Mode_Lighting_Control_small 50px

Title:

Version:

Author:

Date: 14 February 2018

**History**

|  |  |  |  |
| --- | --- | --- | --- |
| Date | Ver | Initials | Changes |
| 8-May-14 | 0.1 | Jph | Initial attempt from Technical Specification v0.6  Gateway version 1.00 |
| 9-Jun-14 | 0.2 | Jph | Gateway version 1.01  Rename ?SCN to ?SCNCHANS  Modify ?AREANAMES and ?SCNNAMES to include area number  Add ?CHANNAMES query |
| 16-Apr-15 | 0.3 | Jph | Gateway version 1.02 Add HTTP connection info Add DBGECHO and DBGACK cmd & query  Add $BTNSTATE cmd  Add SCNOFF, SCNFAST, SCNRAMP, SCNTOOGLE, SCNONOFF cmd & event  Add ?SYSTEMID, ?BTNSTYLE  Remove INPSET and BTNSET cmd & query  Remove SCNBTNCOLR and SCNBTNTEXT events |
| 28-Sep-15 | 1.0.0 | Jph | GATEWAY ver 1.03  Add USER cmds & queries  Add PLATENAMES cmd  Add enumerations section  Minor documentation changes |
| 29-Apr-16 | 1.1.0 | Jph | GATEWAY ver 1.04  Add DMX cmds & queries  Add INPPIR cmd & event |
| 9-May-16 | 1.1.1 | Jph | Add SCNBACKON cmd & event  Correct SCNDMX to SCNDMXLEVEL |
| 13-Sep-16 | 1.2.0 | jph | GATEWAY ver 1.05 Add Status codes table Add DALISCAN, DALIREPAIR, DALIACCEPT cmd & event Add SHOWxxx cmds Add EMxxx cmds & queries |
| 22-Sep-16 | 1.2.1 | Jph | Add missing ?SCN/?SCNS queries  Add missing $SHOWxxx commands  Add EVTADV cmd & query  Amend DALISCAN & add DALIFIX |
| 12-Dec-16 | 1.3.0 | Jph | GATEWAY ver 1.06 Add CHANCOLR cmd & event Add CHANMODE cmd & event  Add MODUPALETTE cmd |
| 28-Mar-17 | 1.3.1 | Jph | GATEWAY ver 1.07  Add DALIERR query  Add ERRORS query  Add DALI fixture status codes table |
| 14-Feb-17 | 1.3.2 | Jph | GATEWAY ver 1.08  Document corrections and improvements  Add DALIDONE command  Revise DALIERR query  Remove unimplemented CHANCOLR, CHANMODE, and MODUPALETTE references |

**Table of Contents**

[1 Introduction 4](#_Toc506375676)

[1.1 Purpose and scope 4](#_Toc506375677)

[1.2 Terminology 4](#_Toc506375678)

[1.3 References 4](#_Toc506375679)

[2 Overview 5](#_Toc506375680)

[3 General Syntax and Operation 6](#_Toc506375681)

[3.1 Syntax 6](#_Toc506375682)

[3.2 Connections 7](#_Toc506375683)

[3.2.1 Raw TCP/IP 7](#_Toc506375684)

[3.2.2 RS232 7](#_Toc506375685)

[3.2.3 HTTP 7](#_Toc506375686)

[3.3 Message acknowledgements 8](#_Toc506375687)

[4 Commands 9](#_Toc506375688)

[4.1 Administration Level Interface 9](#_Toc506375689)

[4.1.1 Commands 9](#_Toc506375690)

[4.1.2 Queries 12](#_Toc506375691)

[4.1.3 Events: 14](#_Toc506375692)

[4.2 Channel Level Interface 14](#_Toc506375693)

[4.2.1 Output Channel Commands 15](#_Toc506375694)

[4.2.2 Input Channel Commands 16](#_Toc506375695)

[4.2.3 DMX Colour Commands 16](#_Toc506375696)

[4.2.4 Identify Channel Commands 17](#_Toc506375697)

[4.2.5 Channel State Queries 18](#_Toc506375698)

[4.2.6 Channel Access Queries 20](#_Toc506375699)

[4.2.7 General Channel Events 21](#_Toc506375700)

[4.2.8 DMX Colour Events 23](#_Toc506375701)

[4.2.9 Channel Status Events 23](#_Toc506375702)

[4.2.10 Advanced DALI 24](#_Toc506375703)

[4.2.11 Advanced DALI – Errors & Repair 25](#_Toc506375704)

[4.2.12 Advanced DALI – Emergency Lighting 27](#_Toc506375705)

[4.3 System Level Interface 30](#_Toc506375706)

[4.3.1 Scene Recall Commands 30](#_Toc506375707)

[4.3.2 Scene Adjustment Commands 32](#_Toc506375708)

[4.3.3 Scene Set Commands 33](#_Toc506375709)

[4.3.4 DMX Set Commands 34](#_Toc506375710)

[4.3.5 Configuration Discovery Queries 34](#_Toc506375711)

[4.3.6 System Queries 38](#_Toc506375712)

[4.3.7 Scene Queries 38](#_Toc506375713)

[4.3.8 Scene Set Queries 39](#_Toc506375714)

[4.3.9 System Events 40](#_Toc506375715)

[5 Enumerations 42](#_Toc506375716)

# Introduction

## Purpose and scope

This is the technical reference for the eDIN Gateway Interface for end users. This document describes version 1.08 of the Gateway Interface.

## Terminology

|  |  |
| --- | --- |
| Phrase | Description |
| DALI | Digital Addressable Lighting Interface. An industry standard physical interface/network for control of lighting equipment. It allows bi-directional data exchange, allowing its use with sensors, switches and emergency lighting as well as dimmable luminaries. |

## References

[1] BS EN 62386-102:2014 Digital addressable lighting interface. Part 102: General Requirements – Control gear.

[2] Gateway Interface Application Notes.

# Overview

The interface provides access to all runtime aspects of a working eDIN system in a consistent manner. It can *not*:

* Configure the input and output channels.
* Change/create a system’s *config* (e.g. rules, scenes, channels) except for some exceptions including:
  + scene setting,
  + Module colour palette adjustments
  + repairing DALI universes.

The interