 2 – 12	Data 13 – 23	Data 24 – 34	…	Data (n-11) - n
Object one in System table	Object two in System table	Object three in System table	…	Last object in this frame

Table 47 - Frame format of the parameter SystemTableObjects.

Class: General Actuator	
Byte Index	**Description**

1	System table index.
2	Actuator address Highest Byte
3	Actuator address Middle Byte
4	Actuator address Lowest Byte
5-6	Actuator Type (MSBits), Actuator Type (LSBits) – Actuator Sub Type
7	Bit 0-1 : PowerSave Mode Bit 2 : io-Membership Bit 3 : RF support Bit 6-7 : Actuator Turnaround time.
8	io-Manufacturer Id
9	Backbone reference number Highest byte
10	Backbone reference number Middle byte
11	Backbone reference number Lowest byte

Table 48 - Format of a SystemTable object.

Normally the only field of interest for the product which interface to the gateway is the System table index field and Actuator type and sub type field.

7.4.1.2.1 System table index parameter

System table index can be a number from 0 to 203.

7.4.1.2.2 Actuator address parameter

In one system, every io-homecontrol® node has a unique three-byte long address.

7.4.1.2.3 Actuator Type and Sub Type parameter

Actuator Type is 10 bit (AT0 to AT9) and Actuator Sub Type is 6 bit (ST0 to ST5).
The format of the Actuator Type and Actuator Sub type is as shown below:

Byte 4								Byte 5							
AT9	AT8	AT7	AT6	AT5	AT4	AT3	AT2	AT1	AT0	ST5	ST4	ST3	ST2	ST1	ST0

Table 49 - Actuator Type and Sub Type

AT9 is the MSBit of the Actuator type and ST5 is MSBit of the Actuator Sub type.

Actuator type value	Description
1	Venetian blind
2	Roller shutter
3	Awning (External for windows)
4	Window opener
5	Garage opener
6	Light
7	Gate opener
8	Rolling Door Opener
9	Lock
10	Blind
12	Beacon
13	Dual Shutter
14	Heating Temperature Interface
15	On / Off Switch
16	Horizontal Awning
17	External Venetian Blind
18	Louvre Blind
19	Curtain track
20	Ventilation Point
21	Exterior heating
22	Heat pump (Not currently supported)
23	Intrusion alarm
24	Swinging Shutter
Other values	Reserved.

Table 50 - NodeType data parameter description.

See Appendix 2: List of actuator types and their use of Main Parameter and Functional Parameters page 104 for a combined Actuator Type and Sub Type value for known actuators.

7.4.1.2.4 PowerSave Mode parameter

PowerSave Mode	Description
0	ALWAYS_ALIVE
1	LOW_POWER_MODE
Other values	Reserved.

Table 51 - PowerSave Mode parameter description.

7.4.1.2.5 io-Membership parameter

io-Membership is always 1.

7.4.1.2.6 RF support parameter

Nodes on backbone has not its own RF support.

RF support	Description
0	No RF support
1	RF support

Table 52 - RF support parameter description.

7.4.1.2.7 Actuator Turnaround time parameter

Actuator Turnaround time is the time each node must respond within.
Response seen by the operator of the gateway is obviously larger.

Actuator Turnaround time parameter value	Actuator Turnaround time
0	5 ms
1	10 ms
2	20 ms
3	40 ms

Table 53 - Actuator Turnaround time parameter description.

7.4.1.2.8 io-Manufacturer Id parameter

io-Manufacturer Id value	Description
1	VELUX
2	Somfy
3	Honeywell
4	Hörmann
5	ASSA ABLOY
6	Niko
7	WINDOW MASTER
8	Renson
9	CIAT
10	Secuyou
11	OVERKIZ
12	Atlantic Group
Other values	Not defined at the moment.

Table 54 - io-Manufacturer Id parameter description.

7.4.1.2.9 Backbone reference number

The backbone reference number is a unique number for all actuators connected to the same backbone. An example; In a VELUX integra window 4 different products is mounted (a window opener, a shutter, a blind and light) each of these actuators are connected to Powerlink, which function as a backbone net. Each of the 4 products has a unique address, but the same backbone address which is equal to one of the 4 addresses.

7.4.1.3 RemainingNumberOfEntry parameter

This parameter tells the remaining number of system table object to be transferred. This means if RemainingNumberOfEntry ≠ 0, the gateway will send at least one more GW_CS_GET_SYSTEMTABLE_DATA_CFM frame.

7.4.2 GW_CS_GET_SYSTEMTABLE_DATA_NTF frame if system table are empty.

The GW_CS_GET_SYSTEMTABLE_DATA_NTF frame are a little different if there no nodes in the system table. See example in Table 55.

Command	Data 1	Data 2
GW_CS_GET_SYSTEMTABLE_DATA_NTF	NumberOfEntry = 0	RemainingNumberOfEntry = 0

Table 55 - GW_CS_GET_SYSTEMTABLE_DATA_NTF frame format. Example where there are no nodes in the system table.

7.5 Discover nodes

The command GW_CS_DISCOVER_NODES_REQ is used to add new nodes to the system table. The GW_CS_DISCOVER_NODES_REQ function also validates nodes already in system table.
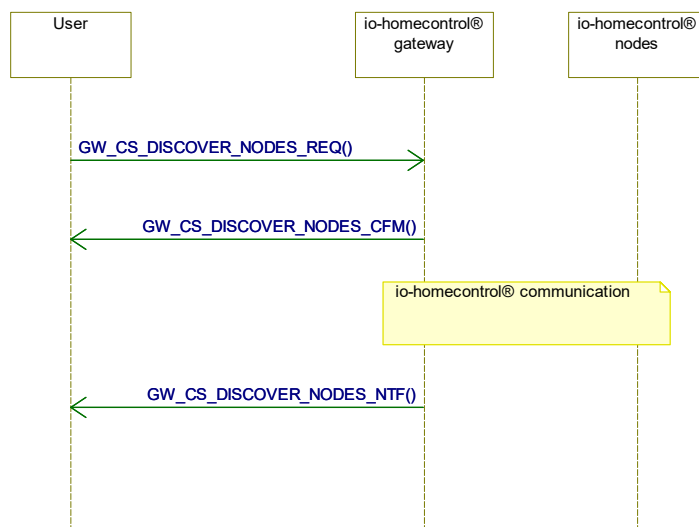
Figure 8 - Discover nodes sequence diagram.

7.5.1 GW_CS_DISCOVER_NODES_REQ

Command	Data 1
GW_CS_DISCOVER_NODES_REQ	NodeType

Table 56 - GW_CS_DISCOVER_NODES_REQ frame format.

7.5.1.1 NodeType parameter

The NodeType is used to limit the type of nodes to be obtained in the system table. For example, if NodeType = 6 only actuators with ActuatorType = 6 (Light) will added to the system table. The other node types will remain open for configuration. The gateway user can send GW_CS_DISCOVER_NODES_REQ command several times with different NodeType value. Existing nodes in the system table will not be deleted.

See known node types at the moment in Table 57.

NodeType value	NodeType description
0	NO_TYPE (All nodes except controller)
1	Venetian blind
2	Roller shutter
3	Awning (External for windows)
4	Window opener
5	Garage opener
6	Light
7	Gate opener
8	Rolling Door Opener
9	Lock
10	Blind
12	Beacon
13	Dual Shutter
14	Heating Temperature Interface
15	On / Off Switch
16	Horizontal Awning
17	External Venetian Blind
18	Louvre Blind
19	Curtain track
20	Ventilation Point
21	Exterior heating
22	Heat pump (Not currently supported)
23	Intrusion alarm
24	Swinging Shutter

Table 57 - NodeType data parameter description.

For more information about io-homecontrol® node types and sub types, see "Appendix 2: List of actuator types and their use of Main Parameter and Functional Parameters" at page 104.

7.5.2 GW_CS_DISCOVER_NODES_CFM

When the gateway receives GW_CS_DISCOVER_NODES_REQ it will return GW_CS_DISCOVER_NODES_CFM.

Command
GW_CS_DISCOVER_NODES_CFM

Table 58 - GW_CS_DISCOVER_NODES_CFM frame format.

7.5.3 GW_CS_DISCOVER_NODES_NTF

GW_CS_DISCOVER_NODES_NTF carry information about the discover nodes proces.

Command	Data 1 – 26	Data 27 – 52	Data 53 - 78
GW_CS_DISCOVER_NODES_NTF	AddedNodes	RFConnectionError	ioKeyErrorExistingNode

Data 79-104	Data 105-130	Data 131
Removed	Open	DiscoverStatus

Table 59 - GW_CS_DISCOVER_NODES_NTF frame format.

The GW_CS_DISCOVER_NODES_NTF command carries following data:

7.5.3.1 AddedNodes

Bit-array where Bit 0 – Bit 7 are in data byte 1 and Bit 200 – Bit 207 in data byte 26.

Bit 0 – Bit 199:   1 = Actuator added, 0 = Actuator not added.

Bit 201 – Bit 203:   1 = Beacon added, 0 = Beacon not added.
Bit 204 – Bit 207 are not used.

7.5.3.2 RFConnectionError

This bit-array informs about which nodes already in the system table, that there is no contact to now. These nodes are not automatically removed by the discover nodes macro.

Bit 0 – Bit 199:   1 = No RF contact to Actuator, 0 = OK.
Bit 201 – Bit 203:   1 = No RF contact to Beacon, 0 = OK.
Bit 204 – Bit 207:   Are not used.

7.5.3.3 ioKeyErrorExistingNode

If a node has a wrong system key it can't complete the security test.
Bit-array indicates which nodes there should be removed from system table.
Bit 0 – Bit 199:   1 = Actuator should be removed, 0 = Actuator should not be removed.
Bit 201 – Bit 203:   1 = Beacon should be removed, 0 = Beacon should not be removed.
Bit 204 – Bit 207:   Are not used.

7.5.3.4 Removed

Bit-array indicates which nodes there removed from the system table. It could happen if discovered is not a io-homecontrol® member.

7.5.3.5 Open

Bit-array indicates which nodes there are in configuration mode.Iit can arise if a given nodes already in the system table, is now open for configuration.

7.5.3.6 DiscoverStatus

DiscoverStatus value	Description
0	OK. Discovered nodes. See bit array.
5	Failed. CS not ready.
6	OK. Same as DISCOVER_NODES_PERFORMED but some nodes were not added to system table (e.g. System table has reached its limit).
7	CS busy with another task.

Table 60 - Parameter DiscoverStatus description.

7.6 Remove Nodes command set

One or more nodes can be removed from the system table using GW_CS_REMOVE_NODES_REQ command.

7.6.1 GW_CS_REMOVE_NODES_REQ

Command	Data 1 – 26
GW_CS_REMOVE_NODES_REQ	RemoveNodes

Table 61 - GW_CS_REMOVE_NODES_REQ frame format.

7.6.1.1 RemoveNodes

Information about which nodes to remove from the system table is carried in this bit-array.

Bit 0 – Bit 7 in data byte 1 and Bit 200 – Bit 207 in data byte 26.

Bit 0 – Bit 199: 1 = Remove Actuator, 0 = Do not remove Actuator.

Bit 201 – Bit 203: 1 = Remove Beacon, 0 = Do not remove Beacon.

Bit 200 and Bit 204 – Bit 207 are not used.

7.6.2 GW_CS_REMOVE_NODES_CFM

Command	Data 1
GW_CS_REMOVE_NODES_CFM	SceneDeleted

Table 62 - GW_CS_REMOVE_NODES_CFM frame format.

GW_CS_REMOVE_NODES_CFM command is returned when selected nodes are removed from system table.

7.6.2.1 SceneDeleted

Scenes using one or more of the deleted nodes are deleted.

If one or more scenes are deleted, the parameter SceneDeleted is set. See table below.

SceneDeleted value	Description
0	No scene is deleted.
1	One or more scenes are deleted.

Table 63 - Parameter SceneDeleted description.

7.7 Virgin State command set

The command GW_CS_VIRGIN_STATE_REQ performs the following four actions:

- Clear all nodes in the system table.
- Pick out a new io-Address.
- Pick out a new io-System Key.
- Clear all scenes in scene list.

7.7.1 GW_CS_VIRGIN_STATE_REQ

Command
GW_CS_VIRGIN_STATE_REQ

Table 64 - GW_CS_VIRGIN_STATE_REQ frame format.

7.7.2 GW_CS_VIRGIN_STATE_CFM

When the controller is cleared, it returns GW_CS_VIRGIN_STATE_CFM command.

Command
GW_CS_VIRGIN_STATE_CFM

Table 65 - GW_CS_VIRGIN_STATE_CFM frame format.

7.8 Controller Copy command set

The GW_CS_CONTROLLER_COPY commands makes it possible to either copy system table and io-system key from one controller to KLF200 gateway or the other way around.

The two modes for the gateway to make controller copy:

- Transmitting Configuration mode (TCM): The gateway is master and copy key and system table from a controller in Receiving Configuration mode. If the gateway finds a controller in Receiving Configuration mode, the gateway will lose existing system table nodes and system key and get nodes and new system key from the found controller.

- Receiving Configuration mode (RCM): This puts the gateway into a slave mode and the gateway will give its key and system table to the first controller which asks for it. RCM will timeout after 2 minutes if system has not been transferred to another controller or if RCM job is cancelled.
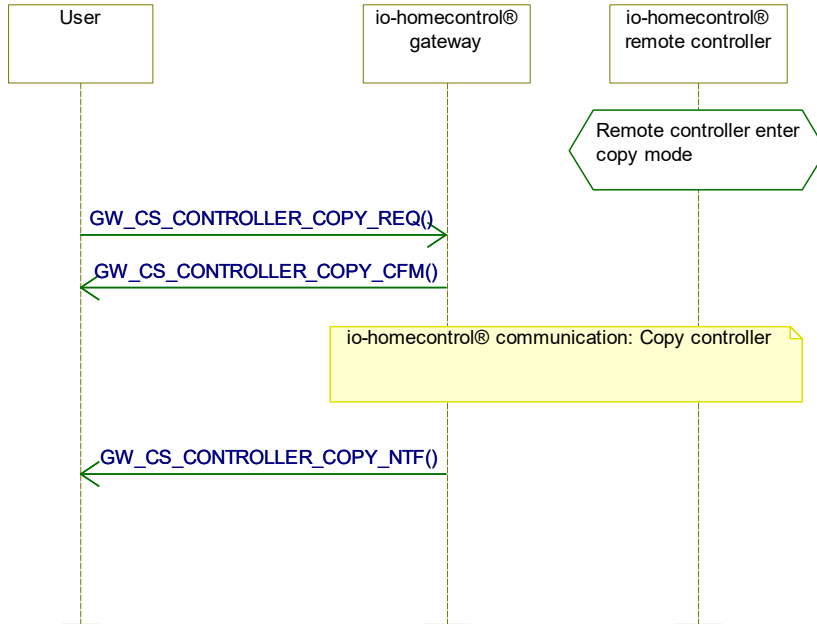
Figure 9 - Sequence diagram -Normal controller copy from remote controller to gateway (ControllerCopyMode = 0).
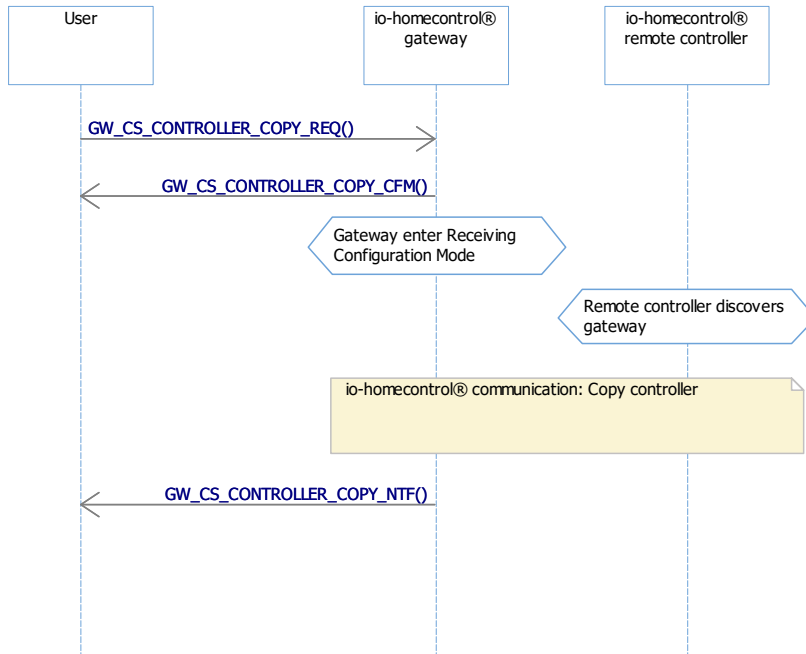
Figure 10 -   Sequence diagram -Normal controller copy from gateway to remote controller (ControllerCopyMode = 1).

7.8.1 GW_CS_CONTROLLER_COPY_REQ

Command	Data 1
GW_CS_CONTROLLER_COPY_REQ	ControllerCopyMode

Table 66 - GW_CS_CONTROLLER_COPY_REQ frame format

ControllerCopyMode value	Description
0	Transmitting Configuration Mode (TCM): The gateway gets key and system table from another controller.
1	Receiving Configuration Mode (RCM): The gateway gives key and system table to another controller.

Table 67 - ControllerCopyMode parameter description.

7.8.2 GW_CS_CONTROLLER_COPY_CFM

Command
GW_CS_CONTROLLER_COPY_CFM

Table 68 - GW_CS_CONTROLLER_COPY_CFM frame format.

7.8.3 GW_CS_CONTROLLER_COPY_NTF

Command	Data 1	Data 2
GW_CS_CONTROLLER_COPY_NTF	ControllerCopyMode	ControllerCopyStatus

Table 69 - GW_CS_CONTROLLER_COPY_NTF frame format.

ControllerCopyMode is always the same value as the one set in
GW_CS_CONTROLLER_COPY_REQ command.
ControllerCopyStatus content varies depending on the ControllerCopyMode chosen in
request.

ControllerCopyStatus value (TCM)	Description
0	OK. System table and key received from another io-node.
2	Failed. Not possible to find another controller in receiving configuration mode.
4	Failed. DTS not ready. (DTS stands for Data Transport Service)
5	Failed. DTS error. Client must activate Virgin State. Reason: The Client Controller contains a defect system.
9	Failed. Configuration service not ready.

Table 70 - Parameter ControllerCopyStatus description with Transmitting Configuration Mode

ControllerCopyStatus value (RCM)	Description
0	OK. Data transfer to or from client controller.
1	Failed. Data transfer to or from client controller interrupted.
4	Ok. Receiving configuration mode is cancelled in the client controller.
5	Failed. Timeout.
11	Failed. Configuration service not ready.

Table 71 - Parameter ControllerCopyStatus description with Receiving Configuration Mode

7.8.4 GW_CS_CONTROLLER_COPY_CANCEL_NTF

Command
GW_CS_CONTROLLER_COPY_CANCEL_NTF

Table 72 - GW_CS_CONTROLLER_COPY_CANCEL_NTF frame format.

GW_CS_CONTROLLER_COPY_CANCEL_NTF can be used to cancel the transmission of a
system to another controller (meaning it can only be used with ControllerCopyMode = 1).
It clears the timeout timer for Receiving Configuration Mode.

Note: No acknowledge event (_CFM) is returned when using this event. The
GW_CS_CONTROLLER_COPY_NTF status byte will indicate that Copy job was cancelled.
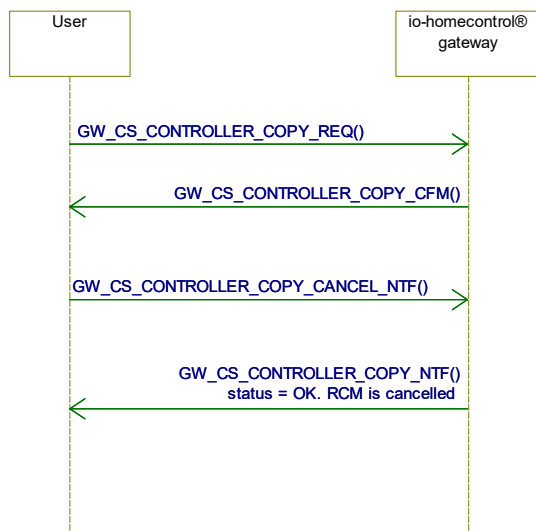
Figure 11 - Sequence diagram - Cancel controller copy.

Figure 11 shows sequence diagram for normal controller copy (Receiving Configuration Mode) which is cancelled.

There are three ways the gateway can leave Receiving Configuration Mode.
1. The gateway has successfully delivered the system to another controller.
2. After 2 minutes.
3. The command GW_CS_CONTROLLER_COPY_CANCEL_NTF is sent to the gateway.

7.9 Generate new system Key

7.9.1 GW_CS_GENERATE_NEW_KEY_REQ

Command
GW_CS_GENERATE_NEW_KEY_REQ

Table 73 - GW_CS_GENERATE_NEW_KEY_REQ frame format.

7.9.2 GW_CS_GENERATE_NEW_KEY_CFM

Command
GW_CS_GENERATE_NEW_KEY_CFM

Table 74 - GW_CS_GENERATE_NEW_KEY_CFM frame format.

7.9.3 GW_CS_GENERATE_NEW_KEY_NTF

Command	Data 1	Data 2 - 27	Data 28 - 53
GW_CS_GENERATE_NEW_KEY_NTF	ChangeKeyStatus	KeyChanged	KeyNotChanged

Table 75 - GW_CS_GENERATE_NEW_KEY_NTF frame format.

7.9.3.1 ChangeKeyStatus parameter

ChangeKeyStatus value	Description
0	Ok. Key Change in client controller.
2	Ok. Key change in system table all nodes updated with current key.
3	Ok. Key Change in System table. Not all nodes in system table was updated with current key. Check bit array.
5	Ok. Client controller received a key.
7	Failed. Local Stimuli not disabled in all Client System table nodes. See bit array.

9	Failed. Not able to find a controller to get key from.
10	Failed. DTS not ready.
11	Failed. DTS error. At DTS error no key change will take place. Backup of beacon at the beginning of key change, is restored in the client controller.
16	Failed. CS not ready.

Table 76 - Parameter ChangeKeyStatus description.

7.9.3.2 KeyChanged parameter

26-byte long bit array.

7.9.3.3 KeyNotChanged parameter

26-byte long bit array.

7.10 Receive Key command set

The GW_CS_RECEIVE_KEY commands give the opportunity to receive a system key from another system.

7.10.1 GW_CS_RECEIVE_KEY_REQ

Command
GW_CS_RECEIVE_KEY_REQ

Table 77 - GW_CS_RECEIVE_KEY_REQ frame format.

7.10.2 GW_CS_RECEIVE_KEY_CFM

Command
GW_CS_RECEIVE_KEY_CFM

Table 78 - GW_CS_RECEIVE_KEY_CFM frame format.

7.10.3 GW_CS_RECEIVE_KEY_NTF

Command	Data 1	Data 2 - 27	Data 28 - 53
GW_CS_RECEIVE_KEY_NTF	ChangeKeyStatus	KeyChanged	KeyNotChanged

Table 79 - GW_CS_RECEIVE_KEY_NTF frame format.

7.10.3.1 ChangeKeyStatus parameter

ChangeKeyStatus value	Description
0	Ok. Key Change in client controller.
2	Ok. Key change in system table all nodes updated with current key.
3	Ok. Key Change in System table. Not all nodes in system table was updated with current key. Check bit array.
5	Ok. Client controller received a key.
7	Failed. Local Stimuli not disabled in all Client System table nodes. See bit array.
9	Failed. Not able to find a controller to get key from.
10	Failed. DTS not ready.
11	Failed. DTS error. At DTS error no key change will take place. Backup of beacon at the beginning of key change, is restored in the client controller.
16	Failed. CS not ready.

Table 80 - Parameter ChangeKeyStatus description.

7.10.3.2 KeyChanged parameter

26-byte long bit array.

7.10.3.3 KeyNotChanged parameter

26-byte long bit array.

7.11 Update new key in actuators with old key

If one or more bit in KeyNotChanged bit array in GW_CS_GENERATE_NEW_KEY_NTF or GW_CS_RECEIVE_KEY_NTF frame is set, then one or more actuators has an old key. The resent for that can be that an actuator is powered off or out of range.
The gateway stores the latest old key. With GW_CS_REPAIR_KEY_REQ command the gateway will try to update the key in the remaining actuators.

7.11.1 GW_CS_REPAIR_KEY_REQ

Command
GW_CS_REPAIR_KEY_REQ

Table 81 - GW_CS_REPAIR_KEY_REQ frame format.

7.11.2 GW_CS_REPAIR_KEY_CFM

Command
GW_CS_REPAIR_KEY_CFM

Table 82 - GW_CS_REPAIR_KEY_CFM frame format.

7.11.3 GW_CS_REPAIR_KEY_NTF

Command	Data 1	Data 2 - 27	Data 28 - 53
GW_CS_REPAIR_KEY_NTF	ChangeKeyStatus	KeyChanged	KeyNotChanged

Table 83 - GW_CS_REPAIR_KEY_NTF frame format.

7.12 Product Generic Configuration (PGC)

The gateway has a button for generic configuration (look for key symbol). You need to use a pointed object, such as a straightened paper clip, to enable the button.

Figure 12 - Straightened paper clip used to enable the PGC button.

Three different button presses with various lengths can make gateway initiate different Configuration Services. LED flashing will indicate the PGC job to be started on button release. LED will also show status for performed PGC job.
If gateway is busy doing other tasks while button is pressed, the button press is ignored. This will not give any feedback on LED.

7.12.1 Button presses overview

Button press	Objective
Short press (0,2 – 2 sec.)	**Get the 2W io-SystemKey** from another controller and update all the nodes in its system table with the new key.

Long press (2 – 7 sec.)	**Give the 2W io-SystemKey** to another controller (either configured or empty) ie. when adding a 2W controller to an existing 2W system (Receiving Configuration mode)
Very long press (7 - 15 sec.)	**Generate a new 2W io-SystemKey** in the controller and update all the nodes in its system table with the new key.
Too long press (> 15 sec.)	Button press ignored. PGC LED stop flashing and turns off.

Table 84 - Button presses overview

7.12.2 PGC job descriptions

7.12.2.1 Get the 2W io-SystemKey

- GET KEY mode is initiated by a <u>short press</u> on the PGC button;
  [0.2s < short press < 2s].
- GET KEY can also be performed on a virgin gateway.
- GET KEY mode can't be cancelled.
  Any additional press on the PGC button when this mode is entered has no effect.
- GET KEY mode can't be initiated during an ongoing control of an actuator – the PGC button is inactive.
- GET KEY mode is automatically cancelled if no controller or another gateway is discovered after several attempts (times out).

7.12.2.2 Give the 2W io-SystemKey
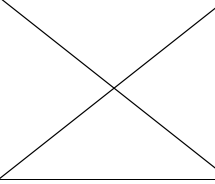
- GIVE KEY mode is initiated by a <u>long press</u> on the PGC button;
  [2s ≤ long press < 7s].
- GIVE KEY can also be performed on a virgin gateway.
- GIVE KEY mode can be cancelled by <u>any additional press</u> on the PGC button;
  [additional press > 0.2s]
  Once a transfer has been initiated for one object, then the transfer can't be cancelled.
  GIVE KEY mode can be cancelled by <u>a "Config Close" fro</u>m another controller.
  Once a transfer has been initiated for one object, then the transfer can't be cancelled.
- GIVE KEY mode can't be initiated during an ongoing control of an actuator – the PGC button is inactive.
- GIVE KEY mode on a virgin gateway (empty system table) causes only the default 2W io-SystemKey to be transmitted (with success feedback).

7.12.2.3 Generate a new 2W io-SystemKey

- GENERATE KEY mode is initiated by a <u>very long press</u> on the PGC button;
  [7s ≤ very long press < 15s]
  Note: The io-SystemKey change is performed when the Key button is <u>released</u>.
- GENERATE KEY mode can only be cancelled by <u>not releasing</u> the PGC button until <u>the LED turns off</u>; [press > 15s]
  Once the generate key mode has been initiated, the key change can't be cancelled.
- GENERATE KEY mode can't be initiated during an ongoing control of an actuator – the PGC button is inactive.
- GENERATE KEY mode on a virgin gateway (empty system table) causes only the default 2W io-SystemKey to be changed (with success feedback).

7.12.3 LED feedback overview

Function	Feedback Function	Feedback SUCCES	Feedback ERROR	Feedback PARTLY SUCCESS
Get the 2W io-SystemKey	1,25 Hz blinking Green (Flashes of 400 ms)			500 msec off + 3 sets of 3 flashes blinking at 10Hz (Flashes of 50ms) with breaks of 500ms
Give the 2W io-SystemKey	3 Hz blinking Green (Flashes of ~170 ms)	500 msec off + 2s continuous light Green	500 msec off + 3 sets of 3 flashes blinking red at 10Hz (Flashes of 50ms) with breaks of 500ms	
Generate a new 2W io-SystemKey				500 msec off + 3 sets of 3 flashes blinking at 10Hz (Flashes of 50ms) with breaks of 500ms

Table 85 - LED feedback overview

Partly success for 'Get the 2W io-SystemKey' is when a key has been received but not all system table nodes can be updated with the new key.

Partly success 'Generate a new 2W io-SystemKey' is when a key has been generated but not all system table nodes can be updated with the new key.

7.12.4 GW_CS_PGC_JOB_NTF

Both when PGC job has started and ended, the gateway will send a notify indicating the PGC job state, the status and the job type performed.

Command	Data 1	Data 2	Data 3
GW_CS_PGC_JOB_NTF	PgcJobState	PgcJobStatus	PgcJobType

Table 86 - GW_CS_PGC_JOB_NTF frame format.

7.12.4.1   PgcJobState

PgcJobState indicates if a PGC job has started, completed or been rejected.

PgcJobState	Description
0	PGC job started
1	PGC job ended. Either OK or with error.
2	CS busy with other services

Table 87 - Parameter PgcJobState description

7.12.4.2   PgcJobStatus

PgcJobStatus indicates if a PGC job has Completed OK, partly OK or with errors.

PgcJobStatus	Description
0	OK - PGC and CS job completed
1	Partly success.
2	Failed - Error in PGC/CS job.
3	Failed - Too long key press or cancel of CS service.

Table 88 - Parameter PgcJobStatus description

7.12.4.3 PgcJobType

PgcJobType indicates what type of CS service PGC has performed. If new system has been received, it would make sense to request gateway for new nodetable.

PgcJobType	Description	Initiated by
0	Receive system copy or only get key.	Short PGC button press.
1	Receive key and distribute.	
2	Transmit key (and system).	Long PGC button press.
3	Generate new key and distribute or only generate new key.	Very long PGC button press.
4-255	Don't care.	Can initiated by Too long key press.

Table 89 - Parameter PgcJobType description

7.13 System table change notification

When the system table has been changed a GW_CS_SYSTEM_TABLE_UPDATE_NTF command is sent to all attached clients.
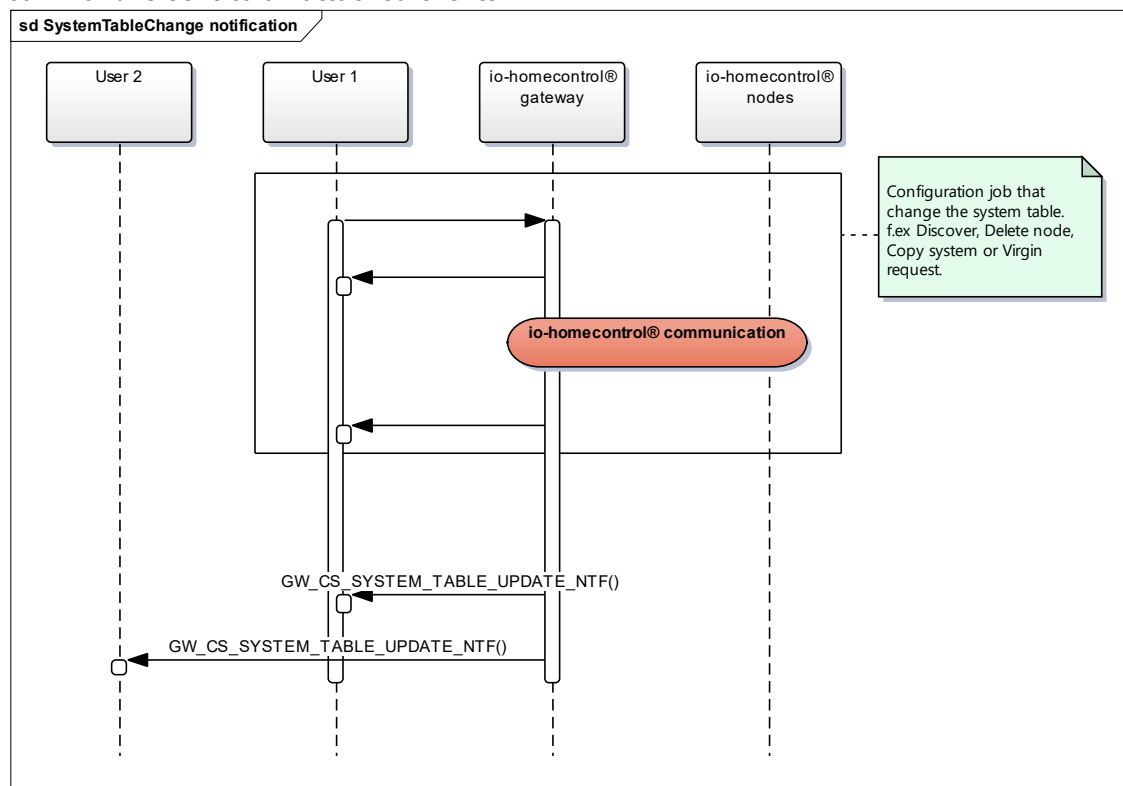
Figure 13 - GW_CS_SYSTEM_TABLE_UPDATE_NTF is sent to all clients.

7.13.1 GW_CS_SYSTEM_TABLE_UPDATE_NTF

Command	Data 1-26	Data 27-52
GW_CS_SYSTEM_TABLE_UPDATE_NTF	AddedNodesBitArray	RemovedNodesBitArray

Figure 14 - GW_CS_SYSTEM_TABLE_UPDATE_NTF frame format.

7.13.1.1 AddedNodesBitArray parameter

AddedNodesBitArray is a 26 byte (208 bit) long array indicating new nodes in system table. Least significant bit in first byte holds information of the actuator node with index 0 and most significant bit of the last byte is for actuator with index 199 and up to tree

beacons from index 201 to 203. Index 200, 204, 205, 206 and 207 is reserved and will not be set.

7.13.1.2 RemovedNodesBitArray parameter

RemovedNodesBitArray is a 26 byte (208 bit) long array indicating removed nodes in system table. Least significant bit in first byte holds information of the actuator node with index 0 and most significant bit of the last byte is for actuator with index 199 and up to tree beacons from index 201 to 203. Index 200, 204, 205, 206 and 207 is reserved and will not be set.

7.14 Open actuator for configuration

One or more actuator can be opened for configuration. This can be used to let a One-Way remote controller operate a given actuator.

7.14.1 GW_CS_ACTIVATE_CONFIGURATION_MODE_REQ

Command	Data 1 – 26
GW_CS_ACTIVATE_CONFIGURATION_MODE_REQ	ActivateConfiguration

Table 90 - GW_CS_ACTIVATE_CONFIGURATION_MODE_REQ frame format.

7.14.1.1 ActivateConfiguration parameter

Information about which nodes to open for configuration is carried in this bit-array.
Bit 0 – Bit 7 in data byte 1 and Bit 200 – Bit 207 in data byte 26.
If bit is true, then the node must be opened for configuration.

7.14.2 GW_CS_ACTIVATE_CONFIGURATION_MODE_CFM

Command	Data 1 – 26	Data 27 – 52	Data 53 – 78	Data 79
GW_CS_ACTIVATE_CONFIGURATION_MODE_CFM	Activated	NoContact	OtherError	Status

Table 91 - GW_CS_ACTIVATE_CONFIGURATION_MODE_CFM frame format.

7.14.2.1 Activated parameter

26 byte long bit array.
1 = Node is in configuration mode. 0 = Node is not in configuration mode.

7.14.2.2 NoContact parameter

26 byte long bit array.
1 = No RF contact with Node. 0 = No RF error.

7.14.2.3 OtherError parameter

26 byte long bit array.
1 = Node error. 0 = No error. Wrong parameter in the acknowledgement frame or wrong response command from actuator or beacon.

7.14.2.4 Status parameter

Status	Description
0	OK
1-255	Some error occurred.

Table 92 – Status parameter description.

8 Information Service

Information service is a set of commands used to exchange information of nodes and groups.

8.1 House Status Monitor service

The gateway can monitor all nodes in the system, if House Status Monitor service (HSM) is enabled. If a parameter, then is changed a notification is sent to all attach users, using GW_NODE_INFORMATION_CHANGED_NTF commands.
The change can come from a user that sets a new parameter using information service commands. Or if a user sets a new position at an actuator, using a remote control for sending an io-homecontrol® command to open window or rolls down blind.
VELUX window openers all have a build in rain sensor. If it starts raining, while the window is opened, the window will then be set to rain protected ventilation position. The rain sensor communicates directly to window opener, and House Status Monitor service therefore don't know about closure of window due to rain.
If HSM is enabled, KLF200 will request information from known actuators, two minutes after boot and then every 15 minutes to 24 hour, depending of actuator type and state, as decried in table below.

Actuator type and state	How often HSM request information
Mains powered actuator or low power actuator with high battery level.	Every 15 minute.
Low power actuator with medium battery level.	Every 1 hour.
Low power actuator with critical low battery level.	Every 24 hour.

Table 93 – How often information is requested from actuator, depending of its type and state.

8.2 Enable or disable House Status Monitor.

From factory, the House Status Monitor service is disabled. The service will be enabled if GW_HOUSE_STATUS_MONITOR_ENABLE_REQ is send to gateway.
This setting is persistent and the service will still be running after a reboot. The service can be stopped if GW_HOUSE_STATUS_MONITOR_DISABLE_REQ command is sent to gateway, or request factory default settings by sending GW_SET_FACTORY_DEFAULT_REQ.

8.2.1 GW_HOUSE_STATUS_MONITOR_ENABLE_REQ

House Status Monitor service will be enabled if GW_HOUSE_STATUS_MONITOR_ENABLE_REQ is send to gateway.

Command
GW_HOUSE_STATUS_MONITOR_ENABLE_REQ

Table 94 - GW_HOUSE_STATUS_MONITOR_ENABLE_REQ frame format.

8.2.2 GW_HOUSE_STATUS_MONITOR_ENABLE_CFM

GW_HOUSE_STATUS_MONITOR_ENABLE_CFM is acknowledged to GW_HOUSE_STATUS_MONITOR_ENABLE_REQ.

Command
GW_HOUSE_STATUS_MONITOR_ENABLE_CFM

Table 95 - GW_HOUSE_STATUS_MONITOR_ENABLE_CFM frame format.

8.2.3 GW_HOUSE_STATUS_MONITOR_DISABLE_REQ

House Status Monitor service will be disabled if GW_HOUSE_STATUS_MONITOR_DISABLE_REQ is send to gateway.

Command
GW_HOUSE_STATUS_MONITOR_DISABLE_REQ

Table 96 - GW_HOUSE_STATUS_MONITOR_DISABLE_REQ frame format.

8.2.4 GW_HOUSE_STATUS_MONITOR_DISABLE_CFM

GW_HOUSE_STATUS_MONITOR_DISABLE_CFM is acknowledged to
GW_HOUSE_STATUS_MONITOR_DISABLE_REQ.

Command
GW_HOUSE_STATUS_MONITOR_DISABLE_CFM

Table 97 - GW_HOUSE_STATUS_MONITOR_DISABLE_CFM frame format.

8.3 Node information

8.3.1 GW_GET_NODE_INFORMATION_REQ

Command	Data 1
GW_GET_NODE_INFORMATION_REQ	NodeID

Table 98 - GW_GET_NODE_INFORMATION_REQ frame format

8.3.1.1 NodeID

NodeID is an Actuator index in the system table, to get information from. It must be a
value from 0 to 199.

8.3.2 GW_GET_NODE_INFORMATION_CFM

Command	Data 1	Data 2
GW_GET_NODE_INFORMATION_CFM	Status	NodeID

Table 99 - GW_GET_NODE_INFORMATION_CFM frame format

8.3.2.1 Status

Status value	Description
0	OK - Request accepted
1	Error – Request rejected
2	Error – Invalid node index
Other values	Reserved

Table 100 - Status parameter

8.3.2.2 NodeID

NodeID is an Actuator index in the system table, to get information from. It must be a
value from 0 to 199.

8.3.3 GW_GET_NODE_INFORMATION_NTF

Command	Data 1	Data 2 - 3	Data 4	Data 5 - 68	Data 69
GW_GET_ALL_NODES_INFORMATION_NTF	NodeID	Order	Placement	Name	Velocity

Data 70 - 71	Data 72	Data 73	Data 74	Data 75	Data 76
NodeTypeSubType	ProductGroup	ProductType	NodeVariation	PowerMode	BuildNumber

Data 77 - 84	Data 85	Data 86 - 87	Data 88 - 89	Data 90 - 91	Data 92 - 93
SerialNumber	State	CurrentPosition	Target	FP1CurrentPosition	FP2CurrentPosition

Data 94 - 95	Data 96 - 97	Data 98 - 99	Data 100 - 103	Data 104	Data 105 - 125
FP3CurrentPosition	FP4CurrentPosition	RemainingTime	TimeStamp	NbrOfAlias	AliasArray

Table 101 - GW_GET_NODE_INFORMATION_NTF frame format

8.3.3.1 NodeID

See paragraph 8.3.1.1 page 41 for description.

8.3.3.2 Order

Order can be used to store a sort order. The sort order is used in client end, when presenting a list of nodes for the user. This field is set and read by client only.

8.3.3.3 Placement

Placement can be used to store a room group index or house group index number.

8.3.3.4 Name

This field Name holds the name of the actuator, ex. "Window 1". This field is 64 bytes long, formatted as UTF-8 characters.

8.3.3.5 Velocity

This field indicates what velocity the node is operation with.

Velocity value	Tag	Description
0	*DEFAULT*	The node operates by its default velocity.
1	*SILENT*	The node operates in silent mode (slow).
2	*FAST*	The node operates with fast velocity.
3-254	-	Not defined value.
255	*VELOCITY_NOT_AVAILABLE*	Not supported by node.

Table 102 - Velocity parameter

*) Some old actuators will only move at default velocity, even if speed parameter are set to SILENT or FAST

8.3.3.6 NodeTypeSubType

This field indicates the node type, ex. Window, Roller shutter, Light etc.
See Table 276 at page 105 for translation of NodeType/SubType value.

8.3.3.7 ProductType

This field indicates what type of product within the type parameter, the nodes is. Ex. KMG, KMX etc.

8.3.3.8 NodeVariation

The node variation.

NodeVariation value	Tag	Description
0	*NOT_SET*	Not set
1	*TOPHUNG*	Window is a top hung window
2	*KIP*	Window is a kip window.
3	*FLAT_ROOF*	Window is a flat roof.
4	*SKY_LIGHT*	Window is a sky light.

Table 103 - NodeVariation parameter

8.3.3.9 PowerMode

This field indicates the power mode of the node.

PowerMode value	Description
0	ALWAYS_ALIVE
1	LOW_POWER_MODE

Table 104 - State parameter

8.3.3.10   SerialNumber

This field tells the serial number of the node. This field is 8 bytes.

8.3.3.11   State

This field indicates the operating state of the node.

State value	Tag	Description
0	Non-executing	This status information is only returned about an ACTIAVTE_FUNC, an ACTIVATE_MODE, an ACTIVATE_STATE or a WINK command. The parameter is unable to execute due to given conditions. An example can be that the temperature is too high. It indicates that the parameter could not execute per the contents of the present activate command.
1	Error while execution	This status information is only returned about an ACTIVATE_STATUS_REQ command. An error has occurred while executing. This error information will be cleared the next time the parameter is going into 'Waiting for executing', 'Waiting for power' or 'Executing'. A parameter can have the execute status 'Error while executing' only if the previous execute status was 'Executing'. Note that this execute status gives information about the previous execution of the parameter, and gives no indication whether the following execution will fail.
2	'Not used'	
3	Waiting for power	The parameter is waiting for power to proceed execution
4	Executing	Execution for the parameter is in progress
5	Done	The parameter is not executing and no error has been detected. No activation of the parameter has been initiated. The parameter is ready for activation.
255	State unknown	The state is unknown

Table 105 - State parameter

8.3.3.12   CurrentPosition

This field indicates the current position of the node. This will be a relative value (0000_{HEX} - $C800_{HEX}$) or 'No feed-back value known' ($F7FF_{HEX}$) in case the current position is outside the relative value range or the current position is not known.

8.3.3.13   Target

This field indicates the target position of the current operation. This will be a relative value (0000_{HEX} - $C800_{HEX}$) or 'No feed-back value known' ($F7FF_{HEX}$) in case the target position is outside the relative value range or the target position is not known.

8.3.3.14   FP1CurrentPosition

This field indicates the current position of functional parameter 1. This will be a relative value (0000_{HEX} - $C800_{HEX}$) or 'No feed-back value known' ($F7FF_{HEX}$) in case the FP1 current position is outside the relative value range or the FP1 current position is not known.

8.3.3.15   BuildNumber

Software Build number of actuator software.

8.3.3.16   FP2CurrentPosition

This field indicates the current position of functional parameter 2. This will be a relative value (0000_{HEX} - $C800_{HEX}$) or 'No feed-back value known' ($F7FF_{HEX}$) in case the FP2 current position is outside the relative value range or the FP2 current position is not known.

8.3.3.17 FP3CurrentPosition

This field indicates the current position of functional parameter 3. This will be a relative value (0000_{HEX} - $C800_{HEX}$) or 'No feed-back value known' ($F7FF_{HEX}$) in case the FP3 current position is outside the relative value range or the FP3 current position is not known.

8.3.3.18 FP4CurrentPosition

This field indicates the current position of functional parameter 4. This will be a relative value (0000_{HEX} - $C800_{HEX}$) or 'No feed-back value known' ($F7FF_{HEX}$) in case the FP4 current position is outside the relative value range or the FP4 current position is not known.

8.3.3.19 RemainingTime

This field indicates the remaining time for a node activation in seconds. If 0 is returned remaining time is unknown or node has reached its target position.

8.3.3.20 TimeStamp

UTC time stamp for last known position.

8.3.3.21 NbrOfAlias

This field indicates the number of alias these nodes contains. Max number of alias is 5.

8.3.3.22 Alias

The parameter Alias is an array of alias'. See Table 106.

Data 1 – 4	Data 5 – 8	…
Object one in Alias	Object two in Alias	…

Table 106 - Frame format of the parameter Alias.

Each Alias contains the type of alias, and the value of the alias. See the Alias structure in Table 107

Data 1 - 2	Data 3 – 4
Type	Value

Table 107 - Alias structure.

8.3.3.22.1 Type

The alias type is the value found as 'Alias ID' in paragraph 14.2 Alias for actuator specific parameter values at page 106.

8.3.3.22.2 Value

The parameter value for the alias.

8.3.4 GW_SET_NODE_VARIATION_REQ

Set the node variation on a node.
Ex. Window type can be set to either kip or top hung.

Command	Data 1	Data 2
GW_SET_NODE_VARIATION_REQ	NodeID	NodeVariation

Table 108 - GW_SET_NODE_VARIATION_REQ frame format

8.3.4.1 NodeID

See paragraph 8.3.1.1 page 41 for description.

8.3.4.2 NodeVariation

The node variation to set.

NodeVariation value	Tag	Description
0	*NOT_SET*	Not set
1	*TOPHUNG*	Window is a top hung window
2	*KIP*	Window is a kip window.
3	*FLAT_ROOF*	Window is a flat roof.
4	*SKY_LIGHT*	Window is a sky light.

Table 109 - NodeVariation parameter

8.3.5 GW_SET_NODE_VARIATION_CFM

Command	Data 1	Data 2
GW_SET_NODE_VARIATION_CFM	Status	NodeID

Table 110 - GW_SET_NODE_VARIATION_CFM frame format

8.3.5.1 Status

Status value	Description
0	OK - Request accepted
1	Error – Request rejected
2	Error – Invalid system table index
Other values	Reserved

Table 111 - Status parameter

8.3.5.2 NodeID

NodeID is the index of that node had its node variation changed. See paragraph 8.3.1.1 page 41 for further description.

8.3.6 GW_SET_NODE_NAME_REQ

Command	Data 1	Data 2 - 65
GW_SET_NODE_NAME_REQ	NodeID	Name

Table 112 - GW_SET_NODE_NAME_REQ frame format

8.3.6.1 NodeID

See paragraph 8.3.1.1 page 41 for description.

8.3.6.2 Name

The name to set, ex. "Johns window". This field is 64 bytes, formatted as UTF-8 character.

8.3.7 GW_SET_NODE_NAME_CFM

Command	Data 1	Data 2
GW_SET_NODE_NAME_CFM	Status	NodeID

Table 113 - GW_SET_NODE_NAME_CFM frame format

8.3.7.1 Status

Status value	Description
0	OK - Request accepted
1	Error – Request rejected
2	Error – Invalid system table index
Other values	Reserved

Table 114 - Status parameter

8.3.7.2 NodeID

NodeID is the index of the node that had its name changed. See paragraph 8.3.1.1 page 41 for further description.

8.3.8 GW_NODE_INFORMATION_CHANGED_NTF

Command	Data 1	Data 2 - 65	Data 66 - 67	Data 68
GW_NODE_INFORMATION_CHANGED_NTF	NodeID	Name	Order	Placement

Data 69
NodeVariation

Table 115 - GW_NODE_INFORMATION_CHANGED_NTF frame format.

8.3.8.1 Parameter description

A detailed parameter description can be found as subparagraphs under paragraph 8.3.3 at page 41.

8.3.9 GW_NODE_STATE_POSITION_CHANGED_NTF

If House Status Monitor has been enabled then GW_NODE_STATE_POSITION_CHANGED_NTF will be send when somebody change state or position on a known actuator.

Command	Data 1	Data 2	Data 3 - 4	Data 5 - 6
GW_NODE_STATE_POSITION_CHANGED_NTF	NodeID	State	CurrentPosition	Target

Data 7 - 8	Data 9 - 10	Data 11 -12	Data 13 - 14	Data 15 - 16
FP1CurrentPosition	FP2CurrentPosition	FP3CurrentPosition	FP4CurrentPosition	RemainingTime

Data 17 - 20
TimeStamp

Table 116 - GW_NODE_INFORMATION_CHANGED_NTF frame format.

8.3.10 GW_GET_ALL_NODES_INFORMATION_REQ

This event will get the information on all nodes. Every node information is sent in a GW_GET_ALL_NODES_INFORMATION_NTF event. The event GW_GET_ALL_NODES_INFORMATION_FINISHED_NTF is sent after the last node information.

Command
GW_GET_ALL_NODES_INFORMATION_REQ

Table 117 - GW_GET_ALL_NODES_INFORMATION_REQ frame format.

8.3.11 GW_GET_ALL_NODES_INFORMATION_CFM

Command	Data 1	Data 2
GW_GET_ALL_NODES_INFORMATION_CFM	Status	TotalNumberOfNodes

Table 118 – GW_GET_ALL_NODES_INFORMATION_CFM

This event is sent as a confirm on GW_GET_ALL_NODES_INFORMATION_REQ.

8.3.11.1 Status

Status value	Description
0	OK - Request accepted
1	Error – System table empty
Other values	Reserved

Table 119 - Status parameter

8.3.12 GW_GET_ALL_NODES_INFORMATION_NTF

This event holds the information on a node.

Command	Data 1	Data 2 - 3	Data 4	Data 5 - 68	Data 69
GW_GET_ALL_NODES_INFORMATION_NTF	NodeID	Order	Placement	Name	Velocity

Data 70 - 71	Data 72	Data 73	Data 74	Data 75	Data 76
NodeTypeSubType	ProductGroup	ProductType	NodeVariation	PowerMode	BuildNumber

Data 77 - 84	Data 85	Data 86 - 87	Data 88 - 89	Data 90 - 91	Data 92 - 93
SerialNumber	State	CurrentPosition	Target	FP1CurrentPosition	FP2CurrentPosition

Data 94 - 95	Data 96 - 97	Data 98 - 99	Data 100 - 103	Data 104	Data 105 - 125
FP3CurrentPosition	FP4CurrentPosition	RemainingTime	TimeStamp	NbrOfAlias	AliasArray

Table 120 - GW_GET_ALL_NODES_INFORMATION_NTF frame format.

8.3.12.1 Parameter description

A detailed parameter description can be found as subparagraphs under paragraph 8.3.3 at page 41.

8.3.13 GW_GET_ALL_NODES_INFORMATION_FINISHED_NTF

This event is sent after the last node information, indicating no more nodes.

Command
GW_GET_ALL_NODES_INFORMATION_FINISHED_NTF

Table 121 - GW_GET_ALL_NODES_INFORMATION_CFM frame format.

8.3.14 GW_SET_NODE_ORDER_AND_PLACEMENT_REQ

GW_SET_NODE_ORDER_AND_PLACEMENT_REQ are used to set new sort order and room placement parameter for a given actuator node.

Command	Data 1	Data 2 - 3	Data 4
GW_SET_NODE_ORDER_AND_PLACEMENT_REQ	NodeID	Order	Placement

Table 122 - GW_SET_NODE_ORDER_AND_PLACEMENT_REQ frame format.

8.3.14.1 NodeID

See paragraph 8.3.1.1 page 41 for description.

8.3.14.2 Order

Order can be used to store a sort order. The sort order is used in client end, when presenting a list of nodes for the user.

8.3.14.3 Placement

Placement can be used to store a room group index or house group index number.

8.3.15 GW_SET_NODE_ORDER_AND_PLACEMENT_CFM

GW_SET_NODE_ORDER_AND_PLACEMENT_CFM is acknowledged to GW_SET_NODE_ORDER_AND_PLACEMENT_REQ.

Command	Data 1	Data 2
GW_SET_NODE_ORDER_AND_PLACEMENT_CFM	Status	NodeID

Table 123 - GW_SET_NODE_ORDER_AND_PLACEMENT_CFM frame format

8.3.15.1 Status

Status value	Description
0	OK - Request accepted
1	Error – Request rejected
2	Error – Invalid system table index
Other values	Reserved

Table 124 - Status parameter

8.3.15.2 NodeID

NodeID of the node that had its Order and Placement changed. See paragraph 8.3.1.1 page 41 for further description.

8.4 Group information

The gateway can hold up to 100 groups. A group is a collection of actuator nodes in conjunction with a name and some other come characteristics.
There are three different group types. House, Room and User defined. There can be only one instance of the group type house. The GroupID = 0 is reserved for the house group.
An actuator can only be represented in one room group. So, if an actuator is assigned to a room group is will automatically be removed from another existing room group.

8.4.1 GW_GET_GROUP_INFORMATION_REQ

Command	Data 1
GW_GET_GROUP_INFORMATION_REQ	GroupID

Table 125 - GW_GET_GROUP_INFORMATION_REQ frame format.

8.4.1.1 GroupID

GroupID indicates the group ID to get information from.

8.4.2 GW_GET_GROUP_INFORMATION_CFM

Command	Data 1	Data 2
GW_GET_GROUP_INFORMATION_CFM	Status	GroupID

Table 126 – GW_DELETE_GROUP_INFORMATION_CFM frame format.

8.4.2.1 Status

Status value	Description
0	OK – Request accepted
1	Error – Request failed
2	Error – Invalid group index
Other values	Reserved

Table 127 - Status parameter description.

8.4.2.2 GroupID

GroupID is the group index.

8.4.3 GW_GET_GROUP_INFORMATION_NTF

Command	Data 1	Data 2 - 3	Data 4	Data 5 - 68
GW_GET_GROUP_INFORMATION_NTF	GroupID	Order	Placement	Name

Data 69	Data 70	Data 71	Data 72	Data 73 – 97	Data 98 – 99
Velocity	NodeVariation	GroupType	NbrOfObjects	ActuatorBitArray	Revision

Table 128 - GW_GET_GROUP_INFORMATION_NTF frame format.

8.4.3.1 GroupID

GroupID is the group index.

8.4.3.2 Order

Order can be used to store a sort order. The sort order is used in client end, when presenting a list of groups for the user.

8.4.3.3 Placment

Placement can be used to store a room group index or house group index number.

8.4.3.4 Name

This field holds the name of the group, ex. "My Group". This field is 64 bytes long, formatted as UTF-8 characters.

8.4.3.5 Velocity

This field indicates what velocity the nodes in the group are operation with.

Velocity value	Tag	Description
0	*DEFAULT*	The node operates by its default velocity.
1	*SILENT*	The node operates in silent mode (slow).
2	*FAST*	The node operates with fast velocity.
3-255	-	Not defined value.

Table 129 – Velocity parameter.

*) Some old actuators will only move at default velocity, even if speed parameter are set to SILENT or FAST

8.4.3.6 NodeVariation

The node variation. This is not used by KLF200. It is only stored for the client.

NodeVariation value	Tag	Description
0	*NOT_SET*	Not set
1	*TOPHUNG*	Window is a top hung window
2	*KIP*	Window is a kip window.
3	*FLAT_ROOF*	Window is a flat roof.
4	*SKY_LIGHT*	Window is a sky light.

Table 130 - NodeVariation parameter.

8.4.3.7 GroupType

This field indicates the type of the group.

GroupType value	Tag	Description
0	*USER_GROUP*	The group type is a user group.
1	*ROOM*	The group type is a Room.
2	*HOUSE*	The group type is a House.

Table 131 – GroupType parameter.

8.4.3.8 NbrOfObjects

This field indicates the number of objects the group contains.
Note: NbrOfObjects is used only when GroupType = USER_GROUP. If GroupType is ROOM or HOUSE, then NbrOfObjects data must be ingored.

8.4.3.9 ActuatorBitArray

The parameter ActuatorBitArray is a 25-byte long bit array. Least significant bit in first byte holds information of the actuator node with index 0 and most significant bit of the

last byte is for actuator with index 199. If a bit is set the given actuator is a part of the group.

Note: ActuatorBitArray is used only when GroupType = USER_GROUP. If GroupType is ROOM or HOUSE, then ActuatorBitArray data must be ingored.

8.4.3.10   Revision

Revision number for group data. This revision number is only used for validation of the input data received from EV_GW_SET_GROUP_INFORMATION_REQ. Group data is only accepted if Revision number in EV_GW_SET_GROUP_INFORMATION_REQ is equal to the revision number in the existing group data.

8.4.4 GW_NEW_GROUP_REQ

GW_NEW_GROUP_INFORMATION_REQ is used to create a new group object. A valid group must contain two or more actuators and the actuators must be of the same type.

Command	Data 1 - 2	Data 3	Data 4 - 67	Data 68	Data 69	Data 70
GW_NEW_GROUP_REQ	Order	Placement	Name	Velocity	NodeVariation	GroupType

Data 71	Data 72 – 96
NbrOfObjects	ActuatorBitArray

Table 132 – GW_NEW_GROUP_REQ frame format.

8.4.4.1 GroupType

This field indicates the type of the group.

GroupType value	Tag	Description
0	*USER_GROUP*	The group type is a user group.
1	*ROOM*	The group type is a Room.
2	*HOUSE*	The group type is a House. Se note.
3	*ALL-GROUP*	The group type is an All-group. Se note.

Table 133 – GroupType parameter.

Note: Only one instance of GroupType = House is allowed. Because it is already set up from the beginning, you can obviously not re-created a house group. It is not possible to create a new All-group. All-groups are created and updated automatically.

8.4.4.2 Parameter description for remaining parameters

A detailed parameter description for remaining parameters can be found as subparagraphs under paragraph 8.4.3 at page 48.

8.4.5 GW_NEW_GROUP_CFM

Command	Data 1	Data 2
GW_NEW_GROUP_CFM	Status	GroupID

Table 134 – GW_NEW_GROUP_CFM frame format.

8.4.5.1 Status

Status value	Description
0	OK – Request accepted
1	Error – Request failed
2	Error – Invalid parameter
Other values	Reserved

Table 135 – Status parameter description.

8.4.5.2 GroupID

Identification number for new group.

8.4.6 GW_SET_GROUP_INFORMATION_REQ

GW_SET_GROUP_INFORMATION_REQ is used to edit an existing group object. If no group are found with the given GroupID, a GW_ERROR_NTF will be returned instead of GW_SET_GROUP_INFORMATION_CFM.
A valid group must contain two or more actuators and the actuators must be at the same type.

Command	Data 1	Data 2 – 3	Data 4	Data 5 – 68
GW_SET_GROUP_INFORMATION_REQ	GroupID	Order	Placement	Name

Data 69	Data 70	Data 71	Data 72	Data 73 – 97	Data 98 – 99
Velocity	NodeVariation	GroupType	NbrOfObjects	ActuatorBitArray	Revision

Table 136 – GW_SET_GROUP_INFORMATION_REQ frame format.

8.4.6.1 GroupID

GroupID is the group index. The GroupID = 0 is reserved for GroupType = house.

8.4.6.2 GroupType

The GroupType cannot be changed for an existing group. Therefore, the GroupType in the GW_SET_GROUP_INFORMATION_REQ frame must be equal to the GroupType of the group which is to be edited. If this is not the case, the Status field in GW_SET_GROUP_INFORMATION_CFM will indicate that the request has failed. It is not possible to edit an All-group.

8.4.6.3 Parameter description for remaining parameters

A detailed parameter description for remaining parameters can be found as subparagraphs under paragraph 8.4.3 at page 48.

8.4.7 GW_SET_GROUP_INFORMATION_CFM

Command	Data 1	Data 2
GW_SET_GROUP_INFORMATION_CFM	Status	GroupID

Table 137 – GW_SET_GROUP_INFORMATION_CFM frame format.

8.4.7.1 Status

This field indicates if the request was successful.

Status value	Description
0	The request was successful.
1	Failed. Command rejected.
2	Failed. Invalid parameter.

Table 138 – Status parameter.

8.4.7.2 GroupID

GroupID is the group index from the request.

8.4.8 GW_DELETE_GROUP_REQ

Command	Data 1
GW_DELETE_GROUP_REQ	GroupID

Table 139 – GW_DELETE_GROUP_REQ frame format.

8.4.8.1 GroupID

GroupID is the group index.

8.4.9 GW_DELETE_GROUP_CFM

Command	Data 1	Data 2
GW_DELETE_GROUP_CFM	GroupID	Status

Table 140 – GW_DELETE_GROUP_CFM frame format.

8.4.9.1 GroupID

GroupID is the group index from the request.

8.4.9.2 Status

This field indicates if the request was successful.

Status value	Description
0	OK – Request accepted
1	Error – Request failed
2	Error – Invalid group index
Other values	Reserved

Table 141 – Status parameter.

8.4.10 GW_GROUP_DELETED_NTF

GW_GROUP_DELETED_NTF is broadcasted to all, when a group has been removed.

Command	Data 1
GW_GROUP_DELETED_NTF	GroupID

Table 142 - GW_GROUP_DELETED_NTF frame format.

8.4.11 GW_GET_ALL_GROUPS_INFORMATION_REQ

Command	Data 1	Data 2
GW_GET_ALL_GROUPS_INFORMATION_REQ	UseFilter	GroupType

Table 143 – GW_GET_ALL_GROUPS_INFORMATION_REQ frame format.

8.4.11.1 UseFilter

If bUseFilter == 0 then request information for all groups. Else only request information for groups with same type as defined by GroupType parameter.

8.4.11.2 GroupType

If UseFilter is set, then only request information for groups with same type as set by GroupType parameter.

GroupType value	Tag	Description
0	*USER_GROUP*	The group type is a user group.
1	*ROOM*	The group type is a Room.
2	*HOUSE*	The group type is a House.

Table 144 – GroupType parameter.

8.4.12 GW_GET_ALL_GROUPS_INFORMATION_CFM

Command	Data 1	Data 2
GW_GET_ALL_GROUPS_INFORMATION_CFM	Status	TotalNumberOfGroups

Table 145 - GW_GET_ALL_GROUPS_INFORMATION_CFM frame format.

8.4.12.1 Status

Status value	Description
0	OK – Request accepted
1	Error – Request failed
2	Error – No groups available
Other values	Reserved

Table 146 - Status parameter description

8.4.13 GW_GET_ALL_GROUPS_INFORMATION_NTF

Command	Data 1	Data 2 - 3	Data 4	Data 5 - 68
GW_GET_ALL_GROUPS_INFORMATION_NTF	GroupID	Order	Placement	Name

Data 69	Data 70	Data 71	Data 72	Data 73 – 97	Data 98 – 99
Velocity	NodeVariation	GroupType	NbrOfObjects	ActuatorBitArray	Revision

Table 147 - GW_GET_ALL_GROUPS_INFORMATION_NTF frame format.

8.4.13.1 Parameter description

A detailed parameter description can be found as subparagraphs under paragraph 8.4.3 at page 48.

8.4.14 GW_GET_ALL_GROUPS_INFORMATION_FINISHED_NTF

Command
GW_GET_ALL_GROUPS_INFORMATION_FINISHED_NTF

Table 148 - GW_GET_ALL_GROUPS_INFORMATION_FINISHED_NTF frame format.

8.4.15 GW_GROUP_INFORMATION_CHANGED_NTF

If a group has been changed, a GW_GROUP_INFORMATION_CHANGED_NTF will be broadcasted to all connected clients.

Command	Data 1	Data 2
GW_GROUP_INFORMATION_CHANGED_NTF	ChangeType = "Group Deleted"	GroupID

Table 149 - GW_GROUP_INFORMATION_CHANGED_NTF frame format when a group is deleted.

Command	Data 1	Data 2
GW_GROUP_INFORMATION_CHANGED_NTF	ChangeType = "Information Modified"	GroupID

Data 3 - 4	Data 5	Data 6 - 69	Data 70	Data 71	Data 72	Data 73
Order	Placement	Name	Velocity	NodeVariation	GroupType	NbrOfObjects

Data 74 – 98	Data 99 – 100
ActuatorBitArray	Revision

Table 150 - GW_GROUP_INFORMATION_CHANGED_NTF frame format when group information has changed.

8.4.15.1 ChangeType

ChangeType value	Field name
0	Group Deleted
1	Information modified
Other values	Reserved

Table 151 – ChangeType value description

8.4.15.2 Parameter description

A detailed parameter description can be found as subparagraphs under paragraph 8.4.3 at page 48.

9 Activation Log

9.1.1 GW_GET_ACTIVATION_LOG_HEADER_REQ

Request data from the log header. Contains info about how many are the system has logged

Command
GW_GET_ACTIVATION_LOG_HEADER_REQ

Table 152 - GW_GET_ACTIVATION_LOG_HEADER_REQ frame format.

9.1.2 GW_GET_ACTIVATION_LOG_HEADER_CFM

Command	Data 1-2	Data 3 - 4
GW_GET_ACTIVATION_LOG_HEADER_CFM	MaxLineCount	LineCount

Table 153 - GW_GET_ACTIVATION_LOG_HEADER_CFM frame format.

9.1.2.1 MaxLineCount parameter

Max number of possible lines in log. MaxLineCount is an unsigned 16-bit integer.

9.1.2.2 LineCount parameter

The current number of lines in log. LineCount is an unsigned 16-bit integer.

9.1.3 GW_CLEAR_ACTIVATION_LOG_REQ

Request that the log is cleared.

Command
GW_CLEAR_ACTIVATION_LOG_REQ

Table 154 - GW_CLEAR_ACTIVATION_LOG_REQ frame format.

9.1.4 GW_CLEAR_ACTIVATION_LOG_CFM

Command
GW_CLEAR_ACTIVATION_LOG_CFM

Table 155 - GW_CLEAR_ACTIVATION_LOG_CFM frame format.

9.1.5 GW_GET_ACTIVATION_LOG_LINE_REQ

Request data from one error line in log. The latest error is placed on line 0.

Command	Data 1-2
GW_GET_ACTIVATION_LOG_LINE_REQ	Line

Table 156 - GW_GET_ACTIVATION_LOG_LINE_REQ frame format.

9.1.5.1 Line parameter

Request data from this line. Line is an unsigned 16-bit integer.

9.1.6 GW_GET_ACTIVATION_LOG_LINE_CFM

Command	Data 1 - 4	Data 5 - 6	Data 7	Data 8	Data 9
GW_GET_ACTIVATION_LOG_LINE_CFM	TimeStamp	SessionID	StatusID	Index	NodeParameter

Data 10 – 11	Data 12	Data 13	Data 14 - 17
ParameterValue	RunStatus	StatusReply	InformationCode

Table 157 - GW_GET_ACTIVATION_LOG_LINE_CFM frame format.

9.1.6.1 TimeStamp parameter

UNIX time stamp for when the error occurred.

9.1.6.2 Parameter Data 5 to 17

See GW_COMMAND_RUN_STATUS_NTF

9.1.7 GW_GET_MULTIPLE_ACTIVATION_LOG_LINES_REQ

Request data from several error lines in log, from a specified UNIX timestamp.

Command	Data 1-4
GW_GET_MULTIPLE_ACTIVATION_LOG_LINES_REQ	TimeStamp

Table 158 - GW_GET_MULTIPLE_ACTIVATION_LOG_LINES_REQ frame format.

9.1.7.1 Timestamp parameter

Request data from this timestamp and all following errors after this time. Timestamp is an unsigned 32-bit integer.

9.1.8 GW_GET_MULTIPLE_ACTIVATION_LOG_LINES_NTF

A notify will be sent to client for each log line found.

Command	Data 1 - 4	Data 5 - 6	Data 7	Data 8
GW_GET_MULTIPLE_ACTIVATION_LOG_LINES_NTF	TimeStamp	SessionID	StatusID	Index

Data 9	Data 10 – 11	Data 12	Data 13	Data 14 - 17
NodeParameter	ParameterValue	RunStatus	StatusReply	InformationCode

Table 159 - GW_GET_MULTIPLE_ACTIVATION_LOG_LINES_NTF frame format.

9.1.8.1 TimeStamp parameter

UNIX time stamp for when the error occurred.

9.1.8.2 Parameter Data 5 to 17

See GW_COMMAND_RUN_STATUS_NTF

9.1.9 GW_GET_MULTIPLE_ACTIVATION_LOG_LINES_CFM

Confirm is sent to client after last notify frame. If no errors were found in log, the request frame will just be answered with a confirm.

Command	Data 1+2	Data 3
GW_GET_MULTIPLE_ACTIVATION_LOG_LINES_CFM	LineCount	Status

Table 160 - GW_GET_MULTIPLE_ACTIVATION_LOG_LINES_CFM frame format.

9.1.9.1 LineCount parameter

Number of lines in log, which has been sent as notifies.
Client can choose to compare this number to received notifies. This will indicate if some notifies were lost from gateway to client.
LineCount is an unsigned 16-bit integer.

9.1.9.2 Status parameter

Status parameter shows status of request for log lines.
Status is an unsigned 8-bit integer.

Status value	Description
0	Get multiple activation log lines request failed.
1	Get multiple activation log lines request succeeded.

Table 161 - Status parameter description.

9.1.10 GW_ACTIVATION_LOG_UPDATED_NTF

This notify is send every time a new error is written in the log.

Command
GW_ACTIVATION_LOG_UPDATED_NTF

Table 162 - GW_ACTIVATION_LOG_UPDATED_NTF frame format.

10 Command Handler

Command Handler is a service to operate one or more io-homecontrol® nodes directly.
The Command Handler operational area is divided into following main groups:

- Send activating command
- Stop
- Status request
- Wink
- Limitation
- Mode
- Activate predefined Group

10.1 Send activating command

Figure 15 - Sequence diagram, Send activating command.

10.1.1 GW_COMMAND_SEND_REQ

When the gateway receives a GW_COMMAND_SEND_REQ frame, it will set a new actuator position in one or more actuators.

Command	Data 1 – 2	Data 3	Data 4	Data 5
GW_COMMAND_SEND_REQ	SessionID	CommandOriginator	PriorityLevel	ParameterActive

Data 6	Data 7	Data 8 - 41	Data 42	Data 43 – 62	Data 63
FPI1	FPI2	FunctionalParameterValueArray	IndexArrayCount	IndexArray	PriorityLevelLock

Data 64	Data 65	Data 66
PL_0_3	PL_4_7	LockTime

Table 163 - GW_COMMAND_SEND_REQ frame format.

10.1.1.1   SessionID parameter

SessionID is at 16-bit unsigned integer. SessionID is used to give unique identifications of the command. SessionID value in GW_COMMAND_SEND_REQ will be returned in all GW_COMMAND_SEND_CFM, GW_COMMAND_RUN_STATUS_NTF, GW_COMMAND_REMAINING_TIME_NTF and GW_SESSION_FINISHED_NTF belonging the same session. Make sure that you change SessionID for each session. Typical SessionID is incremented by one.

10.1.1.2   CommandOriginator parameter

Specifies the command originator type (USER/TIMER/SECURITY etc.)
Typically, only USER or SAAC are used.

CommandOriginator value	Tag	Description
1	USER	User Remote control causing action on actuator
2	RAIN	Rain sensor
3	TIMER	Timer controlled
5	UPS	UPS unit
8	SAAC	Stand Alone Automatic Controls
9	WIND	Wind sensor
11	LOAD_SHEDDING	Managers for requiring a particular electric load shed.
12	LOCAL_LIGHT	Local light sensor.
13	UNSPECIFIC_ENVIRONMENT_SENSOR	Used in context with commands transmitted on basis of an unknown sensor for protection of an end-product or house goods.
255	EMERGENCY	Used in context with emergency or security commands

Table 164 - CommandOriginator parameter description

10.1.1.3   PriorityLevel parameter

PriorityLevel defines the priority level, of the activating command.
There are 8 priority levels see Table 165. The 8 priority levels are divided into 3 different groups: Protection (PL0-1), User (PL2-3) and Comfort (PL4-7).
Typically, PriorityLevel will be set to '3' for user level 2 or '5' for Comfort Level 2.

Group	Priority Level Number	Class	Description
Protection	0	Human Protection	Provide the most secured level. Since consequences of misusing this level can deeply impact the system behaviour, and therefore the io-homecontrol® image, it is mandatory for the manufacturer that wants to use this level of priority to receive an agreement from io-homecontrol® In any case the reception of such a command will disable all categories (Level 0 to 7).
	1	Environment Protection	Used by local sensors that are relative to goods protection: end-product protection, house goods protection. *Examples: wind sensor on a terrace awning, rain sensor on a roof window, etc.*
User	2	User Level 1	Used by controller to send one (or a set of one shot) immediate action commands when user manually requested for this. Controllers prescribed as having a higher level of priority than others use this level. *For example, this level can be used in combination with a lock command on other levels of priority, for providing an exclusive access to actuators control. e.g Parents/Children different access rights, …*
	3	User Level 2	Used by controller to send one (or a set of one shot) immediate action commands when user manually requested for this. This level is the default level used by controllers.
Comfort	4	Comfort Level 1	TBD. Don't use
	5	Comfort Level 2	Used by Stand Alone Automatic Controls
	6	Comfort Level 3	TBD. Don't use
	7	Comfort Level 4	TBD. Don't use

Table 165 - Priority Level Groups and Class.

10.1.1.4 ParameterActive parameter

GW_COMMAND_RUN_STATUS_NTF frame carries the current value of one parameter. The ParameterActive parameter in GW_COMMAND_SEND_REQ frame is used to indicate which parameter status is requested for. Default let ParameterActive = 0.

ParameterActive value	Tag	Description
0	MP	Main Parameter.
1	FP1	Functional Parameter number 1.
2	FP2	Functional Parameter number 2.
3	FP3	Functional Parameter number 3.
4	FP4	Functional Parameter number 4.
5	FP5	Functional Parameter number 5.
6	FP6	Functional Parameter number 6.
7	FP7	Functional Parameter number 7.
8	FP8	Functional Parameter number 8.
9	FP9	Functional Parameter number 9.
10	FP10	Functional Parameter number 10.
11	FP11	Functional Parameter number 11.
12	FP12	Functional Parameter number 12.
13	FP13	Functional Parameter number 13.
14	FP14	Functional Parameter number 14.
15	FP15	Functional Parameter number 15.
16	FP16	Functional Parameter number 16.
Other values		Not allowed.

Table 166 - ParameterActive parameter description

10.1.1.5 FPI1 and FPI2 parameters

The Functional Parameter Indicator 1 (FPI1) and Functional Parameter Indicator 2 (FPI2) bytes are used to indicate which Functional Parameters are included in the frame.

If the FPI1 and FPI2 bytes are included in the frame sent to the Actuator, they will also be included in the acknowledge frame returned from the Actuator.

The FPI1 and FPI2 bytes are read in the following way:
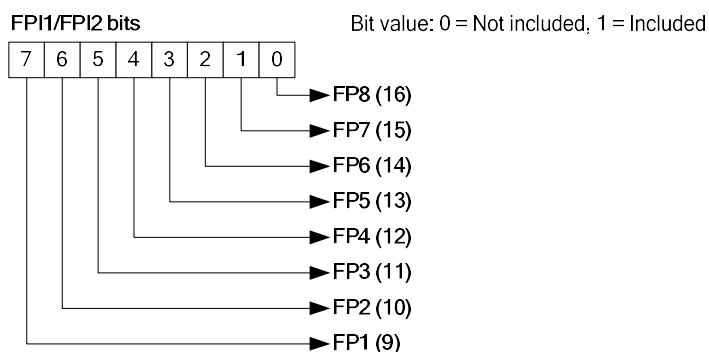
Figure 16 - FPI1/FPI2 bit description

10.1.1.6 FunctionalParameterValueArray parameter

FunctionalParameterValueArray is an array of 16 bit integers. FunctionalParameterValueArray hold the values for main parameter (MP) and all functional parameters. FunctionalParameterValueArray has room for 17 parameter values. Position 0 is the MP value. Position 1 to 16 holds Functional Parameter 1 to 16.

For more information you can read paragraph 13 "Appendix 1: Standard Parameter definition" at page 102 and paragraph 14 "Appendix 2: List of actuator types and their use of Main Parameter and Functional Parameters" at page 104.

10.1.1.7 IndexArrayCount parameter

Number of used indexes in 'IndexArray' parameter. 'IndexArrayCount' must be a number from 1 to 20, both included.

If 'IndexArrayCount' is below 20 then the last byte(s) of 'IndexArray' parameter is ignored.

10.1.1.8 IndexArray parameter

Byte array indicating nodes in the system table. One byte for each node, each byte in array can have value [0;199].
'IndexArray' is always 20 bytes long, even if 'IndexArrayCount' parameter is below 20.
If for example 'IndexArrayCount' parameter is 5, only first 5 bytes of 'IndexArray' is relevant.

10.1.1.9 PriorityLevelLock parameter

PriorityLevelLock tells whether to use priority lock.

PriorityLevelLock value	Description
0	Do not set a new lock on priority level. Information in the parameters PL_0_3, PL_4_7 and LockTime are not used. This is the one typically used.
1	Information in the parameters PL_0_3, PL_4_7 and LockTime are used to lock one or more priority level.

Table 167 - PriorityLevelLock parameter

10.1.1.10   PLI_0_3 and PLI_4_7 parameters
Priority level information

There are eight priority levels. Each priority level can set to one of four values. Those four values are listed in Table 168.

All priority Levels can be individually disabled or enabled with the information carried by the PLI bits, except PL0 that instantly will disable all priority levels to ensure human protection.

PLI Number	Name	Description
0	Disable priority	Disable the priority related to the Master
1	Enable	Enable the priority related to the Master
2	Enable all	Enable all pool entry for the specified priority level Must be used with caution!
3	Keep current	Do not make any action. When used, the priority setting for the specific level will be kept in its current state.

Table 168 - Priority Level Information numbers.

The PLI bits for each priority level are send in the frame as a trailer to the parameters within a parameter management command. To indicate that the frame is carrying the priority level lock information bytes, the PriorityLevelLock value must be set to 1.

Priority Level Lock Information Bytes
These bytes carry the Priority level information on each on the priority levels to manage.

PLI_0_3	PLI_4_7
Bit 7-6 = PLI 0	Bit 7-6 = PLI 4
Bit 5-4 = PLI 1	Bit 5-4 = PLI 5
Bit 3-2 = PLI 2	Bit 3-2 = PLI 6
Bit 1-0 = PLI 3	Bit 1-0 = PLI 7

Table 169 - Priority level lock bytes.

10.1.1.11   LockTime parameter

LockTime defines a common lock time for all priority levels.

LockTime value	Description
0	30 seconds
1	60 seconds
:	:
254	7650 seconds (127 min 30 sec)
255	Unlimited time

Table 170 - LockTime parameter description.

10.1.2 GW_COMMAND_SEND_CFM

The gateway will acknowledge with one GW_COMMAND_SEND_CFM after receiving a GW_COMMAND_SEND_REQ command frame.

Command	Data 1 – 2	Data 3
GW_COMMAND_SEND_CFM	SessionID	Status

Table 171 - GW_COMMAND_SEND_CFM frame format.

10.1.2.1 Status parameter

Indicates if the GW_COMMAND_SEND_REQ command is accepted or rejected by the Command Handler.

Status value	Description
0	Command is rejected
1	Command is accepted
Other values	Not defined

Table 172 - Status parameter description.

10.1.2.2 SessionID parameter

Unique identification of the session. Same value as SessionID in triggering GW_COMMAND_SEND_REQ frame.

10.1.3 GW_COMMAND_RUN_STATUS_NTF

For each actuator addressed by IndexArray in the GW_COMMAND_SEND_REQ frame, the gateway will return with two GW_COMMAND_RUN_STATUS_NTF frames. One before and one after the given actuators movement.

Command	Data 1 - 2	Data 3	Data 4	Data 5	Data 6 – 7
GW_COMMAND_RUN_STATUS_NTF	SessionID	StatusID	Index	NodeParameter	ParameterValue

Data 8	Data 9	Data 10 - 13
RunStatus	StatusReply	InformationCode

Table 173 - GW_COMMAND_RUN_STATUS_NTF frame format.

10.1.3.1 SessionID parameter

SessionID are used to identify the command. SessionID has same value as SessionID parameter in the triggering frame.

10.1.3.2 StatusID parameter

Identification of the status owner.

StatusID value	Tag	Description
0x01	STATUS_USER	The status is from a user activation.
0x02	STATUS_RAIN	The status is from a rain sensor activation.
0x03	STATUS_TIMER	The status is from a timer generated action.
0x05	STATUS_UPS	The status is from a UPS generated action.
0x08	STATUS_PROGRAM	The status is from an automatic program generated action. (SAAC)
0x09	STATUS_WIND	The status is from a Wind sensor generated action.
0x0A	STATUS_MYSELF	The status is from an actuator generated action.
0x0B	STATUS_AUTOMATIC_CYCLE	The status is from a automatic cycle generated action.
0x0C	STATUS_EMERGENCY	The status is from an emergency or a security generated action.
0xFF	STATUS_UNKNOWN	The status is from an unknown command originator action.
Other values		Not defined

Table 174 - StatusID parameter description.

10.1.3.3 Index parameter

Index of the actuator in system table.

10.1.3.4   NodeParameter parameter

Identifies the parameter that ParameterValue carry information about.

NodeParameter value	Tag	Description
0x00	*MP*	Main Parameter.
0x01	*FP1*	Functional Parameter number 1.
0x02	*FP2*	Functional Parameter number 2.
0x03	*FP3*	Functional Parameter number 3.
0x04	*FP4*	Functional Parameter number 4.
0x05	*FP5*	Functional Parameter number 5.
0x06	*FP6*	Functional Parameter number 6.
0x07	*FP7*	Functional Parameter number 7.
0x08	*FP8*	Functional Parameter number 8.
0x09	*FP9*	Functional Parameter number 9.
0x0A	*FP10*	Functional Parameter number 10.
0x0B	*FP11*	Functional Parameter number 11.
0x0C	*FP12*	Functional Parameter number 12.
0x0D	*FP13*	Functional Parameter number 13.
0x0E	*FP14*	Functional Parameter number 14.
0x0F	*FP15*	Functional Parameter number 15.
0x10	*FP16*	Functional Parameter number 16.
0xFF	*NOT_USED*	Value to indicate Functional Parameter not used.

Table 175 - NodeParameter description.

10.1.3.5   ParameterValue parameter

Contains the current value of the active parameter.

10.1.3.6   RunStatus parameter

Contains the execution status of the node.

RunStatus value	Tag	Description
0	*EXECUTION_COMPLETED*	Execution is completed with no errors.
1	*EXECUTION_FAILED*	Execution has failed. (Get specifics in the following error code)
2	*EXECUTION_ACTIVE*	Execution is still active.

Table 176 - RunStatus parameter description.

10.1.3.7   StatusReply parameter

Contains current state of the node. (Error code)

bStatusReply value	Tag	Description
0x00	*UNKNOWN_STATUS_REPLY*	Used to indicate unknown reply.
0x01	*COMMAND_COMPLETED_OK*	Indicates no errors detected.
0x02	*NO_CONTACT*	Indicates no communication to node.
0x03	*MANUALLY_OPERATED*	Indicates manually operated by a user.
0x04	*BLOCKED*	Indicates node has been blocked by an object.
0x05	*WRONG_SYSTEMKEY*	Indicates the node contains a wrong system key.
0x06	*PRIORITY_LEVEL_LOCKED*	Indicates the node is locked on this priority level.
0x07	*REACHED_WRONG_POSITION*	Indicates node has stopped in another position than expected.
0x08	*ERROR_DURING_EXECUTION*	Indicates an error has occurred during execution of command.
0x09	*NO_EXECUTION*	Indicates no movement of the node parameter.
0x0A	*CALIBRATING*	Indicates the node is calibrating the parameters.
0x0B	*POWER_CONSUMPTION_TOO_HIGH*	Indicates the node power consumption is too high.

0x0C	*POWER_CONSUMPTION_TOO_LOW*	Indicates the node power consumption is too low.
0x0D	*LOCK_POSITION_OPEN*	Indicates door lock errors. (Door open during lock command)
0x0E	*MOTION_TIME_TOO_LONG__ COMMUNICATION_ENDED*	Indicates the target was not reached in time.
0x0F	*THERMAL_PROTECTION*	Indicates the node has gone into thermal protection mode.
0x10	*PRODUCT_NOT_OPERATIONAL*	Indicates the node is not currently operational.
0x11	*FILTER_MAINTENANCE_NEEDED*	Indicates the filter needs maintenance.
0x12	*BATTERY_LEVEL*	Indicates the battery level is low.
0x13	*TARGET_MODIFIED*	Indicates the node has modified the target value of the command.
0x14	*MODE_NOT_IMPLEMENTED*	Indicates this node does not support the mode received.
0x15	*COMMAND_INCOMPATIBLE_TO_MOVEMENT*	Indicates the node is unable to move in the right direction.
0x16	*USER_ACTION*	Indicates dead bolt is manually locked during unlock command.
0x17	*DEAD_BOLT_ERROR*	Indicates dead bolt error.
0x18	*AUTOMATIC_CYCLE_ENGAGED*	Indicates the node has gone into automatic cycle mode.
0x19	*WRONG_LOAD_CONNECTED*	Indicates wrong load on node.
0x1A	*COLOUR_NOT_REACHABLE*	Indicates that node is unable to reach received colour code.
0x1B	*TARGET_NOT_REACHABLE*	Indicates the node is unable to reach received target position.
0x1C	*BAD_INDEX_RECEIVED*	Indicates io-protocol has received an invalid index.
0x1D	*COMMAND_OVERRULED*	Indicates that the command was overruled by a new command.
0x1E	*NODE_WAITING_FOR_POWER*	Indicates that the node reported waiting for power.
0xDF	*INFORMATION_CODE*	Indicates an unknown error code received. (Hex code is shown on display)
0xE0	*PARAMETER_LIMITED*	Indicates the parameter was limited by an unknown device. (Same as LIMITATION_BY_UNKNOWN_DEVICE)
0xE1	*LIMITATION_BY_LOCAL_USER*	Indicates the parameter was limited by local button.
0xE2	*LIMITATION_BY_USER*	Indicates the parameter was limited by a remote control.
0eE3	*LIMITATION_BY_RAIN*	Indicates the parameter was limited by a rain sensor.
0xE4	*LIMITATION_BY_TIMER*	Indicates the parameter was limited by a timer.
0xE6	*LIMITATION_BY_UPS*	Indicates the parameter was limited by a power supply.
0xE7	*LIMITATION_BY_UNKNOWN_DEVICE*	Indicates the parameter was limited by an unknown device. (Same as PARAMETER_LIMITED)
0xEA	*LIMITATION_BY_SAAC*	Indicates the parameter was limited by a standalone automatic controller.
0xEB	*LIMITATION_BY_WIND*	Indicates the parameter was limited by a wind sensor.
0xEC	*LIMITATION_BY_MYSELF*	Indicates the parameter was limited by the node itself.
0xED	*LIMITATION_BY_AUTOMATIC_CYCLE*	Indicates the parameter was limited by an automatic cycle.
0xEE	*LIMITATION_BY_EMERGENCY*	Indicates the parameter was limited by an emergency.

Table 177 - StatusReply parameter description.

10.1.3.8 InformationCode parameter

InformationCode is a 32-bit long integer. InformationCode contains the hexadecimal information code to show if system is unable to decode status.

10.1.4 GW_COMMAND_REMAINING_TIME_NTF

This command tells how long it takes until the actuator has reached the desired position.

Command	Data 1 - 2	Data 3	Data 4	Data 5 - 6
GW_COMMAND_REMAINING_TIME_NTF	SessionID	Index	NodeParameter	Seconds

Table 178 - GW_COMMAND_REMAINING_TIME_NTF frame format.

10.1.4.1 SessionID parameter

Unique identification of the session. Same value as SessionID in triggering GW_COMMAND_SEND_REQ frame.

10.1.4.2 Index parameter

The system table index for the current actuator.

10.1.4.3 NodeParameter parameter

Identifies the parameter remaining time is returned for.
See Table 175 for description of valid values.

10.1.4.4 Seconds parameter

Remaining time value in seconds.

10.1.5 GW_SESSION_FINISHED_NTF

GW_SESSION_FINISHED_NTF sent when the session started by GW_COMMAND_SEND_REQ, is over.

Command	Data 1 - 2
GW_SESSION_FINISHED_NTF	SessionID

Table 179 - GW_SESSION_FINISHED_NTF frame format.

10.1.5.1 SessionID parameter

Unique identification of the command. Same value as SessionID in triggering frame.

10.1.6 GW_COMMAND_SEND_REQ frame examples

Here are some examples shown to help using GW_COMMAND_SEND_REQ command.

GW_COMMAND_SEND_REQ example 1:
    This is probably the most default use of GW_COMMAND_SEND_REQ. Activation send to one actuator, as a result of the user pressing a button. Only Main Parameter (MP) value is set. No priority levels lock.
GW_COMMAND_SEND_REQ example 2:
    Two parameters are set on one actuator. If the actuator is an interior venetian blind, then MP is position of the blind and FP1 is orientation of the slats.
GW_COMMAND_SEND_REQ example 3:
    Same position set on two actuators by USER. To prevent "Stand Alone Automatic Controls" to set a new value, priority level 5 is locked for 20 min.
GW_COMMAND_SEND_REQ example 4:
    Position set on two actuators by "Stand Alone Automatic Controls".
GW_COMMAND_SEND_REQ example 5:
    Stop a product, by setting MP to Current position.

For more information about Standard Parameter definition read paragraph 12 at page 98.

Parameter name	Value	Description
Length	69	Frame length before SLIP
Command	0x0300	GW_COMMAND_SEND_REQ
SessionID	0x0001	Unique identification of the command.
CommandOriginator	1	Command originator type = USER
PriorityLevel	3	"User Level 2"
ParameterActive	0	Get info about Main Parameter (MP) in GW_COMMAND_RUN_STATUS_NTF
FPI1	0	No functional parameters value set.
FPI2	0	No functional parameters value set.
FunctionalParameterValue [0]	0x1234	MP must go to position 0x1234.
FunctionalParameterValue [1-16]	Don't care.	No functional parameters value set.
IndexArrayCount	1	Send command to one actuator.
IndexArray [0]	0	Activate node with node index 0.
IndexArray [1-19]	Don't care.	Last 19 node slot are not used.
PriorityLevelLock	0	Don't change priority level lock pool.
PLI_0_3	Don't care.	
PLI_4_7	Don't care.	
LockTime	Don't care.	
CRC	CRC	CRC

Table 180 - GW_COMMAND_SEND_REQ example 1.

Parameter name	Value	Description
Length	69	Frame length before SLIP
Command	0x0300	GW_COMMAND_SEND_REQ
SessionID	0x0002	Unique identification of the command.
CommandOriginator	1	Command originator type = USER
PriorityLevel	3	"User Level 2"
ParameterActive	0	Get info about Main Parameter (MP) in GW_COMMAND_RUN_STATUS_NTF
FPI1	0x80	FP1 is pointed out.
FPI2	0	No functional parameters value set.
FunctionalParameterValue [0]	0x1234	MP must go to position 0x1234.
FunctionalParameterValue [1]	0x5678	FP1 must go to position 0x5678.
FunctionalParameterValue [2-16]	Don't care.	FP2 to FP16 are not assigned a value.
IndexArrayCount	1	Send command to one actuator.
IndexArray [0]	1	Activate node with node index 1.
IndexArray [1-19]	Don't care.	Last 19 node slot are not used.
PriorityLevelLock	0	Don't change priority level lock pool.
PLI_0_3	Don't care.	
PLI_4_7	Don't care.	
LockTime	Don't care.	
CRC	CRC	CRC

Table 181 - GW_COMMAND_SEND_REQ example 2.

Parameter name	Value	Description
Length	69	Frame length before SLIP
Command	0x0300	GW_COMMAND_SEND_REQ
SessionID	0x0003	Unique identification of the command.
CommandOriginator	1	Command originator type = USER
PriorityLevel	3	"User Level 2"
ParameterActive	0	Get info about Main Parameter (MP) in GW_COMMAND_RUN_STATUS_NTF
FPI1	0	No functional parameters value set.
FPI2	0	No functional parameters value set.
FunctionalParameterValue [0]	0x1234	MP must go to position 0x1234.
FunctionalParameterValue [1-16]	Don't care.	No functional parameters value set.
IndexArrayCount	2	Send command to two actuators.
IndexArray [0]	2	Activate node with node index 2.
IndexArray [1]	7	Activate node with node index 7.
IndexArray [2-19]	Don't care.	Last 18 node slot are not used.

PriorityLevelLock	1	Make change on priority level lock pool.
PLI_0_3	0xFF	Keep current PL_0, PL_1, PL_2 and PL_3.
PLI_4_7	0xCF	Lock PL_5. Keep current PL_4, PL_6 and PL_7.
LockTime	39	Lock PL_5 in 20 min.
CRC	CRC	CRC

Table 182 - GW_COMMAND_SEND_REQ example 3.

Parameter name	Value	Description
Length	69	Frame length before SLIP
Command	0x0300	GW_COMMAND_SEND_REQ
SessionID	0x0004	Unique identification of the command.
CommandOriginator	8	Stand Alone Automatic Controls
PriorityLevel	5	"Comfort Level 2"
ParameterActive	0	Get info about Main Parameter (MP) in GW_COMMAND_RUN_STATUS_NTF
FPI1	0	No functional parameters value set.
FPI2	0	No functional parameters value set.
FunctionalParameterValue [0]	0x1234	MP must go to position 0x1234.
FunctionalParameterValue [1-16]	Don't care.	No functional parameters value set.
IndexArrayCount	2	Send command to two actuators.
IndexArray [0]	3	Activate node with node index 3.
IndexArray [1]	4	Activate node with node index 4.
IndexArray [2-19]	Don't care.	Last 18 node slot are not used.
PriorityLevelLock	0	Don't change priority level lock pool.
PLI_0_3	Don't care.	
PLI_4_7	Don't care.	
LockTime	Don't care.	
CRC	CRC	CRC

Table 183 - GW_COMMAND_SEND_REQ example 4.

Parameter name	Value	Description
Length	69	Frame length before SLIP
Command	0x0300	GW_COMMAND_SEND_REQ
SessionID	0x0005	Unique identification of the command.
CommandOriginator	1	Command originator type = USER
PriorityLevel	3	"User Level 2"
ParameterActive	0	Get info about Main Parameter (MP) in GW_COMMAND_RUN_STATUS_NTF
FPI1	0	No functional parameters value set.
FPI2	0	No functional parameters value set.
FunctionalParameterValue [0]	0xD200	MP must stay at current position.
FunctionalParameterValue [1-16]	Don't care.	No functional parameters value set.
IndexArrayCount	1	Send command to one actuator.
IndexArray [0]	0	Activate node with node index 0.
IndexArray [1-19]	Don't care.	Last 19 node slot are not used.
PriorityLevelLock	0	Don't change priority level lock pool.
PLI_0_3	Don't care.	
PLI_4_7	Don't care.	
LockTime	Don't care.	
CRC	CRC	CRC

Table 184 - GW_COMMAND_SEND_REQ example 5.

10.2 STOP

The GW_COMMAND_SEND_REQ command can be used to stop one or more actuators movement, by setting MP to CURRENT (0xD200). See example in Table 184 at page 66.
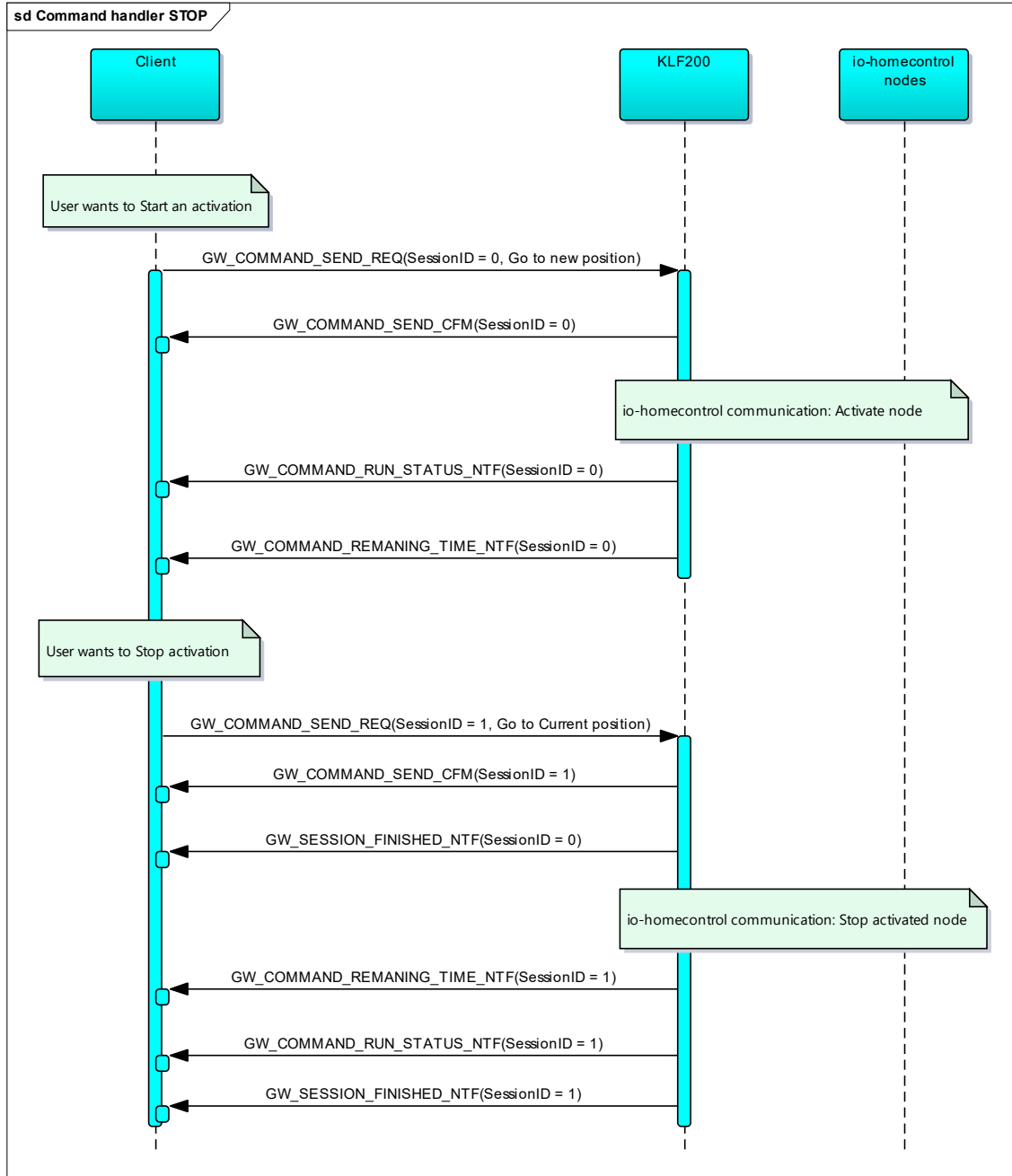
Figure 17 - Sequence diagram, Stop activated node.

10.3 Status request
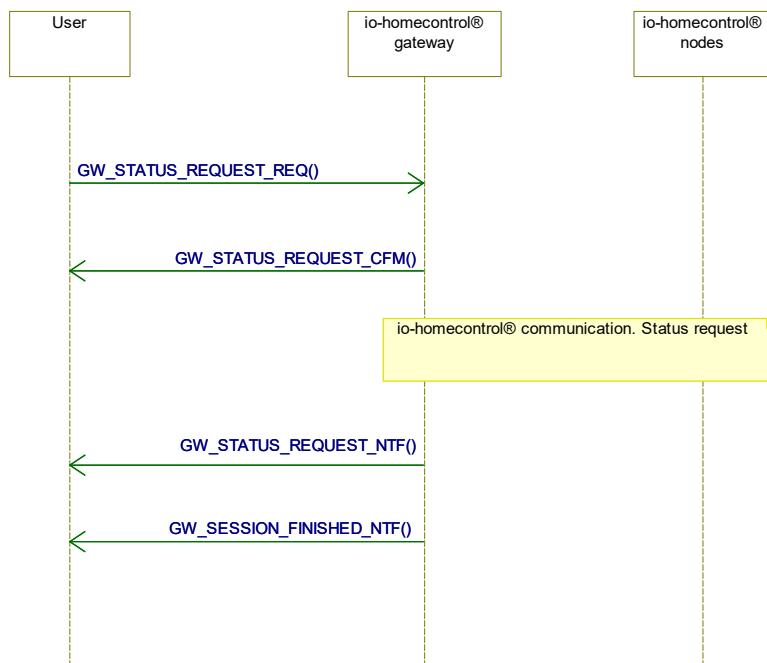
Figure 18 - Sequence diagram, Status request

10.3.1 GW_STATUS_REQUEST_REQ

Command	Data 1 – 2	Data 3	Data 4 – 23	Data 24
GW_STATUS_REQUEST_REQ	SessionID	IndexArrayCount	IndexArray	StatusType

Data 25	Data 26
FPI1	FPI2

Table 185 - GW_STATUS_REQUEST_REQ frame format.

10.3.1.1 SessionID parameter

Unique identification of the command. See paragraph 10.1.1.1 at page 57 for more information.

10.3.1.2 IndexArrayCount parameter

See paragraph 10.1.1.3 page 59 for description.

10.3.1.3 IndexArray parameter

See paragraph 10.1.1.8 page 59 for description.

10.3.1.4 StatusType parameter

Defines the status type to pack into the response GW_STATUS_REQUEST_NTF frame. See paragraph 10.3.3 for GW_STATUS_REQUEST_NTF description.

StatusType value	Description
0	Request Target position
1	Request Current position
2	Request Remaining time
3	Request Main info.

Other values	Not valid value.

Table 186 - StatusType parameter.

10.3.1.4.1   Target position

The gateway will reply with target position value for the main parameter. If FPI1 or FPI2 is different from 0x00, target position value will also be returned for one or more functional parameters. The maximum number of functional parameters is limited to seven for each request.

10.3.1.4.2   Current position

The gateway will reply with current position value for the main parameter. If FPI1 or FPI2 is different from 0x00, current position value will also be returned for one or more functional parameters. The maximum number of functional parameters is limited to seven for each request.

10.3.1.4.3   Remaining time

The gateway will reply with remaining time value in seconds for the main parameter. Remaining time is the actuators estimation of the time when execution is done. If FPI1 or FPI2 is different from 0x00, remaining time value will also be returned for one or more functional parameters. The maximum number of functional parameters is limited to seven for each request.

10.3.1.4.4   Main info.

The returned GW_STATUS_REQUEST_NTF will be packed with target and current position for main parameter, remaining time, last master execution address and last command originator.

10.3.1.5   FPI1 and FPI2 parameters

See paragraph 10.1.1.5 page 59 for description.

10.3.2 GW_STATUS_REQUEST_CFM

Command	Data 1 – 2	Data 3
GW_STATUS_REQUEST_CFM	SessionID	Status

Table 187 - GW_STATUS_REQUEST_CFM frame format.

10.3.2.1   Status parameter

Indicates if the command is accepted (true) or rejected (false) by the Command Handler.

10.3.2.2   SessionID parameter

Unique identification of the command. Same value as SessionID in triggering frame.

10.3.3 GW_STATUS_REQUEST_NTF

GW_STATUS_REQUEST_NTF have two forms, depends on the value of StatusType parameter. See Table 188 and Table 189.

Command	Data 1 – 2	Data 3	Data 4	Data 5	Data 6
GW_STATUS_REQUEST_NTF	SessionID	StatusID	NodeIndex	RunStatus	StatusReply

Data 7	Data 8	Data 9 - 59
StatusType = "Target Position" or StatusType = "Current Position" or StatusType = "Remaining Time"	StatusCount	ParameterData

Table 188 - GW_STATUS_REQUEST_NTF frame format, when StatusType = "Target Position" or "Current Position" or "Remaining Time".

Command	Data 1 – 2	Data 3	Data 4	Data 5	Data 6
GW_STATUS_REQUEST_NTF	SessionID	StatusID	NodeIndex	RunStatus	StatusReply

Data 7	Data 8 - 9	Data 10 - 11	Data 12 - 13
StatusType = "Main Info"	TargetPosition	CurrentPosition	RemainingTime

Data 14 - 17	Data 18
LastMasterExecutionAddress	LastCommandOriginator

Table 189 - GW_STATUS_REQUEST_NTF frame format, when StatusType = "Main Info".

10.3.3.1   SessionID parameter

SessionID is a 16 bit integer. SessionID is used to identify the status request. Same value as SessionID in triggering frame.

10.3.3.2   bStatusID parameter

Identification of the status owner.

StatusID value	Tag	Description
0	STATUS_LOCAL_USER	The status is from a local user activation. (My self)
1	STATUS_USER	The status is from a user activation.
2	STATUS_RAIN	The status is from a rain sensor activation.
3	STATUS_TIMER	The status is from a timer generated action.
5	STATUS_UPS	The status is from a UPS generated action.
8	STATUS_PROGRAM	The status is from an automatic program generated action. (SAAC)
9	STATUS_WIND	The status is from a Wind sensor generated action.
10	STATUS_MYSELF	The status is from an actuator generated action.
11	STATUS_AUTOMATIC_CYCLE	The status is from a automatic cycle generated action.
12	STATUS_EMERGENCY	The status is from an emergency or a security generated action.
13	STATUS_UNKNOWN	The status is from an unknown command originator action.

Table 190 - StatusID parameter description.

10.3.3.3   NodeIndex parameter

Index of the node.

10.3.3.4   RunStatus parameter

Contains the running status of the node.
See paragraph 10.1.3.6 page 62.

10.3.3.5   StatusReply parameter

Contains current state of the node. (Error code)
See paragraph 10.1.3.7 page 62.

10.3.3.6   StatusType parameter

Contains the type of status requested.
See paragraph 10.3.1.4 page 68.

10.3.3.7   StatusCount parameter

Contains the number entries in stParameterData[] array, there are in use.

10.3.3.8   ParameterData parameter

ParameterData is an array that holds 1 to 17 entries. The array is fixed size. Data is packed in the start of the array. If it for example holds data about Main parameter (MP)

and Functional parameter 2 (FP2), then ParameterData [0] = MP and ParameterData [1] = FP2.

Each ParameterData entry holds the parameters NodeParameter and ParameterValue. See Table 191, paragraph 10.3.3.8.1 and paragraph 10.3.3.8.2.

Data 1	Data 2 – 3
NodeParameter	ParameterValue

Table 191 - ParameterData entry format.

10.3.3.8.1 NodeParameter parameter

Identifies the functional parameter. See Table 175 at page 62.

10.3.3.8.2 ParameterValue parameter

Status request value for the parameter. Holds Target position, Current position or Remaining time. ParameterValue is an unsigned 16 bit integer.

10.3.3.9 TargetPosition parameter

Status request Target position value. TargetPosition is an unsigned 16 bit integer.

10.3.3.10 CurrentPosition parameter

Status request Current position value. CurrentPosition is an unsigned 16 bit integer.

10.3.3.11 RemainingTime parameter

This is an estimation of the time needed for the actuator to reach its target position. RemainingTime is a 16 bit integer. Resolution = 1 second, Range = 0 to 65533 seconds (18 hours), 0xFFFE = More than 65533 seconds. 0xFFFF = undefined.

10.3.3.12 LastMasterExecutionAddress parameter

Address of the Master that has executed the last command.

Last Master execution address is updated when an activating command is received and a successful acknowledges is send.

LastMasterExecutionAddress is an unsigned 32 bit integer, holding a value in the interval from 0x00000000 to 0x00FFFFFF.

10.3.3.13 LastCommandOriginator parameter

Command Originator information about the Master that has executed the last command. Last Command Originator is updated when an activating command is received and a successful acknowledges is send. See more about CommandOriginator in paragraph 10.1.1.2 at page 57.

10.3.4 GW_SESSION_FINISHED_NTF

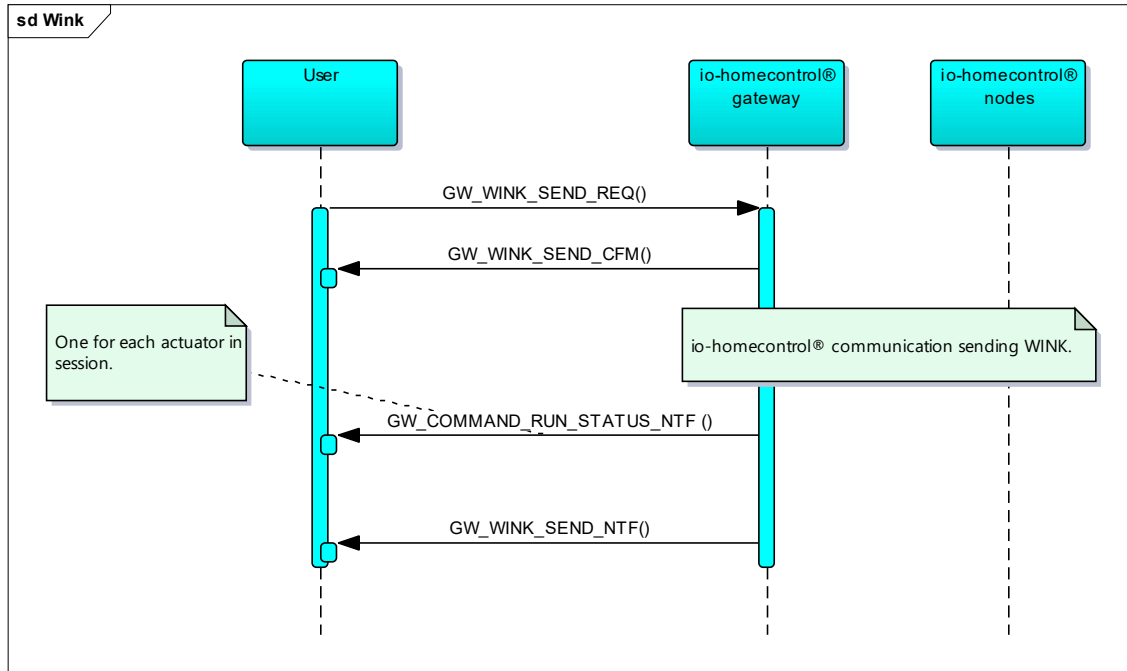See description in paragraph 10.1.5 page 64.

10.4 WINK

Figure 19 - Sequence diagram for send WINK command.

10.4.1 GW_WINK_SEND_REQ

In order to identify a specific node, the gateway can send a wink request to the node.

Command	Data 1 – 2	Data 3	Data 4	Data 5	Data 6
GW_WINK_SEND_REQ	SessionID	CommandOriginator	PriorityLevel	WinkState	WinkTime

Data 7	Data 8 – 27
IndexArrayCount	IndexArray

Table 192 - GW_WINK_SEND_REQ frame format.

10.4.1.1   SessionID parameter

Unique identification of the command. See paragraph 10.1.1.1 at page 57 for more information.

10.4.1.2   CommandOriginator parameter

See paragraph 10.1.1.2 page 57 for description.

10.4.1.3   PriorityLevel parameter

See paragraph 10.1.1.3 page 57 for description.

10.4.1.4   WinkState parameter

Contains the state of the wink command.

WinkState value	Description
0	Disable wink
1	Enable wink

Table 193 - WinkState parameter description.

10.4.1.5 WinkTime parameter

Contains the time for the wink command.

WinkTime value	Description
0	Stop wink.
1	Wink in 1 sec.
2	Wink in 2 sec.
:	:
253	Wink in 253 sec.
254	Manufacturer specific wink time. Can be different from actuator to actuator.
255	Wink forever.

Table 194 - bWinkTime parameter description.

10.4.1.6 IndexArrayCount parameter

See paragraph 10.1.1.3 page 59 for description.

10.4.1.7 IndexArray parameter

See paragraph 10.1.1.8 page 59 for description.

10.4.2 GW_WINK_SEND_CFM

Command	Data 1 – 2	Data 3
GW_WINK_SEND_CFM	SessionID	Status

Table 195 - GW_WINK_SEND_CFM frame format.

10.4.2.1 Status parameter

Status value	Description
0	Wink command is rejected.
1	Wink command is accepted.

Table 196 - Status parameter description.

10.4.3 GW_COMMAND_RUN_STATUS_NTF

See paragraph 10.1.3 at page 61 for a detailed description.

10.4.4 GW_WINK_SEND_NTF

Command	Data 1 – 2
GW_WINK_SEND_NTF	SessionID

Table 197 - GW_WINK_SEND_NTF frame format.

10.5 Limitation

An actuator node can be limited on its movement area, for a given period. The limitation commands let you set or read a limitation.
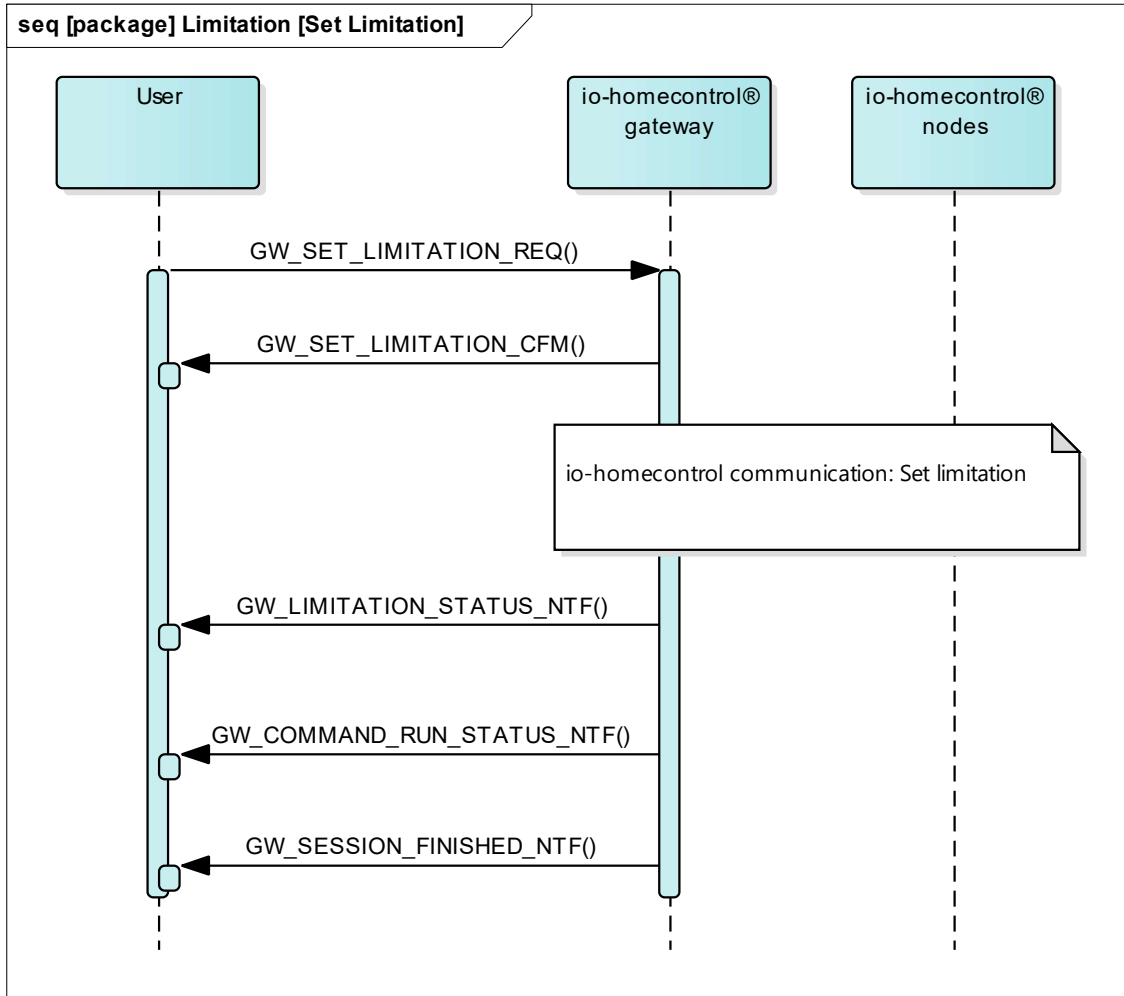
10.5.1 Set limitation

Figure 20 - Sequence diagram, Set limitation. Note: GW_LIMITATION_STATUS_NTF is only send if the limitation is set successfully.

10.5.2 GW_SET_LIMITATION_REQ

Command	Data 1 - 2	Data 3	Data 4	Data 5
GW_SET_LIMITATION_REQ	SessionID	CommandOriginator	PriorityLevel	IndexArrayCount

Data 6 – 25	Data 26	Data 27 - 28	Data 29 - 30	Data 31
IndexArray[20]	ParameterID	LimitationValueMin	LimitationValueMax	LimitationTime

Table 198 - GW_SET_LIMITATION_REQ frame format.

10.5.2.1   SessionID parameter

Unique identification of the command. See paragraph 10.1.1.1 at page 57 for more information.

10.5.2.2   CommandOriginator parameter

See paragraph 10.1.1.2 page 57 for description.

10.5.2.3 PriorityLevel parameter

See paragraph 10.1.1.3 page 57 for description.

10.5.2.4 IndexArrayCount parameter

See paragraph 10.1.1.3 page 59 for description.

10.5.2.5 IndexArray parameter

See paragraph 10.1.1.8 page 59 for description.

10.5.2.6 ParameterID parameter

ParameterID identify which parameter to operate.
Normally ParameterID is set to MP. Deviations from this rule are Internal Venetian blinds, Exterior Venetian blind and Louver blind where the slat angle is set by FP3.

ParameterID value	Tag	Description
0	MP	Main Parameter.
1	FP1	Functional Parameter number 1.
2	FP2	Functional Parameter number 2.
3	FP3	Functional Parameter number 3.
4	FP4	Functional Parameter number 4.
5	FP5	Functional Parameter number 5.
6	FP6	Functional Parameter number 6.
7	FP7	Functional Parameter number 7.
8	FP8	Functional Parameter number 8.
9	FP9	Functional Parameter number 9.
10	FP10	Functional Parameter number 10.
11	FP11	Functional Parameter number 11.
12	FP12	Functional Parameter number 12.
13	FP13	Functional Parameter number 13.
14	FP14	Functional Parameter number 14.
15	FP15	Functional Parameter number 15.
16	FP16	Functional Parameter number 16.
Other values		Not allowed.

Table 199 - ParameterID parameter description

The use of Functional Parameters in different Actuator types is further described in paragraph 14 at page 104.

10.5.2.7 LimitationValueMin parameter

The minimum limitation value for the parameter set in 'ParameterID' parameter
The following access methods can be used:
- Relative
- Target
- Current
- Default
- Ignore

See paragraph 12 at page 12 for detailed description on access methods.

10.5.2.8 LimitationValueMax parameter

The maximum limitation value for the parameter set in 'ParameterID' parameter
The following access methods can be used:
- Relative
- Target
- Current
- Default
- Ignore

See paragraph 12 at page 12 for detailed description on access methods.

10.5.2.9 LimitationTime parameter

The limitation timer specifies the time for the limitation send to be active. At timeout, the actual pool entry will be deleted (not active anymore). The timer can hold the following values.

Limitation timer
0 = 30 seconds
1 = 60 seconds
..
252 = 7590 seconds (126 min 30 sec)
253 = unlimited
254 = clear entry for the Master
255 = clear all

Table 200 – LimitationTime parameter decription.

If the limitation timer is set to 254, the limitation settings for this Master will be deleted if any.
If limitation timer is set to 255, all the entries will be deleted.

10.5.3 GW_SET_LIMITATION_CFM

Command	Data 1 - 2	Data 3
ST_GW_SET_LIMITATION_CFM	SessionID	Status

Table 201 - ST_GW_SET_LIMITATION_CFM frame format.

10.5.3.1 SessionID parameter

Unique identification of the command. Same value as SessionID in triggering frame.

10.5.3.2 Status parameter

Status value	Description
0	Set Limitation request command is rejected.
1	Set Limitation request command is accepted.

Table 202 - Status parameter description.

10.5.4 GW_LIMITATION_STATUS_NTF

GW_LIMITATION_STATUS_NTF is only send if the limitation is set successfully.
GW_COMMAND_RUN_STATUS_NTF will tell the resent if not succeeded.

Command	Data 1 - 2	Data 3	Data 4	Data 5-6	Data 7-8
GW_LIMITATION_STATUS_NTF	SessionID	NodeID	ParameterID	MinValue	MaxValue

Data 9	Data 10
LimitationOriginator	LimitationTime

Table 203 - ST_GW_SET_LIMITATION_CFM frame format.

10.5.4.1 SessionID parameter

Unique identification of the command. Same value as SessionID in triggering frame.

10.5.4.2 NodeID parameter

See paragraph 8.3.1.1 page 41 for further description.

10.5.4.3 ParameterID parameter

See paragraph 10.5.2.6 at page 75 for description.

10.5.4.4   MinValue parameter

Minimum allowed value for current parameter.

10.5.4.5   MaxValue parameter

Maximum allowed value for current parameter.

10.5.4.6   LimitationOriginator parameter

Command originator for current limitation. See paragraph 10.1.1.2 page 57 for description.

10.5.4.7   LimitationTime parameter

Remaining limitation time. See Table 200 how to interpret the value

10.5.5 GW_COMMAND_RUN_STATUS_NTF

See paragraph 10.1.3 at page 61 for a detailed description.

10.5.6 GW_SESSION_FINISHED_NTF

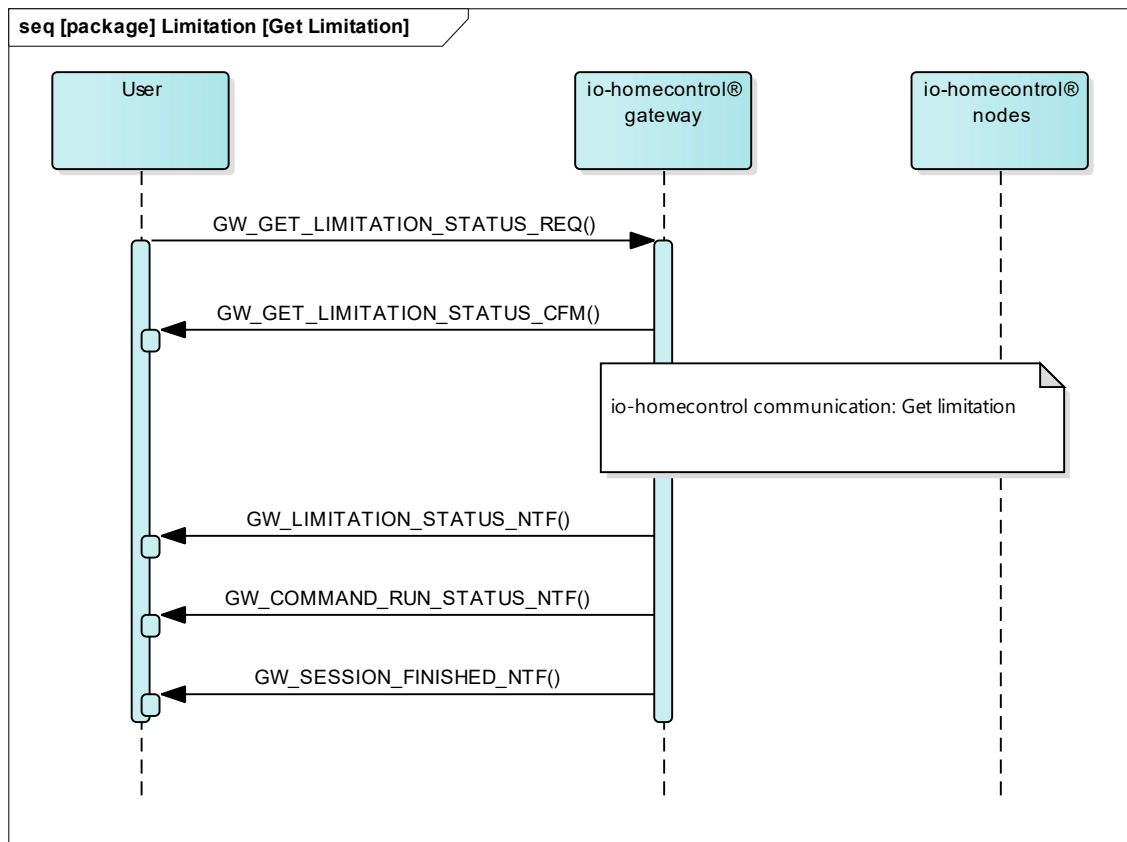See description in paragraph 10.1.5 page 64.

10.5.7 Get limitation

Figure 21 - Sequence diagram , Get limitation.

10.5.8 GW_GET_LIMITATION_STATUS_REQ

Command	Data 1 - 2	Data 3	Data 4	Data 5
GW_GET_LIMITATION_STATUS_REQ	SessionID	CommandOriginator	PriorityLevel	IndexArrayCount

Data 6 – 25	Data 26	Data 27
IndexArray[20]	ParameterID	LimitationType

Table 204 - GW_GET_LIMITATION_STATUS_REQ frame format.

10.5.8.1   SessionID parameter

Unique identification of the command. See paragraph 10.1.1.1 at page 57 for more information.

10.5.8.2   CommandOriginator parameter

See paragraph 10.1.1.3 page 57 for description.

10.5.8.3   PriorityLevel parameter

See paragraph 10.1.1.3 page 57 for description.

10.5.8.4   IndexArrayCount parameter

See paragraph 10.1.1.3 page 59 for description.

10.5.8.5   IndexArray parameter

See paragraph 10.1.1.8 page 59 for description.

10.5.8.6   ParameterID parameter

See paragraph 10.5.2.6 at page 75 for description.

10.5.8.7   LimitationType parameter

LimitationType value	Description
0	Resulting minimum limitation.
1	Resulting maximum limitation.

Table 205 - LimitationType parameter description.

10.5.9 GW_GET_LIMITATION_STATUS_CFM

Command	Data 1 - 2	Data 3
GW_GET_LIMITATION_STATUS_CFM	SessionID	Status

Table 206 - GW_GET_LIMITATION_STATUS_CFM frame format.

10.5.9.1   SessionID parameter

Unique identification of the command. See paragraph 10.1.1.1 at page 57 for more information.

10.5.9.2   Status parameter

Status value	Description
0	Get Limitation request command is rejected.
1	Get Limitation request command is accepted.

Table 207 - Status parameter description.

10.5.10 GW_LIMITATION_STATUS_NTF

See paragraph 10.5.4 at page 76 for a detailed description.

10.5.11 GW_COMMAND_RUN_STATUS_NTF

See paragraph 10.1.3 at page 61 for a detailed description.

10.5.12 GW_SESSION_FINISHED_NTF

See description in paragraph 10.1.5 page 64.

10.6 Mode

All actuators must accept Mode 0.
When a Mode 0 is received by the actuator, it's typically is functionality as follows:
1. If the actuator is moving it will stop (Main parameter set to current).
2. If the actuator is not moving, then the actuator normally runs to the maximum opposite direction (Main parameter set to 0% or 100%) from last time it was activated, by any access method.

10.6.1 GW_MODE_SEND_REQ

When the gateway receives a GW_MODE_SEND_REQ frame, it will send a mode activation to one or more actuators.

Command	Data 1 – 2	Data 3	Data 4	Data 5	Data 6
GW_MODE_SEND_REQ	SessionID	CommandOriginator	PriorityLevel	ModeNumber	ModeParameter

Data 7	Data 8-27	Data 28	Data 29	Data 30	Data 31
IndexArrayCount	IndexArray	PriorityLevelLock	PL_0_3	PL_4_7	LockTime

Table 208 - GW_MODE_SEND_REQ frame format.

SessionID, COmmandOriginator, PriorityLevel, IndexArrayCount, IndexArray, PriorityLevelLock, PL-0_3, PL_4_7 & LockTime parameters are the same as described in GW_COMMAND_SEND_REQ section.

10.6.1.1 ModeNumber parameter

ModeNumber is the Mode type, supported by actuator.
All actuators must accept ModeNumber = 0.

10.6.1.2 ModeParameter parameter

ModeParameter is the parameter for mode type, supported by actuator. The combination of ModeNumber and ModeParameter points out the specific mode to be run in actuator.
All actuators must accept ModeParameter = 0.

10.6.2 GW_MODE_SEND_CFM

The gateway will acknowledge with a GW_MODE_SEND_CFM after receiving a GW_MODE_SEND_REQ frame.

Command	Data 1 – 2	Data 3
GW_MODE_SEND_CFM	SessionID	Status

Table 209 - GW_MODE_SEND_CFM frame format.

10.6.2.1 Status parameter

Indicates if the GW_MODE_SEND_REQ command is accepted by the Command Handler or rejected with some err

Status value	Description
0	OK. Accepted by Command Handler
1	Failed. Rejected by Command Handler
2	Failed with unknown Client ID
3	Failed. Session ID already in use
4	Failed. Busy – no free session slots – try again
5	Failed. Illegal parameter value
255	Failed. Not further defined error

Other values	Not defined

Table 210 - Status parameter description.

10.6.2.2 SessionID parameter

Unique identification of the session. Same SessionID as used in
GW_MODE_SEND_REQ frame.

10.6.3 GW_COMMAND_RUN_STATUS_NTF

See paragraph 10.1.3 for a detailed description.

10.6.4 GW_COMMAND_REMAINING_TIME_NTF

See paragraph 10.1.4 for a detailed description.

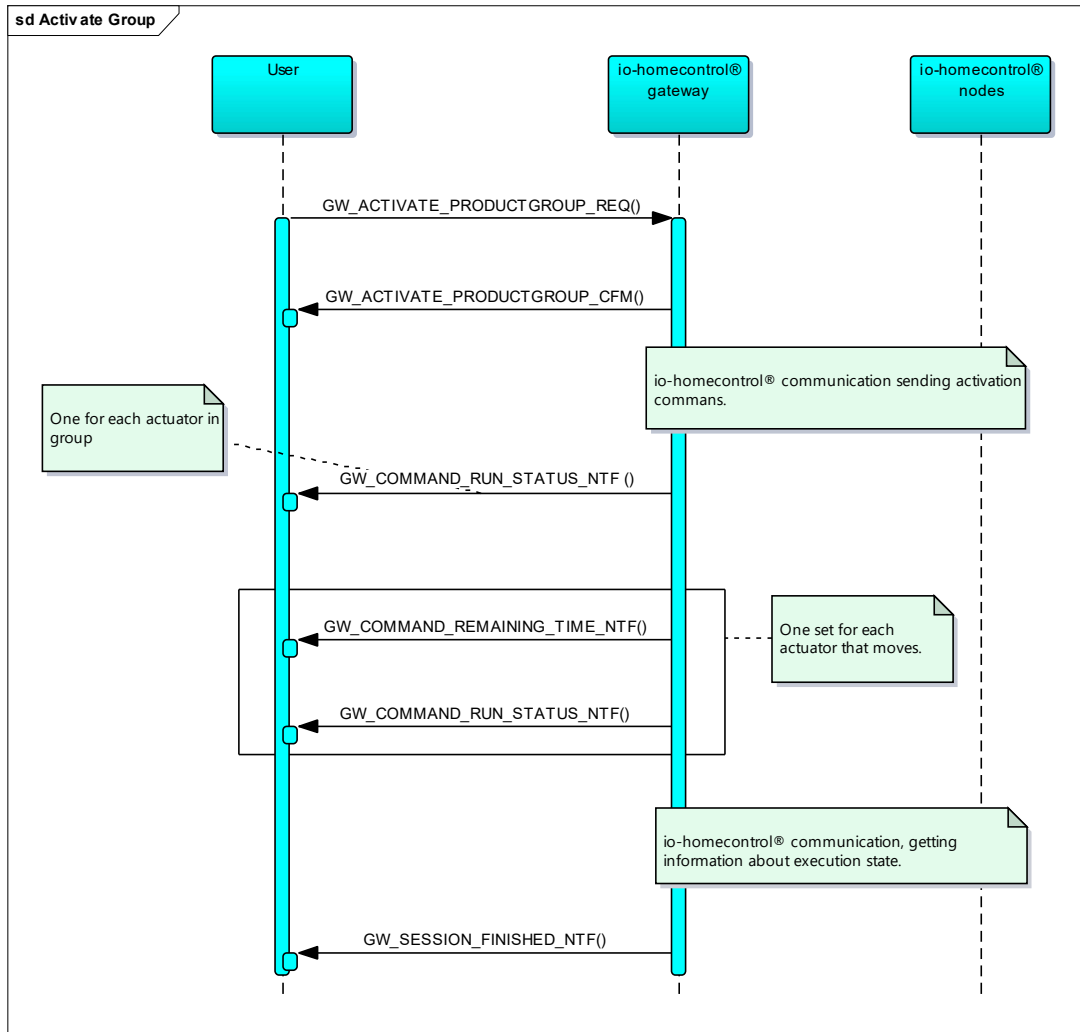### 10.6.5 GW_SESSION_FINISHED_NTF
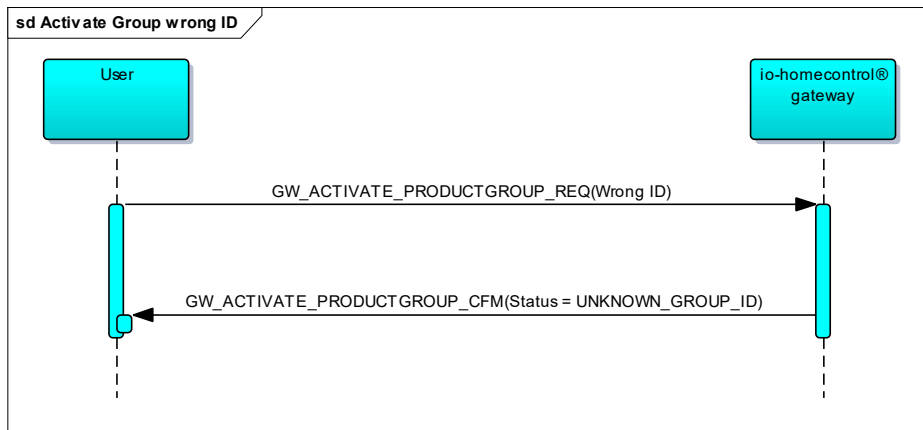
See paragraph 10.1.5 for a detailed description.

10.7 Product Group Activation

The gateway can handle activation of all actuators in user defined product group. Groups

Figure 22 – Activate group sequence diagram.

Figure 23 – Activate group with wrong ID sequence diagram.

10.7.1 GW_ACTIVATE_PRODUCTGROUP_REQ

Command	Data 1 - 2	Data 3	Data 4	Data 5

GW_ACTIVATE_PRODUCTGROUP_REQ	SessionID	CommandOriginator	PriorityLevel	ProductGroupID

Date 6	Data 7 - 8	Data 9	Data 10	Data 11	Data 12	Data 13
ParameterID	Position	Velocity	PriorityLevelLock	PL_0_3	PL_4_7	LockTime

Table 211 - GW_ACTIVATE_PRODUCTGROUP_REQ frame format.

10.7.1.1 SessionID parmeter

SessionID is at 16 bit unsigned integer. SessionID is used to give unique identifications of the command. SessionID value in GW_ACTIVATE_PRODUCTGROUP_REQ will be returned in all GW_ACTIVATE_PRODUCTGROUP_CFM and GW_ACTIVATE_PRODUCTGROUP_NTF belonging the same session. Make sure that you change SessionID for each session. Typical SessionID is incremented by one.

10.7.1.2 CommandOriginator parameter

See paragraph 10.1.1.2 page 57 for description.

10.7.1.3 PriorityLevel parameter

See paragraph 10.1.1.3 page 57 for description.

10.7.1.4 ProductGroupID parameter

Identification number of product group to be activated.

10.7.1.5 ParameterID parameter

See paragraph 10.5.2.6 at page 75 for description.

10.7.1.6 Position parameter

The position parameter are used to set actuators position for the parameter given by ParameterID. For more information read paragraph 12 'Appendix 1: Standard Parameter definition'.

10.7.1.7   Velocity parameter

Velocity value	Tag	Description
0	*DEFAULT*	The product group operates by its default velocity.
1	*SILENT*	The product group operates in silent mode (slow)*.
2	*FAST*	The product group operates with fast velocity*.
3-255	-	Not defined value.

Table 212 - Velocity parameter description.

*) Some old actuators will only move at default velocity, even if speed parameter are set to SILENT or FAST.

10.7.1.8   PriorityLevelLock parameter

PriorityLevelLock tells whether to use priority lock.

PriorityLevelLock value	Description
0	Do not set a new lock on priority level. Information in the parameters PL_0_3, PL_4_7 and LockTime are not used. This is the one typically used.
1	Information in the parameters PL_0_3, PL_4_7 and LockTime are used to lock one or more priority level.

Table 213 - PriorityLevelLock parameter

10.7.1.9   PL_0_3 and PL_4_7 parmeters
Priority level information
There are eight priority levels. Each priority level can set to one of four values. Those four values are listed in Table 168.

All priority Levels can be individually disabled or enabled with the information carried by the PLI bits, except PL0 that instantly will disable all priority levels to ensure human protection.

PLI Number	Name	Description
0	Disable priority	Disable the priority related to the Master
1	Enable	Enable the priority related to the Master
2	Enable all	Enable all pool entry for the specified priority level Must be used with caution!
3	Keep current	Do not make any action. When used, the priority setting for the specific level will be kept in its current state.

Table 214 - Priority Level Information numbers.

The PLI bits for each priority level are send in the frame as a trailer to the parameters within a parameter management command. To indicate that the frame is carrying the priority level lock information bytes, the PriorityLevelLock value must be set to 1.

Priority Level Lock Information Bytes
These bytes carry the Priority level information on each on the priority levels to manage.

PLI_0_3	PLI_4_7
Bit 7-6 = PLI 0	Bit 7-6 = PLI 4
Bit 5-4 = PLI 1	Bit 5-4 = PLI 5
Bit 3-2 = PLI 2	Bit 3-2 = PLI 6
Bit 1-0 = PLI 3	Bit 1-0 = PLI 7

Table 215 - Priority level lock bytes.

10.7.1.10   LockTime parmeter

LockTime defines a common lock time for all priority levels.

LockTime value	Description
0	30 seconds
1	60 seconds
:	:
254	7650 seconds (127 min 30 sec)
255	Unlimited time

Table 216 - LockTime parameter description.

10.7.2 GW_ACTIVATE_PRODUCTGROUP_CFM

Command	Data 1 - 2	Data 3
GW_ACTIVATE_PRODUCTGROUP_CFM	SessionID	Status

Table 217 - GW_ACTIVATE_PRODUCTGROUP_CFM frame format.

10.7.2.1 SessionID parmeter

Unique identification of the command. Same value as SessionID in triggering frame.

Read more about GW_COMMAND_RUN_STATUS_NTF in paragraph 10.1.3 page 61.
Read more about GW_COMMAND_REMAINING_TIME_NTF in paragraph10.1.4 page 64.
Read more about GW_SESSION_FINISHED_NTF in paragraph 10.1.5 page 64.

10.7.2.2 Status parameter

Status value	Description
0	Request accepted.
1	Unknown ProductGroupID.
2	SessionID already in use.
3	Busy, all activation slot in use. Try again later.
4	Wrong group type.
5	Not further defined error.
6	Invalid parameter used.
7-255	Reserved.

Table 218 - Status parameter description.

11 Scenes

A Scene is a group of positions of one or more nodes. The KLF200 gateway can hold up to 32 scenes, holding up to 192 node positions. For example, one scene with 192 node positions or 32 scenes each holding 6 node positions.

The gateway user has the following interface to scene handler:

- Define a scene.
- Delete a scene.
- Request the list of scenes.
- Activate a scene.

While KLF200 handle a scene command, attempts to start configurations service or command handler commands will be acknowledge with a GW_ERROR_NTF(BUSY) command.

If the gateway has not the required room for a scene or node slot to define a new scene, the user will then receive an GW_INITIALIZE_SCENE_CFM(Error - Can't store more scenes) or GW_RECORD_SCENE_NTF(Error - Can't store more nodes. Scene not created) frame to closing the define scene session, see Figure 25 and Figure 26.

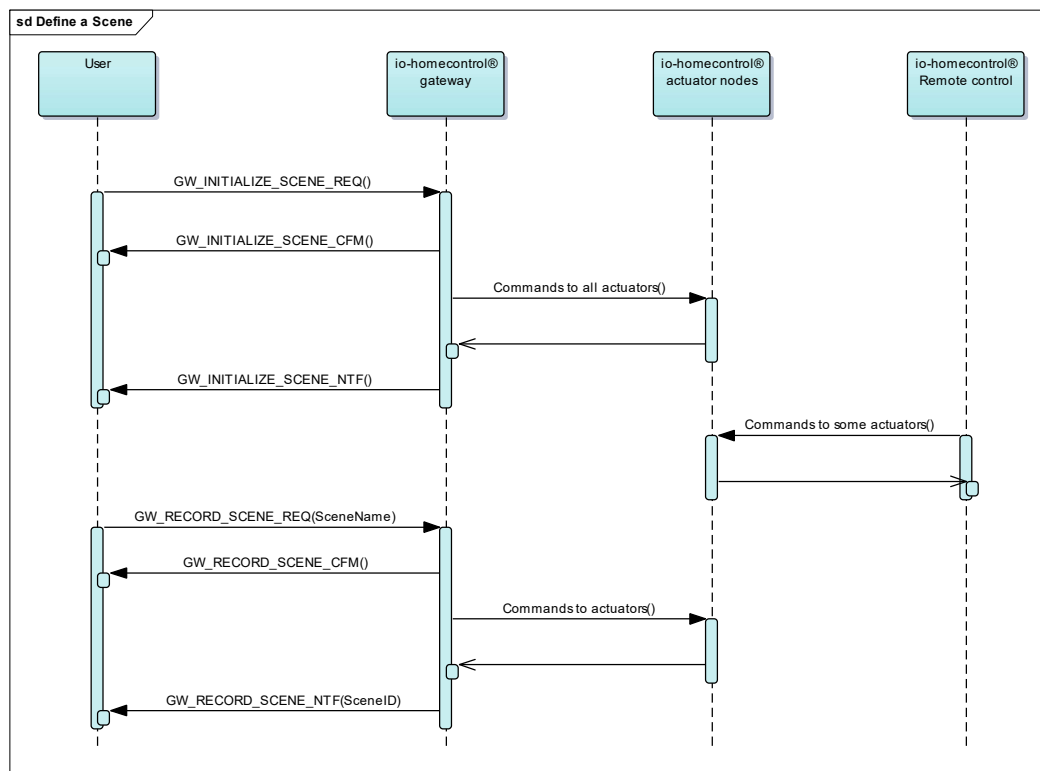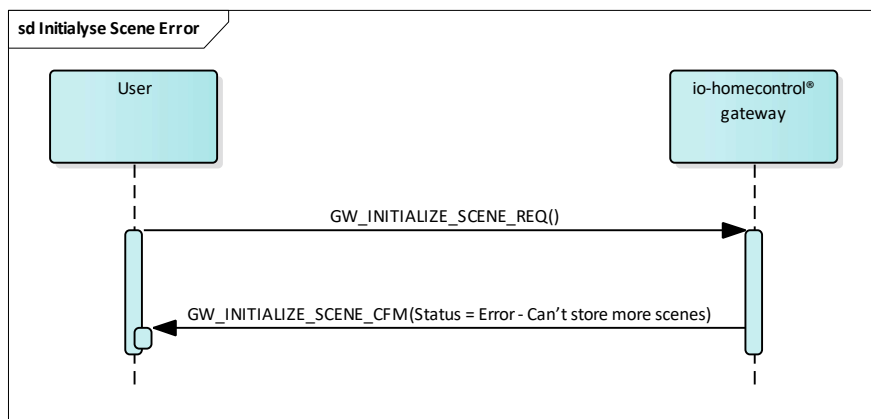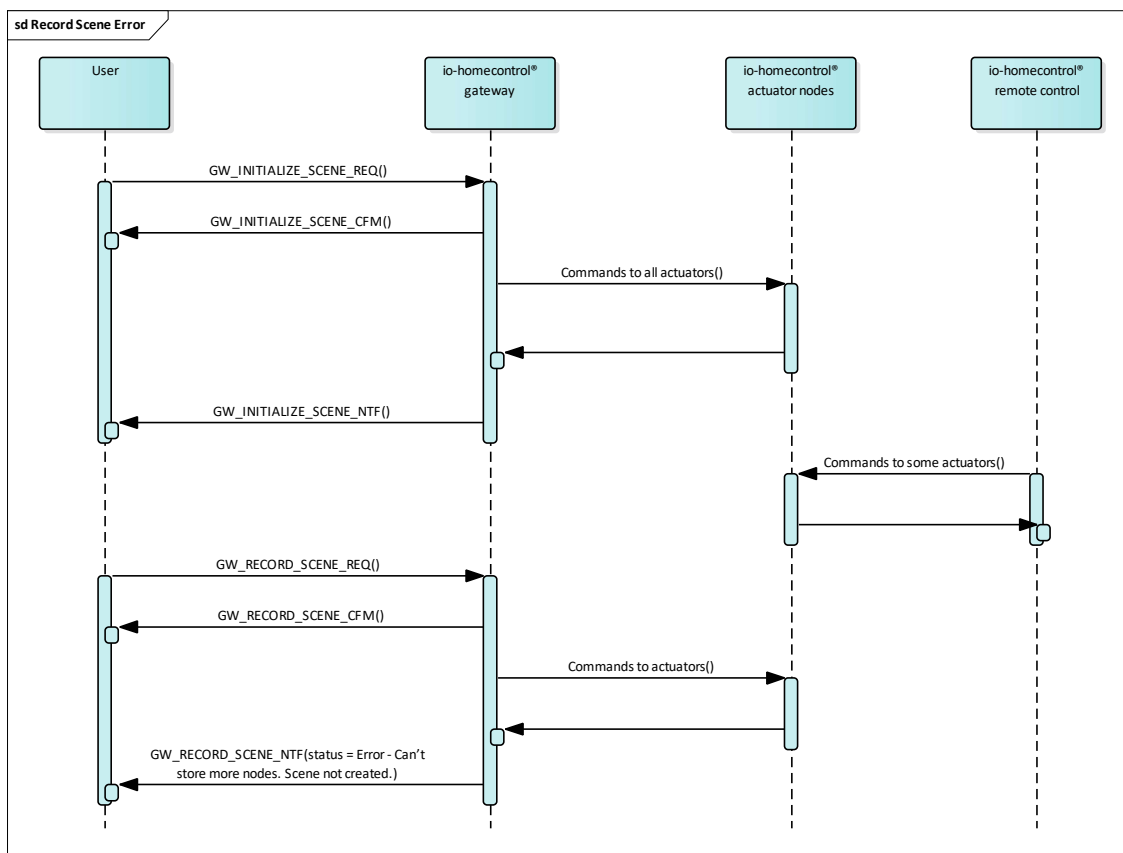You can read more about the GW_ERROR_NTF frame in paragraph 10.6.1.1 page 79.

Figure 24 - Sequence diagram show how a scene is defined.

Figure 25 - Sequence diagram show when out of memory for scene slot.

Figure 26 - Sequence diagram show when out of memory for node slot.

11.1 Define a new scene

As shown at Figure 24, defining a new scene in KLF200 takes place in three phases.

1. Prepare KLF200 and io-homecontrol® nodes.
2. Set io-homecontrol® nodes to desired position.
3. Store scene in KLF200 with a text label and a scene identification number.

11.1.1 Prepare Gateway and io-homecontrol® nodes

To prepare a new scene, a GW_INITIALIZE_SCENE_REQ frame is sent to gateway, who will return with a GW_INITIALIZE_SCENE_CFM frame and a GW_INITIALIZE_SCENE_NTF frame when scene initialization is done.

11.1.2 GW_INITIALIZE_SCENE_REQ

Command
GW_INITIALIZE_SCENE_REQ

Table 219 - GW_INITIALIZE_SCENE_REQ frame format.

11.1.3 GW_INITIALIZE_SCENE_CFM

Command	Data 1
GW_INITIALIZE_SCENE_CFM	Status

Table 220 - GW_INITIALIZE_SCENE_CFM frame format.

11.1.3.1 Status

Status value	Description
0	OK - Request accepted
1	Error - System table is empty
2	Error - Can't store more scenes
Other values	Reserved

Table 221 - Status parameter description.

11.1.4 GW_INITIALIZE_SCENE_NTF

Command	Data 1	Data 2 - 26
GW_INITIALIZE_SCENE_NTF	Status	NodeState

Table 222 - GW_INITIALIZE_SCENE_NTF frame format.

11.1.4.1 Status

Status value	Description
0	OK - Request successful
1	Partly OK – Some nodes not initialized
2	Error – No nodes initialized
Other values	Reserved

Table 223 – Status parameter description.

11.1.4.2 NodeState

Bit array indicating the initialize state of a node.

0 = Actuator initialized successful
1 = Actuator initialization failed

11.2 Initialize scene Cancel command set

After GW_INITIALIZE_SCENE_REQ, KLF200 is hold in a state, waiting for GW_RECORD_SCENE_REQ. If user wants to cancel new scene define, the user can send GW_INITIALIZE_SCENE_CANCEL_REQ.

11.2.1 GW_INITIALIZE_SCENE_CANCEL_REQ

Command
GW_INITIALIZE_SCENE_CANCEL_REQ

Table 224 - GW_INITIALIZE_SCENE_CANCEL_REQ frame format.

11.2.2 GW_INITIALIZE_SCENE_CANCEL_CFM

GW_INITIALIZE_SCENE_CANCEL_CFM is acknowledging to
GW_INITIALIZE_SCENE_CANCEL_REQ.

Command	Data 1
GW_INITIALIZE_SCENE_CANCEL_CFM	Status

Table 225 - GW_INITIALIZE_SCENE_CANCEL_CFM frame format.

11.2.2.1   Status

Status value	Description
0	OK – Request accepted
1	Error – GW_INITIALIZE_SCENE has not been performed
Other values	Reserved

Table 226 – Status parameter description.

11.3 Set io-homecontrol® nodes to desired position.

Use an io-homecontrol® remote control to set the desired position of involved io-homecontrol® actuator nodes.

11.4 Store scene in Gateway with a text label and a scene identification number.

The scene is stored with a text label and an identification number.
The scene will be stored when Gateway receive a GW_RECORD_SCENE_REQ frame with the text label. Gateway will select a unique scene identification number and return it with a GW_RECORD_SCENE_CFM frame and a GW_RECORD_SCENE_NTF frame when scene is recorded.

11.4.1 GW_RECORD_SCENE_REQ

Command	Data 1 - 64
GW_RECORD_SCENE_REQ	SceneName

Table 227 - GW_RECORD_SCENE_REQ frame format.

11.4.1.1   Status

Status value	Description
0	OK – Request accepted
1	Error – GW_INITIALIZE_SCENE has not been performed
Other values	Reserved

Table 228 – Status parameter description.

11.4.2 GW_RECORD_SCENE_CFM

Command	Data 1
GW_RECORD_SCENE_CFM	Status

Table 229 - GW_RECORD_SCENE_CFM frame format.

11.4.3 GW_RECORD_SCENE_NTF

Command	Data 1	Data 2
GW_RECORD_SCENE_NTF	Status	SceneID

Table 230 - GW_RECORD_SCENE_NTF frame format.

11.4.3.1 Status

Status value	Description
0	OK – Request successful
1	Error – Request failed
2	Error – No io-homecontrol® products has been stimulated
3	Error - Can't store more nodes. Scene not created.
Other values	Reserved

**Table 231 – Status parameter description.**

11.4.3.2 SceneID

The parameter SceneID is the one byte long unique number.

If Status indicates success, the parameter SceneID contain obtained scene ID.
If Status indicates failure, the parameter SceneID contain 0xFF and should not be used.

11.5 Delete a scene

When deleting a scene in KLF200, the remaining scenes will not change their identification number.
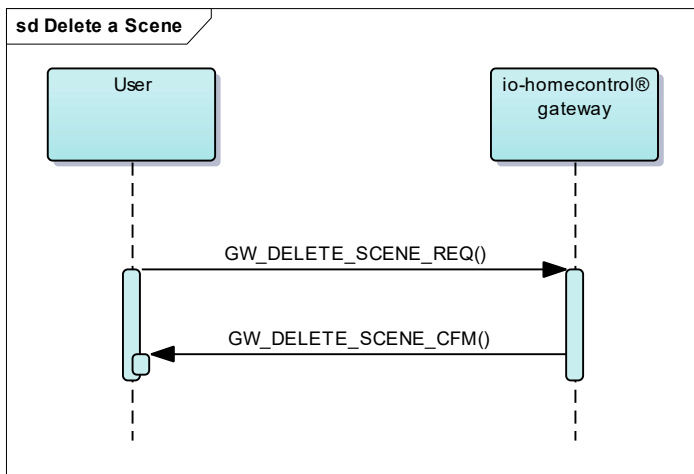
Figure 27 - Sequence diagram show how to delete a scene.

11.5.1 GW_DELETE_SCENE_REQ

Command	Data 1
GW_DELETE_SCENE_REQ	SceneID

Table 232 - GW_DELETE_SCENE_REQ frame format.

11.5.1.1 SceneID parameter

The parameter SceneID is the one byte long unique number.

11.5.2 GW_DELETE_SCENE_CFM

GW_DELETE_SCENE_CFM is send to acknowledge GW_DELETE_SCENE_REQ.

Command	Data 1	Data 2
GW_DELETE_SCENE_CFM	Status	SceneID

Table 233 - GW_DELETE_SCENE_CFM frame format.

11.5.2.1 Status

Status value	Description
0	OK - Request accepted
1	Error – Invalid scene index
Other values	Reserved

Table 234 - Status parameter description.

11.5.2.2 SceneID

Parameter description at paragraph 11.4.3.2

11.6 Rename a scene

A new name can be set to an existing scene.

11.6.1 GW_RENAME_SCENE_REQ

Command	Data 1	Data 2 - 65
GW_RENAME_SCENE_REQ	SceneID	SceneName

Table 235 - GW_RENAME_SCENE_REQ frame format.

11.6.1.1 SceneID parameter

The parameter SceneID is the one byte long unique number.

11.6.1.2 SceneName parameter

The parameter SceneName is a 64-byte long byte array. SceneName are UTF-8 encoded. If you don't use all 64 bytes, you must use space characters as padding, to fill out the remaining room.

GW_RENAME_SCENE_CFM is send to acknowledge GW_RENAME_SCENE_REQ if a scene exists with the given SceneID. If no scene exist an error command is returned instead.

Command	Data 1	Data 2
GW_RENAME_SCENE_CFM	Status	SceneID

Table 236 - GW_RENAME_SCENE_CFM frame format.

11.6.1.3 Status

Status value	Description
0	OK - Request accepted
1	Error – Invalid scene index
2	Error – Name already stored
Other values	Reserved

Table 237 - Status parameter description.

11.6.1.4 SceneID

Parameter description at paragraph 11.4.3.2

11.7 Get a list of scenes
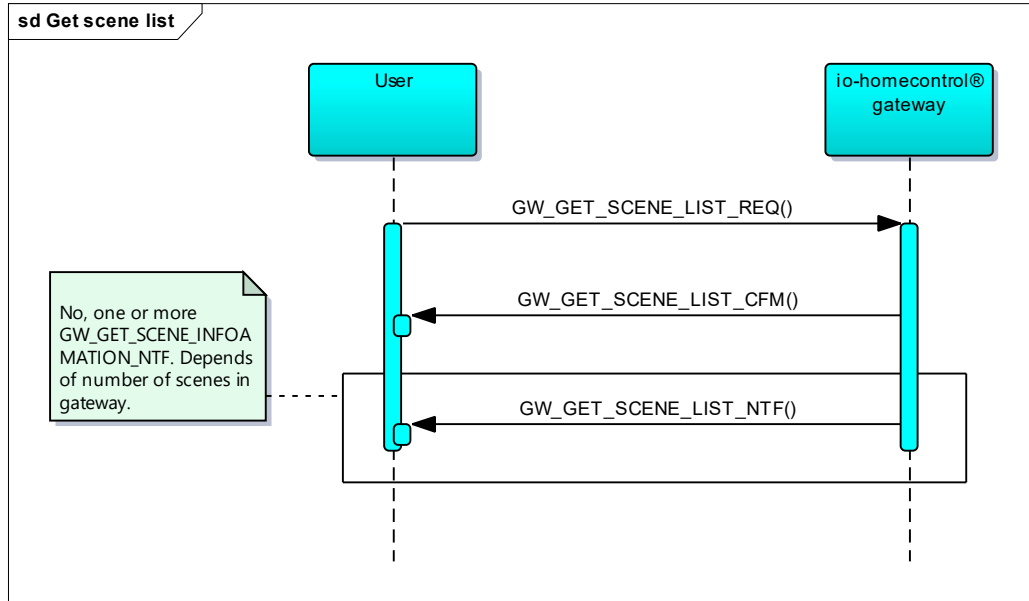
After receiving a GW_GET_SCENE_LIST_REQ frame, Gateway will send one GW_GET_SCENE_LIST_CFM and the list of known scenes, and one or more GW_GET_SCENE_LIST_NTF frame(s). See in the three tables below.

11.7.1 GW_GET_SCENE_LIST_REQ

Command
GW_GET_SCENE_LIST_REQ

Table 238 - GW_GET_SCENE_LIST_REQ frame format.

11.7.2 GW_GET_SCENE_LIST_CFM

Command	Data 1
GW_GET_SCENE_LIST_CFM	TotalNumberOfObjects

Table 239 - GW_GET_SCENE_LIST_CFM frame format.

11.7.2.1 TotalNumberOfObjects

The total number of scene objects to be returned.

11.7.3 GW_GET_SCENE_LIST_NTF

Command	Data 1	Data 2 - (n+1)	Data (n+2)
GW_GET_SCENE_LIST_NTF	NumberOfObject	SceneListObjects	RemainingNumberOfObject

Table 240 - GW_GET_SCENE_LIST_NTF frame format. Note n ∈ {65; 130; 195}.

The GW_GET_SCENE_LIST_NTF frame are a little different if there no scenes in the scene table. See example in Table 241.

Command	Data 1	Data 2
GW_GET_SCENE_LIST_NTF	NumberOfObject = 0	RemainingNumberOfObject = 0

Table 241 - GW_GET_SCENE_LIST_NTF frame format for empty scene list.

11.7.3.1 NumberOfObject parameter

The parameter NumberOfObject is an unsigned byte, indicating how many scene list objects this GW_GET_SCENE_LIST_CFM frame will carry. Valid values are [0, 1, 3].

11.7.3.2 SceneListObjects parameter

The parameter SceneListObjects are an array of scene list objects. See Table 242 below.

Data 2 – 66	Data 67 – 131	…
Object one in scene list	Object two in scene list	…

Table 242 - Frame format of the parameter SceneListObjects.

Each scene list object contains the scene identification number and the 64-byte long text label. See the objects structure in Table 243.

1 byte	64 byte
SceneID	SceneName

Table 243 - Scene list object structure.

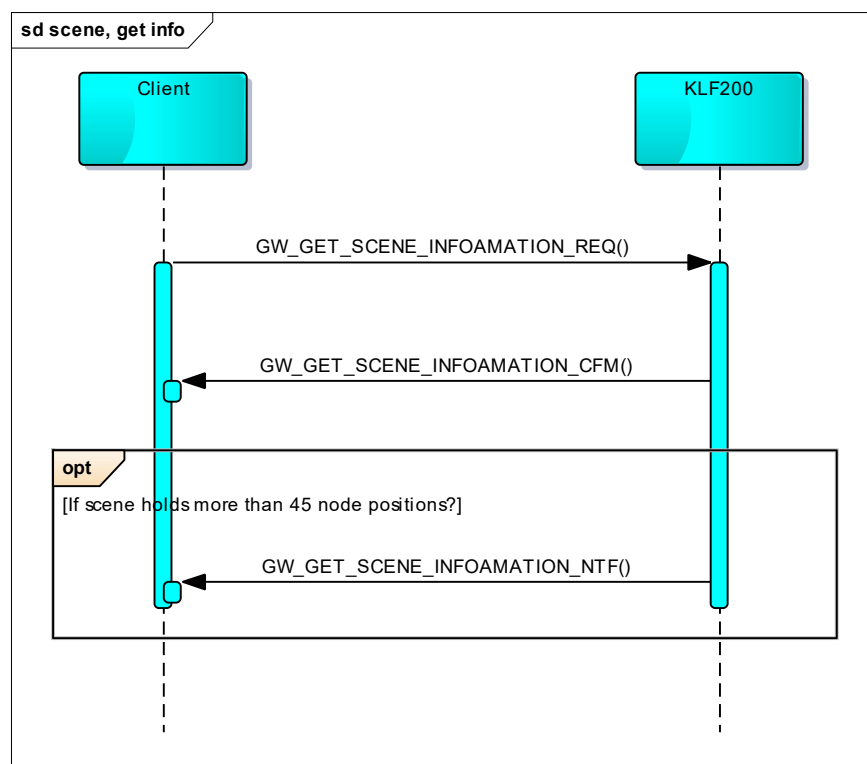### 11.7.3.3 SceneID parameter

SceneID contain scene ID.

11.7.3.4 SceneName parameter

The parameter SceneName is a 64-byte long byte array holding the name of the scene.

11.7.3.5 RemainingNumberOfObject parameter

The parameter RemainingNumberOfObject tells the remaining number of scene list object to be transferred. This means if RemainingNumberOfObject ≠ 0, the gateway will send at least one more GW_GET_SCENE_LIST_CFM frame.

11.8 Get information of one scene

11.8.1 GW_GET_SCENE_INFOAMATION_REQ

Command	Data 1
GW_GET_SCENE_INFOAMATION_REQ	SceneID

Table 244 - GW_GET_SCENE_INFOAMATION_REQ frame format.

11.8.1.1 SceneID parameter

The current scene is specified by SceneID.

11.8.2 GW_GET_SCENE_INFORMATION_CFM

Command	Data 1	Data 1
GW_GET_SCENE_INFORMATION_CFM	Status	SceneID

Table 245 - GW_GET_SCENE_INFOMRATION_CFM frame format.

11.8.2.1 Status

Status value	Description
0	OK - Request accepted
1	Error – Invalid scene index
2-255	Reserved

Table 246 - Status parameter description.

11.8.2.2 SceneID

Parameter description at paragraph 11.4.3.2

11.8.3 GW_GET_SCENE_INFORMATION_NTF

The GW_GET_SCENE_INFORMATION_NTF holds the scene name and up to 45 different node positions. If a scene contains more than 45 node positions, multiple GW_GET_SCENE_INFORMATION_NTF will be returned.

Command	Data 1	Data 2 – 65	Data 66
GW_GET_SCENE_INFORMATION_NTF	SceneID	SceneName	NumberOfNodesObjects

Data 67 - n	Data (n+1)
NodeObjects	RemaningNodeObjects

Table 247 - GW_GET_SCENE_INFORMATION_NTF frame format. Note n ∈ {70; 74; 78; … ; 246}.

11.8.3.1 NumberOfNodesObjects

NumberOfNodesObjects is the number of node positions in this frame.

11.8.3.2 NodeObjects

The parameter NodeObjects is one or an array of many instance of NodeObject.
Each NodeObject has following data:

Data Type	Field name	Valid values
8 bit unsigned integer	NodeIndex	[0;199]
8 bit unsigned integer	ParameterID	[0;16]
16 bit unsigned integer	Parameter value	[0;65535]

Table 248 - One NodeObject instance.

11.8.3.3 RemaningNodeObjects

If RemaningNodeObjects > 0, then more frames will come with more node positions for this scene.

11.9 Scene information change notification

11.9.1 GW_SCENE_INFORMATION_CHANGED_NTF

If a scene has been changed, a GW_SCENE_INFORMATION_CHANGED_NTF will be broadcasted to all connected clients. Since the NodeObjects parameter found in GW_GET_SCENE_INFORMATION_NTF is too big to fit into a single broadcast frame, the client must actively request the new information for a modified scene is needed.

Command	Data 1	Data 2
GW_SCENE_INFORMATION_CHANGED_NTF	ChangeType	SceneID

Table 249 - GW_SCENE_INFORMATION_CHANGED_NTF frame format.

11.9.1.1   ChangeType

ChangeType value	Field name
0	Scene Deleted
1	Information modified
Other values	Reserved

Table 250 – ChangeType value description

11.9.1.2   SceneID

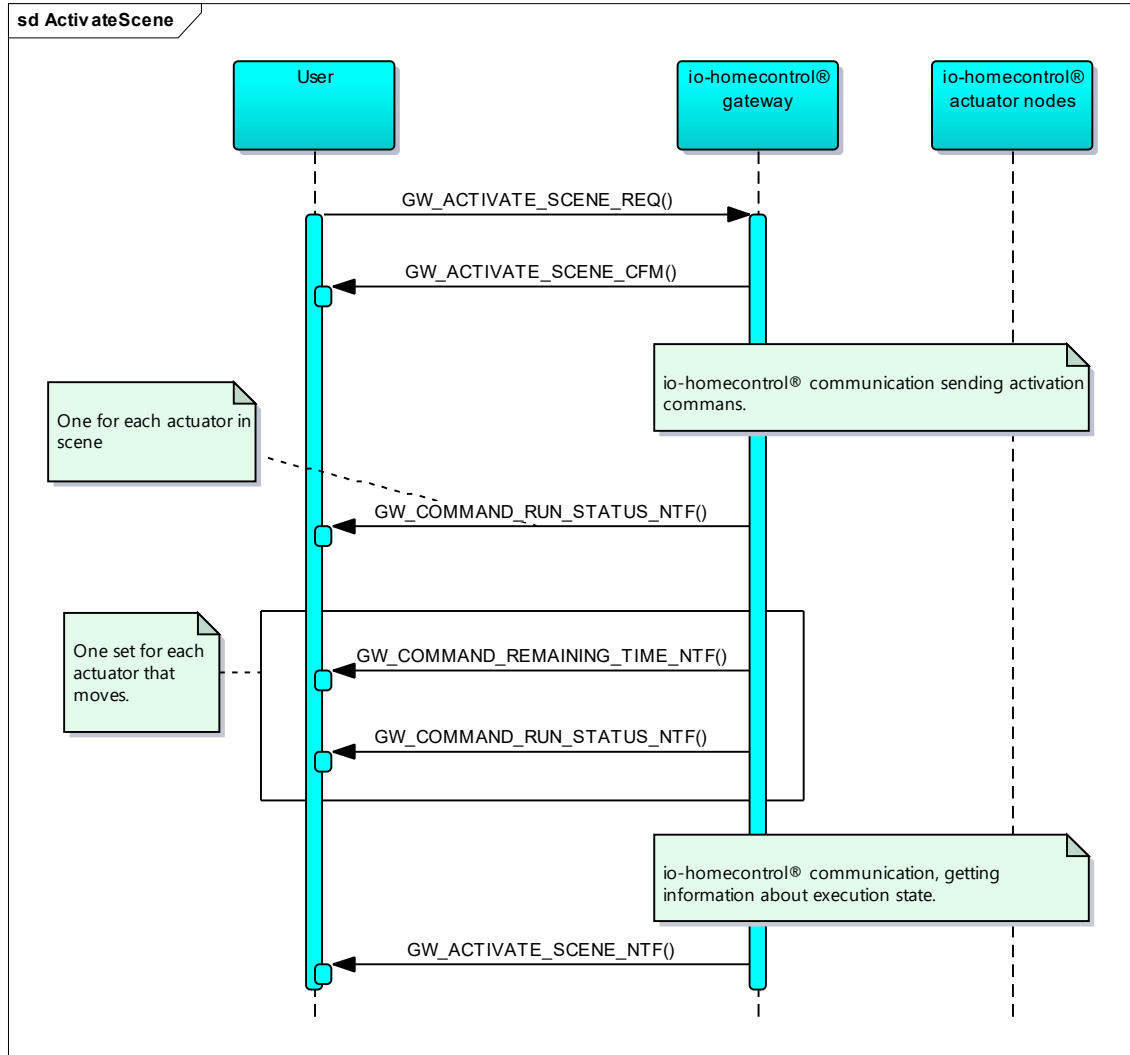Parameter description at paragraph 11.4.3.2

11.10 Activate a scene

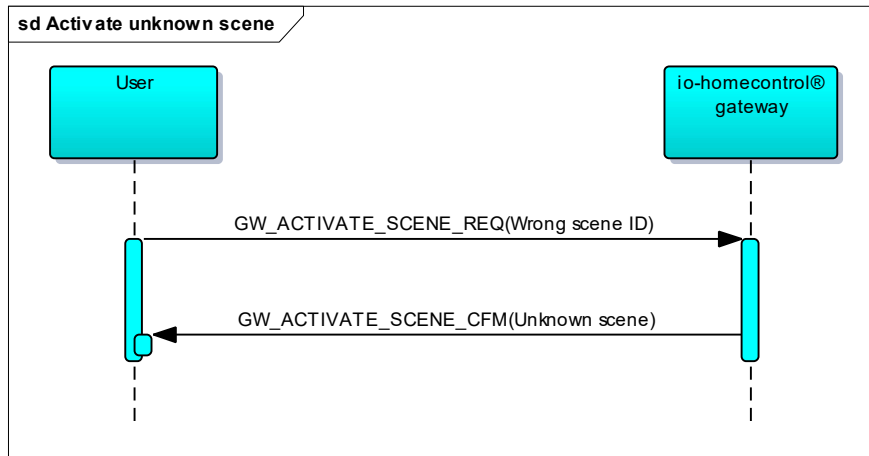**Figure 28 – Activate scene sequence diagram.**

Figure 29 – Activate unknown scene sequence diagram.

A scene can be activated by its SceneID with the command GW_ACTIVATE_SCENE_REQ. If the gateway accepts the request, it's acknowledge with GW_ACTIVATE_SCENE_CFM first and GW_ACTIVATE_SCENE_NTF after the scene is set.

11.10.1 GW_ACTIVATE_SCENE_REQ

A scene is activated by its SceneID identification number.

Command	Data 1 – 2	Data 3	Data 4	Data 5	Data 6
GW_ACTIVATE_SCENE_REQ	SessionID	CommandOriginator	PriorityLevel	SceneID	Velocity

Table 251 - GW_ACTIVATE_SCENE_REQ frame format.

11.10.1.1   SessionID parameter

Unique identification of the command. See paragraph 10.1.1.1 at page 57 for more information.

11.10.1.2   CommandOriginator parameter

See paragraph 10.1.1.2 page 57 for description.

11.10.1.3   PriorityLevel parameter

See paragraph 10.1.1.3 PriorityLevel parameter at page 57 for description.

11.10.1.4   SceneID parameter

The parameter SceneID is the one unsigned byte number. If no scene has been created with that number, the acknowledge will be GW_ERROR_NTF(Unknown scene). For more information concering the GW_ERROR_NTF command see paragraph 10 GW_ERROR_NTF command.

11.10.1.5   Velocity parameter

Velocity parameter is used to set

Velocity value	Tag	Description
0	*DEFAULT*	The product group operates by its default velocity.
1	*SILENT*	The product group operates in silent mode (slow)*.
2	*FAST*	The product group operates with fast velocity*.
3-255	-	Not defined value.

Table 252 - Velocity parameter description.

*) Some old actuators will only move at default velocity, even if speed parameter is set to SILENT or FAST

11.10.2 GW_ACTIVATE_SCENE_CFM

Command	Data 1	Data 2 – 3
GW_ACTIVATE_SCENE_CFM	Status	SessionID

Table 253 - GW_ACTIVATE_SCENE_CFM frame format.

11.10.2.1   Status

Status value	Description
0	OK - Request accepted
1	Error – Invalid parameter
2	Error – Request rejected
Other values	Reserved

Table 254 - Status parameter description.

11.10.2.2   SessionID

Parameter description at paragraph 10.1.1.1

11.10.3 GW_COMMAND_RUN_STATUS_NTF

See command description at paragraph 10.1.3

11.10.4 GW_COMMAND_REMAINING_TIME_NTF

See command description at paragraph 10.1.4

11.10.5 GW_SESSION_FINISHED_NTF

See command description at paragraph 10.1.5

11.11 Stop an activated scene

11.11.1 GW_STOP_SCENE_REQ

An activated scene can be stopped by its SceneID identification number.

Command	Data 1 – 2	Data 3	Data 4	Data 5
GW_STOP_SCENE_REQ	SessionID	CommandOriginator	PriorityLevel	SceneID

Table 255 - GW_STOP_SCENE_REQ frame format.

11.11.1.1   SessionID parameter

See paragraph 10.1.1.1 at page 57 for description.

11.11.1.2   CommandOriginator parameter

See paragraph 10.1.1.2 page 57 for description.

11.11.1.3   PriorityLevel parameter

See paragraph 10.1.1.3 at page 57 for description.

11.11.1.4   SceneID parameter

ID of scene to be stopped.

11.11.2 GW_STOP_SCENE_CFM

Command	Data 1	Data 2 – 3
GW_STOP_SCENE_CFM	Status	SessionID

Table 256 - GW_STOP_SCENE_CFM frame format.

11.11.2.1   Status

Status value	Description
0	OK - Request accepted
1	Error – Invalid parameter
2	Error – Request rejected
Other values	Reserved

Table 257 - Status parameter description.

11.11.2.2   SessionID

Parameter description at paragraph 10.1.1.1

11.11.3 GW_SESSION_FINISHED_NTF

See command description at paragraph 10.1.5

12 Contact input interface

KLF200 has 10 contact input and 5 output relays. Different actions can be assigned to each input. The output relays con be used to indicate success or failure performing the desired action. The contact input interface can be configured in two ways. A simple or advanced way. Using the simple way, contact input interface are configure using learn button and contact inputs only. The advanced way use the frames below or web interface. The advanced way offers much more than the simple.

12.1.1 GW_SET_CONTACT_INPUT_LINK_REQ

Command	Data 1	Data 2	Data 3
GW_SET_CONTACT_INPUT_LINK_REQ	ContactInputID	ContactInputAssignment	ActionID

Date 4	Data 5	Data 6	Data 7 - 8	Data 9	Data 10
CommandOriginator	PriorityLevel	ParameterID	Position	Velocity	LockPriorityLevel

Data 11	Data 12	Data 13	Data 14	Data 15	Data 16	Data 17
PLI_3	PLI_4	PLI_5	PLI_6	PLI_7	SuccessOutputID	ErrorOutputID

Table 258 - GW_SET_CONTACT_INPUT_LINK_REQ frame format for empty scene list.

12.1.1.1 ContactInputID parameter

ContactInputID can be a number from 0 to 9.

12.1.1.2 ContactInputAssignment parameter

ContactInputAssignment value	Description
0	Input not assigned.
1	Scene
2	Product group
3	One node controlled by mode
4 to 255	Not to be used!

Table 259 - ContactInputAssignment value description.

12.1.1.3 ActionID parameter

ActionID depends of 'ContactInputAssignment' parameter. ActionID can hold SceneID or ProductGroupID or NodeID.

SceneID: See paragraph 0 at page 89 for detailed description.
ProductGroupID: See paragraph 10.7.1.4 at page 82 for detailed description.
NodeID: point out a node in the system table. NodeID ϵ [0;199].

12.1.1.4 ParameterID parameter

If ContactInputAssignment is 'Product group' then ParameterID is used to specify with parameter to set a new position on. See paragraph 10.5.2.6 at page 75 for further description.

12.1.1.5 Position parameter

Position parameter is used only when ContactInputAssignment is "Product group" else position don't care. See paragraph 10.7.1.6 at page 82 for detailed description

12.1.1.6 Velocity parameter

Velocity parameter is used only when ContactInputAssignment is "Product group" else Velocity don't care.

Velocity value	Tag	Description
0	DEFAULT	Move actuator with default speed.
1	SILENT	Move actuator in silent mode*.
2	FAST	Move actuator as fast as possible*.
3 - 255	-	Do not use this values.

Table 260 - Velocity parameter description.

*) Some old actuators will only move at default velocity, even if speed parameter is set to SILENT or FAST

12.1.1.7 CommandOriginator parameter

Specifies the command originator type (USER/TIMER/SECURITY etc.)
Typically, only USER or SAAC are used.

CommandOriginator value	Tag	Description
1	USER	User Remote control causing action on actuator
2	RAIN	Rain sensor
3	TIMER	Timer controlled
5	UPS	UPS unit
8	SAAC	Stand Alone Automatic Controls
9	WIND	Wind sensor
11	LOAD_SHEDDING	Managers for requiring a particular electric load shed.
12	LOCAL_LIGHT	Local light sensor.
13	UNSPECIFIC_ENVIRONMENT_SENSOR	Used in context with commands transmitted on basis of an unknown sensor for protection of an end-product or house goods.
255	EMERGENCY	Used in context with emergency or security commands

Table 261 - CommandOriginator parameter description

12.1.1.8 PriorityLevel parameter

See paragraph 10.1.1.3 at page 57 for detailed description.

12.1.1.9 LockPriorityLevel parameter

LockPriorityLevel parameter are used only when ContactInputAssignment is "Product group" else the value is ignored.

LockPriorityLevel value	Tag	Description
0	NO	Do not lock any priority level.
1	30MIN	Lock one or more priority level in 30 minutes.
2	FOREVER	Lock one or more priority level forever
3-255	-	Not to be used!

Table 262 – LockPriorityLevel parameter description.

12.1.1.10 PLI_3, PLI_4, PLI_5, PLI_6 and PLI_7 parameters

If ContactInputAssignment is set to "Product group" then PLI_3 to PLI_7 are used.
If LockPriorityLevel parameter is set different from null, PLI_3 to PLI_7 are used to define which of the priority levels from 3 to 7 to lock.

PLI_3, PLI_4, PLI_5, PLI_6 and PLI_7 parameter value	Description
0	Disable priority (Lock for other)
1	Enable
2	Enable all
3	Keep current
4 – 255	Not to be used!

Table 263 - PLI_3, PLI_4, PLI_5, PLI_6 and PLI_7 parameter value description.

12.1.1.11   SuccessOutputID parameter

SuccessOutputID value	Description
0	Don't send any pulse.
1	Send pulse to output port number 1
2	Send pulse to output port number 2
3	Send pulse to output port number 3
4	Send pulse to output port number 4
5	Send pulse to output port number 5
6 to 255	Not to be used!

Table 264 – SuccessOutputD parameter value description.

12.1.1.12   ErrorOutputID parameter

ErrorOutputID value	Description
0	Don't send any pulse.
1	Send pulse to output port number 1
2	Send pulse to output port number 2
3	Send pulse to output port number 3
4	Send pulse to output port number 4
5	Send pulse to output port number 5
6 to 255	Not to be used!

Table 265 - ErrorOutputD parameter value description.

12.1.2 GW_SET_CONTACT_INPUT_LINK_CFM

Command	Data 1	Data 2
GW_SET_CONTACT_INPUT_LINK_CFM	ContactInputID	Status

Table 266 - GW_SET_CONTACT_INPUT_LINK_CFM frame format for empty scene list.

12.1.2.1   Status parameter

Status value	Description
0	The request failed.
1	The request was successful.

Table 267 - Status parameter

12.1.3 GW_REMOVE_CONTACT_INPUT_LINK_REQ

Command	Data 1
GW_REMOVE_CONTACT_INPUT_LINK_REQ	ContactInputID

Table 268 - GW_REMOVE_CONTACT_INPUT_LINK_REQ frame format for empty scene list.

Note: if a contact input was assigned as a product group and removed, then both ContactInputID is disabled.

12.1.4 GW_REMOVE_CONTACT_INPUT_LINK_CFM

Command	Data 1	Data 2
GW_REMOVE_CONTACT_INPUT_LINK_CFM	ContactInputID	Status

Table 269 - GW_REMOVE_CONTACT_INPUT_LINK_CFM frame format.

12.1.4.1   Status parameter

Status value	Description
0	The request failed.
1	The request was successful.

Table 270 - Status parameter

12.1.5 GW_GET_CONTACT_INPUT_LINK_LIST_REQ

Command
GW_GET_CONTACT_INPUT_LINK_LIST_REQ

Table 271 - GW_GET_CONTACT_INPUT_LINK_LIST_REQ frame format.

12.1.6 GW_GET_CONTACT_INPUT_LINK_LIST_CFM

Command	Data 1	Data 2-161
GW_GET_CONTACT_INPUT_LINK_LIST_CFM	NumberOfObject = 10	ContactInputObjects

Table 272 - GW_GET_CONTACT_INPUT_LINK_LIST_CFM frame format.

Data 2 – 18	Data 19 – 35	Data 36 – 52	…	Data 84 - 171
First ContactInputObject	Second ContactInputObject	Third ContactInputObject	…	Tenth ContactInputObject

Table 273 - Frame format of the parameter ContactInputObjects.

ContactInputObject	
Byte Index	Description
1	ContactInputID [0;9]
2	ContactInputAssignment 0 ~ ContactInput is not assigned. 1 ~ Scene 2 ~ ProductGroup 3 ~ OneNodeControlledByMode 4 – 255 ~ Not defined. Will not be use.
3	SceneID if ContactInputAssignment = Scene. ProductGroupID if ContactInputAssignment = ProductGroup. NodeID if ContactInputAssignment = OneNodeControlledByMode. Undefined if ContactInputAssignment = 0.
4	CommandOriginator
5	PriorityLevel [0;7]
6	ParameterID
7 - 8	Position
9	Velosity [0;2]
10	LockPriorityLevel [0;2]
11	PLI_3
12	PLI_4
13	PLI_5
14	PLI_6
15	PLI_7
16	SuccessOutputID 0 ~ Success are not signalised on any pin. 1 – 5 ~ Success are signalised on pin corresponding to the number. 6 – 255 ~ Not defined. Do not use.
17	ErrorOutputID 0 ~ Error are not signalised on any pin. 1 – 5 ~ Error are signalised on pin corresponding to the number. 6 – 255 ~ Not defined. Do not use.

Table 274 - Format of each ContactInputObject.

13 Appendix 1: Standard Parameter definition

Parameter value is the values possible to set to a specified parameter (MP, FP). The value range is as standard a 16-bit value [0 – 65535]. The value range is split-up several different access methods.

Access Method name for Standard Parameter	Description	Range (Hex)	Size (Dec)
Relative	Relative value (0 – 100%)	0000 $_{HEX}$ – C800 $_{HEX}$	51201
Percent+-	Percentage point plus or minus (-100% – 100%)	C900 $_{HEX}$ – D0D0 $_{HEX}$	2001
Target	The target value for the parameter	D100 $_{HEX}$	1
Current	The current value for the parameter	D200 $_{HEX}$	1
Default	The default value for the parameter	D300 $_{HEX}$	1
Ignore	Ignore the parameter field where this Access Method is written	D400 $_{HEX}$	1

Table 275 - Access Methods.

If a read-only FP is included in the frame its value must be set to 'Ignore'. If this is not respected command status will be 'Parameter incoherence /adjustment'.

13.1 Relative

Using the Relative access method makes it possible to activate a parameter without knowing the absolute value specified for this parameter. 100% is indicated as 51200 which are equal to C800$_{HEX}$.

13.2 Percent+-

The Percent+- access method is used for changing the current relative value in the Actuator per a given percentage point.

If the command is rejected by the actuator (actuator version supports only one-way mode – general ACK command with command status set to Total incoherence) then automatically the controller must use an alternative mean for achieving the expected result.
In two-way mode, an alternative mean can perform the same functionality by making a status request to get the current value, which then can be converted into a new value (e.g. doing a +5-percentage point calculation) and finally sent to the Actuator using the Relative access method.

13.3 Target

It is implemented by taking the target parameter value and loads it into the execution parameter buffer.
When the target value is read, it holds for a given parameter always the latest stored target value about a command execution.

13.4 Current

Used to cancel, stop or freeze an operation.
It is implemented by taking the current parameter value and loading it into the execution parameter buffer.

Current value interpretation when it is read / handled in the Actuator:

The current value for a given parameter is continuously updated. The following two examples show how the current value changes for two different types of parameters.

1. For a parameter indicating speed, the current value can be different from 0 when the Actuator is executing the command, where the current value for this parameter always will be equal to 0 when the command has been executed.
2. For a parameter indicating a position, the current value can be equal to the target value when the Actuator has executed the command.

This means, that the target and current values for a given parameter are not necessarily identical when an Actuator has executed a command. I.e. it cannot be deduced from comparing the current and target values for the different parameters whether the Actuator is finished executing a command. Instead the Execute status / detailed execute status included in the ACTIVATE_ACK command frame must be read.

13.5 Default

Default can either be specified to be a relative value for the parameter, Access method target or Access method current. In the Actuator profile, it is described what Default is specified to.

13.6 Ignore

The Ignore Access Method is used where a parameter in the frame is to be ignored.

14 Appendix 2: List of actuator types and their use of Main Parameter and Functional Parameters

	Node Type / Sub Type value	Actuator Profiles	Actuator Sub-Profiles	Main Parameter	Functional Parameter #1 Generic Function: MP Speed	Functional Parameter #2 Generic Function: Tilting Speed	Functional Parameter #3 Generic Function: Tilting
1	0x0040	**Interior Venetian Blind**		Position of the blind	Orientation of the slats	Speed of the slats during orientation	Linear speed of the blind
2	0x0080	**Roller Shutter**		Position of the shutter	Linear speed of the shutter		
2.1	0x0081		**Adjustable slats rolling shutter**	Position of the shutter	Linear speed of the shutter	Speed of the slats during orientation	Orientation of the slats
2.2	0x0082		**With projection**	Position of the shutter	Linear speed of the shutter		
3	0x00C0	**Vertical Exterior Awning**		Position of the awning	Linear speed of the awning		
4	0x0100	**Window opener**		Position of the Window	Linear speed of Window		
4.1	0x0101		**Window opener with integrated rain sensor**	Position of the Window	Linear speed of Window		
5	0x0140	**Garage door opener**		Linear or angular position of the garage door	Linear or angular speed of the door		
●/○ 5.58	0x017A			Linear or angular position of the garage door			
6	0x0180	**Light**		Light intensity	Light intensity gradient		
●/○ 6.58	0x01BA		**Light only supporting on/off**	Light intensity			
7	0x01C0	**Gate opener**		Position of the gate	Speed of the gate		
●/○ 7.58	0x01FA			Linear or angular position of the gate			
9	0x0240	**Door lock**		Door lock state (opened/closed)			
9.1	0x0241	**Window lock**		Window lock state (opened/closed)			

10	0x0280	**Vertical Interior Blinds**		Position of the blind	Linear speed of the blind		
13	0x0340	**Dual Roller Shutter**		Position of the two curtains	Position of the upper curtain	Position of the lower curtain	Linear speed of the two curtains
15	0x03C0	**On/Off switch**		Switch position			
16	0x0400	**Horizontal awning**		Position of the awning	Linear speed of the awning		
17	0x0440	**Exterior Venetian blind**		Position of the blind	Linear speed of the blind	Speed of the slats during orientation	Orientation of the slats
18	0x0480	**Louver blind**		Position of the curtain	Linear speed of the curtain	Speed of the hangers during orientation	Orientation of the hangers
19	0x04C0	**Curtain track**		Position of the curtain	Linear speed of the curtain		
20	0x0500	**Ventilation point**		Air demand			
20.1	0x0501		**Air inlet**	Air demand			
20.2	0x0502		**Air transfer**	Air demand			
20.3	0x0503		**Air outlet**	Air demand			
21	0x0540	**Exterior heating**		Energy demand	Energy gradient		
21.58	0x57A			Energy demand			
24	0x0600	**Swinging Shutters**		Shutter closure	Shutter speed		
24.1	0x0601		**Swinging Shutter with independent handling of the leaves**	Shutter closure	Shutter speed		

Table 276 - Actuator list.

Cells starts with a red line,
differs from the generic function.

14.1 Effect off Main parameter value

The effect of the main parameter is adjusted so that it is possible to use a keyboard with up, down and stop, so that the up button always sends MP = 0x0000 and down button always sends 0xC800. Stop button sends MP = Current = 0xD200.

Actuator profile name	Main parameter = 0x0000	Main parameter = 0xC800
Interior Venetian Blind	0 % down. Light can freely flow through window.	100 % down. Light flowing through the window is limited.
Roller Shutter	0 % down. Light can freely flow through window.	100 % down. Light flowing through the window is limited.
Vertical Exterior Awning	0 % down. Light can freely flow through window.	100 % down. Light flowing through the window is limited.
Window opener	100 % open.	0 % open.
Garage door opener	0% closed. Means door is open.	100 % closed.
Light	100 % light output.	0 % light output.
Gate opener	0% closed. Gate is open.	100 % closed.
Door lock	Unlocked	Locked
Window lock	Unlocked	Locked
Vertical Interior Blinds	0 % down. Light can freely flow through window.	100 % down. Light flowing through the window is limited.
Dual Roller Shutter	0 % down. Light can freely flow through windows.	100 % down. Light flowing through the windows is limited.
On/Off switch	On	Off
Horizontal awning	0 % out. Awnings are rolled up.	Awnings are rolled 100 % out.
Exterior Venetian blind	0 % down. Light can freely flow through window.	100 % down. Light flowing through the window is limited.
Louver blind	0% covered. Light can freely flow through window.	100 % covered. Light flowing through the window is limited.
Curtain track	0% covered. Light can freely flow through window.	100 % covered. Light flowing through the window is limited.
Ventilation point	Maximum allowable ventilation.	Minimum allowable ventilation.
Exterior heating	100 % heat.	0 % heat.
Swinging Shutters	0% covered. Light can freely flow through window.	100 % covered. Light flowing through the window is limited.

Table 277 - Effect off Main parameter value.

14.2 Alias for actuator specific parameter values

14.2.1 Window Opener Actuator Profile

Alias name   Secured Ventilation
Alias ID   0xD803
Description   A position a window can be opened to for getting some ventilation and where the window is still locked.

15 Appendix 3: List of Gateway commands

Number	Command	Short description
0x0000	GW_ERROR_NTF	Provides information on what triggered the error.
0x0001	GW_REBOOT_REQ	Request gateway to reboot.
0x0002	GW_REBOOT_CFM	Acknowledge to GW_REBOOT_REQ command.
0x0003	GW_SET_FACTORY_DEFAULT_REQ	Request gateway to clear system table, scene table and set Ethernet settings to factory default. Gateway will reboot.
0x0004	GW_SET_FACTORY_DEFAULT_CFM	Acknowledge to GW_SET_FACTORY_DEFAULT_REQ command.
0x0008	GW_GET_VERSION_REQ	Request version information.
0x0009	GW_GET_VERSION_CFM	Acknowledge to GW_GET_VERSION_REQ command.
0x000A	GW_GET_PROTOCOL_VERSION_REQ	Request KLF 200 API protocol version.
0x000B	GW_GET_PROTOCOL_VERSION_CFM	Acknowledge to GW_GET_PROTOCOL_VERSION_REQ command.
0x000C	GW_GET_STATE_REQ	Request the state of the gateway
0x000D	GW_GET_STATE_CFM	Acknowledge to GW_GET_STATE_REQ command.
0x000E	GW_LEAVE_LEARN_STATE_REQ	Request gateway to leave learn state.
0x000F	GW_LEAVE_LEARN_STATE_CFM	Acknowledge to GW_LEAVE_LEARN_STATE_REQ command.
0x00E0	GW_GET_NETWORK_SETUP_REQ	Request network parameters.
0x00E1	GW_GET_NETWORK_SETUP_CFM	Acknowledge to GW_GET_NETWORK_SETUP_REQ.
0x00E2	GW_SET_NETWORK_SETUP_REQ	Set network parameters.
0x00E3	GW_SET_NETWORK_SETUP_CFM	Acknowledge to GW_SET_NETWORK_SETUP_REQ.
0x0100	GW_CS_GET_SYSTEMTABLE_DATA_REQ	Request a list of nodes in the gateways system table.
0x0101	GW_CS_GET_SYSTEMTABLE_DATA_CFM	Acknowledge to GW_CS_GET_SYSTEMTABLE_DATA_REQ
0x0102	GW_CS_GET_SYSTEMTABLE_DATA_NTF	Acknowledge to GW_CS_GET_SYSTEM_TABLE_DATA_REQList of nodes in the gateways systemtable.
0x0103	GW_CS_DISCOVER_NODES_REQ	Start CS DiscoverNodes macro in KLF200.
0x0104	GW_CS_DISCOVER_NODES_CFM	Acknowledge to GW_CS_DISCOVER_NODES_REQ command.
0x0105	GW_CS_DISCOVER_NODES_NTF	Acknowledge to GW_CS_DISCOVER_NODES_REQ command.
0x0106	GW_CS_REMOVE_NODES_REQ	Remove one or more nodes in the systemtable.
0x0107	GW_CS_REMOVE_NODES_CFM	Acknowledge to GW_CS_REMOVE_NODES_REQ.
0x0108	GW_CS_VIRGIN_STATE_REQ	Clear systemtable and delete system key.
0x0109	GW_CS_VIRGIN_STATE_CFM	Acknowledge to GW_CS_VIRGIN_STATE_REQ.
0x010A	GW_CS_CONTROLLER_COPY_REQ	Setup KLF200 to get or give a system to or from another io-homecontrol® remote control. By a system means all nodes in the systemtable and the system key.
0x010B	GW_CS_CONTROLLER_COPY_CFM	Acknowledge to GW_CS_CONTROLLER_COPY_REQ.
0x010C	GW_CS_CONTROLLER_COPY_NTF	Acknowledge to GW_CS_CONTROLLER_COPY_REQ.
0x010D	GW_CS_CONTROLLER_COPY_CANCEL_NTF	Cancellation of system copy to other controllers.
0x010E	GW_CS_RECEIVE_KEY_REQ	Receive system key from another controller.
0x010F	GW_CS_RECEIVE_KEY_CFM	Acknowledge to GW_CS_RECEIVE_KEY_REQ.

0x0110	GW_CS_RECEIVE_KEY_NTF	Acknowledge to GW_CS_RECEIVE_KEY_REQ with status.
0x0111	GW_CS_PGC_JOB_NTF	Information on Product Generic Configuration job initiated by press on PGC button.
0x0112	GW_CS_SYSTEM_TABLE_UPDATE_NTF	Broadcasted to all clients and gives information about added and removed actuator nodes in system table.
0x0113	GW_CS_GENERATE_NEW_KEY_REQ	Generate new system key and update actuators in systemtable.
0x0114	GW_CS_GENERATE_NEW_KEY_CFM	Acknowledge to GW_CS_GENERATE_NEW_KEY_REQ.
0x0115	GW_CS_GENERATE_NEW_KEY_NTF	Acknowledge to GW_CS_GENERATE_NEW_KEY_REQ with status.
0x0116	GW_CS_REPAIR_KEY_REQ	Update key in actuators holding an old key.
0x0117	GW_CS_REPAIR_KEY_CFM	Acknowledge to GW_CS_REPAIR_KEY_REQ.
0x0118	GW_CS_REPAIR_KEY_NTF	Acknowledge to GW_CS_REPAIR_KEY_REQ with status.
0x0119	GW_CS_ACTIVATE_CONFIGURATION_MODE_REQ	Request one or more actuator to open for configuration.
0x011A	GW_CS_ACTIVATE_CONFIGURATION_MODE_CFM	Acknowledge to GW_CS_ACTIVATE_CONFIGURATION_MODE_REQ.
0x0200	GW_GET_NODE_INFORMATION_REQ	Request extended information of one specific actuator node.
0x0201	GW_GET_NODE_INFORMATION_CFM	Acknowledge to GW_GET_NODE_INFORMATION_REQ.
0x0210	GW_GET_NODE_INFORMATION_NTF	Acknowledge to GW_GET_NODE_INFORMATION_REQ.
0x0202	GW_GET_ALL_NODES_INFORMATION_REQ	Request extended information of all nodes.
0x0203	GW_GET_ALL_NODES_INFORMATION_CFM	Acknowledge to GW_GET_ALL_NODES_INFORMATION_REQ
0x0204	GW_GET_ALL_NODES_INFORMATION_NTF	Acknowledge to GW_GET_ALL_NODES_INFORMATION_REQ. Holds node information
0x0205	GW_GET_ALL_NODES_INFORMATION_FINISHED_NTF	Acknowledge to GW_GET_ALL_NODES_INFORMATION_REQ. No more nodes.
0x0206	GW_SET_NODE_VARIATION_REQ	Set node variation.
0x0207	GW_SET_NODE_VARIATION_CFM	Acknowledge to GW_SET_NODE_VARIATION_REQ.
0x0208	GW_SET_NODE_NAME_REQ	Set node name.
0x0209	GW_SET_NODE_NAME_CFM	Acknowledge to GW_SET_NODE_NAME_REQ.
0x020A	GW_SET_NODE_VELOCITY_REQ	Set node velocity.
0x020B	GW_SET_NODE_VELOCITY_CFM	Acknowledge to GW_SET_NODE_VELOCITY_REQ.
0x020C	GW_NODE_INFORMATION_CHANGED_NTF	Information has been updated.
0x0211	GW_NODE_STATE_POSITION_CHANGED_NTF	Information has been updated.
0x020D	GW_SET_NODE_ORDER_AND_PLACEMENT_REQ	Set search order and room placement.
0x020E	GW_SET_NODE_ORDER_AND_PLACEMENT_CFM	Acknowledge to GW_SET_NODE_ORDER_AND_PLACEMENT_REQ.
0x0220	GW_GET_GROUP_INFORMATION_REQ	Request information about all defined groups.
0x0221	GW_GET_GROUP_INFORMATION_CFM	Acknowledge to GW_GET_GROUP_INFORMATION_REQ.
0x0230	GW_GET_GROUP_INFORMATION_NTF	Acknowledge to GW_GET_NODE_INFORMATION_REQ.
0x0222	GW_SET_GROUP_INFORMATION_REQ	Change an existing group.
0x0223	GW_SET_GROUP_INFORMATION_CFM	Acknowledge to GW_SET_GROUP_INFORMATION_REQ.
0x0224	GW_GROUP_INFORMATION_CHANGED_NTF	Broadcast to all, about group information of a group has been changed.
0x0225	GW_DELETE_GROUP_REQ	Delete a group.
0x0226	GW_DELETE_GROUP_CFM	Acknowledge to GW_DELETE_GROUP_INFORMATION_REQ.
0x0227	GW_NEW_GROUP_REQ	Request new group to be created.
0x0228	GW_NEW_GROUP_CFM	
0x0229	GW_GET_ALL_GROUPS_INFORMATION_REQ	Request information about all defined groups.

0x022A	GW_GET_ALL_GROUPS_INFORMATION_CFM	Acknowledge to GW_GET_ALL_GROUPS_INFORMATION_REQ.
0x022B	GW_GET_ALL_GROUPS_INFORMATION_NTF	Acknowledge to GW_GET_ALL_GROUPS_INFORMATION_REQ.
0x022C	GW_GET_ALL_GROUPS_INFORMATION_FINISHED_NTF	Acknowledge to GW_GET_ALL_GROUPS_INFORMATION_REQ.
0x022D	GW_GROUP_DELETED_NTF	GW_GROUP_DELETED_NTF is broadcasted to all, when a group has been removed.
0x0240	GW_HOUSE_STATUS_MONITOR_ENABLE_REQ	Enable house status monitor.
0x0241	GW_HOUSE_STATUS_MONITOR_ENABLE_CFM	Acknowledge to GW_HOUSE_STATUS_MONITOR_ENABLE_REQ.
0x0242	GW_HOUSE_STATUS_MONITOR_DISABLE_REQ	Disable house status monitor.
0x0243	GW_HOUSE_STATUS_MONITOR_DISABLE_CFM	Acknowledge to GW_HOUSE_STATUS_MONITOR_DISABLE_REQ.
0x0300	GW_COMMAND_SEND_REQ	Send activating command direct to one or more io-homecontrol® nodes.
0x0301	GW_COMMAND_SEND_CFM	Acknowledge to GW_COMMAND_SEND_REQ.
0x0302	GW_COMMAND_RUN_STATUS_NTF	Gives run status for io-homecontrol® node.
0x0303	GW_COMMAND_REMAINING_TIME_NTF	Gives remaining time before io-homecontrol® node enter target position.
0x0304	GW_SESSION_FINISHED_NTF	Command send, Status request, Wink, Mode or Stop session is finished.
0x0305	GW_STATUS_REQUEST_REQ	Get status request from one or more io-homecontrol® nodes.
0x0306	GW_STATUS_REQUEST_CFM	Acknowledge to GW_STATUS_REQUEST_REQ.
0x0307	GW_STATUS_REQUEST_NTF	Acknowledge to GW_STATUS_REQUEST_REQ. Status request from one or more io-homecontrol® nodes.
0x0308	GW_WINK_SEND_REQ	Request from one or more io-homecontrol® nodes to Wink.
0x0309	GW_WINK_SEND_CFM	Acknowledge to GW_WINK_SEND_REQ
0x030A	GW_WINK_SEND_NTF	Status info for performed wink request.
0x0310	GW_SET_LIMITATION_REQ	Set a parameter limitation in an actuator.
0x0311	GW_SET_LIMITATION_CFM	Acknowledge to GW_SET_LIMITATION_REQ.
0x0312	GW_GET_LIMITATION_STATUS_REQ	Get parameter limitation in an actuator.
0x0313	GW_GET_LIMITATION_STATUS_CFM	Acknowledge to GW_GET_LIMITATION_STATUS_REQ.
0x0314	GW_LIMITATION_STATUS_NTF	Hold information about limitation.
0x0320	GW_MODE_SEND_REQ	Send Activate Mode to one or more io-homecontrol® nodes.
0x0321	GW_MODE_SEND_CFM	Acknowledge to GW_MODE_SEND_REQ
0x0322	GW_MODE_SEND_NTF	Notify with Mode activation info.
0x0400	GW_INITIALIZE_SCENE_REQ	Prepare gateway to record a scene.
0x0401	GW_INITIALIZE_SCENE_CFM	Acknowledge to GW_INITIALIZE_SCENE_REQ.
0x0402	GW_INITIALIZE_SCENE_NTF	Acknowledge to GW_INITIALIZE_SCENE_REQ.
0x0403	GW_INITIALIZE_SCENE_CANCEL_REQ	Cancel record scene process.
0x0404	GW_INITIALIZE_SCENE_CANCEL_CFM	Acknowledge to GW_INITIALIZE_SCENE_CANCEL_REQ command.
0x0405	GW_RECORD_SCENE_REQ	Store actuator positions changes since GW_INITIALIZE_SCENE, as a scene.
0x0406	GW_RECORD_SCENE_CFM	Acknowledge to GW_RECORD_SCENE_REQ.
0x0407	GW_RECORD_SCENE_NTF	Acknowledge to GW_RECORD_SCENE_REQ.
0x0408	GW_DELETE_SCENE_REQ	Delete a recorded scene.
0x0409	GW_DELETE_SCENE_CFM	Acknowledge to GW_DELETE_SCENE_REQ.
0x040A	GW_RENAME_SCENE_REQ	Request a scene to be renamed.

0x040B	GW_RENAME_SCENE_CFM	Acknowledge to GW_RENAME_SCENE_REQ.
0x040C	GW_GET_SCENE_LIST_REQ	Request a list of scenes.
0x040D	GW_GET_SCENE_LIST_CFM	Acknowledge to GW_GET_SCENE_LIST.
0x040E	GW_GET_SCENE_LIST_NTF	Acknowledge to GW_GET_SCENE_LIST.
0x040F	GW_GET_SCENE_INFOAMATION_REQ	Request extended information for one given scene.
0x0410	GW_GET_SCENE_INFOAMATION_CFM	Acknowledge to GW_GET_SCENE_INFOAMATION_REQ.
0x0411	GW_GET_SCENE_INFOAMATION_NTF	Acknowledge to GW_GET_SCENE_INFOAMATION_REQ.
0x0412	GW_ACTIVATE_SCENE_REQ	Request gateway to enter a scene.
0x0413	GW_ACTIVATE_SCENE_CFM	Acknowledge to GW_ACTIVATE_SCENE_REQ.
0x0415	GW_STOP_SCENE_REQ	Request all nodes in a given scene to stop at their current position.
0x0416	GW_STOP_SCENE_CFM	Acknowledge to GW_STOP_SCENE_REQ.
0x0419	GW_SCENE_INFORMATION_CHANGED_NTF	A scene has either been changed or removed.
0x0447	GW_ACTIVATE_PRODUCTGROUP_REQ	Activate a product group in a given direction.
0x0448	GW_ACTIVATE_PRODUCTGROUP_CFM	Acknowledge to GW_ACTIVATE_PRODUCTGROUP_REQ.
0x0449	GW_ACTIVATE_PRODUCTGROUP_NTF	Acknowledge to GW_ACTIVATE_PRODUCTGROUP_REQ.
0x0460	GW_GET_CONTACT_INPUT_LINK_LIST_REQ	Get list of assignments to all Contact Input to scene or product group.
0x0461	GW_GET_CONTACT_INPUT_LINK_LIST_CFM	Acknowledge to GW_GET_CONTACT_INPUT_LINK_LIST_REQ.
0x0462	GW_SET_CONTACT_INPUT_LINK_REQ	Set a link from a Contact Input to a scene or product group.
0x0463	GW_SET_CONTACT_INPUT_LINK_CFM	Acknowledge to GW_SET_CONTACT_INPUT_LINK_REQ.
0x0464	GW_REMOVE_CONTACT_INPUT_LINK_REQ	Remove a link from a Contact Input to a scene.
0x0465	GW_REMOVE_CONTACT_INPUT_LINK_CFM	Acknowledge to GW_REMOVE_CONTACT_INPUT_LINK_REQ.
0x0500	GW_GET_ACTIVATION_LOG_HEADER_REQ	Request header from activation log.
0x0501	GW_GET_ACTIVATION_LOG_HEADER_CFM	Confirm header from activation log.
0x0502	GW_CLEAR_ACTIVATION_LOG_REQ	Request clear all data in activation log.
0x0503	GW_CLEAR_ACTIVATION_LOG_CFM	Confirm clear all data in activation log.
0x0504	GW_GET_ACTIVATION_LOG_LINE_REQ	Request line from activation log.
0x0505	GW_GET_ACTIVATION_LOG_LINE_CFM	Confirm line from activation log.
0x0506	GW_ACTIVATION_LOG_UPDATED_NTF	Confirm line from activation log.
0x0507	GW_GET_MULTIPLE_ACTIVATION_LOG_LINES_REQ	Request lines from activation log.
0x0508	GW_GET_MULTIPLE_ACTIVATION_LOG_LINES_NTF	Error log data from activation log.
0x0509	GW_GET_MULTIPLE_ACTIVATION_LOG_LINES_CFM	Confirm lines from activation log.
0x2000	GW_SET_UTC_REQ	Request to set UTC time.
0x2001	GW_SET_UTC_CFM	Acknowledge to GW_SET_UTC_REQ.
0x2002	GW_RTC_SET_TIME_ZONE_REQ	Set time zone and daylight savings rules.
0x2003	GW_RTC_SET_TIME_ZONE_CFM	Acknowledge to GW_RTC_SET_TIME_ZONE_REQ.
0x2004	GW_GET_LOCAL_TIME_REQ	Request the local time based on current time zone and daylight savings rules.
0x2005	GW_GET_LOCAL_TIME_CFM	Acknowledge to GW_RTC_SET_TIME_ZONE_REQ.

0x3000	GW_PASSWORD_ENTER_REQ	Enter password to authenticate request
0x3001	GW_PASSWORD_ENTER_CFM	Acknowledge to GW_PASSWORD_ENTER_REQ
0x3002	GW_PASSWORD_CHANGE_REQ	Request password change.
0x3003	GW_PASSWORD_CHANGE_CFM	Acknowledge to GW_PASSWORD_CHANGE_REQ.
0x3004	GW_PASSWORD_CHANGE_NTF	Acknowledge to GW_PASSWORD_CHANGE_REQ. Broadcasted to all connected clients.

Table 278 - List of KLF 200 API commands.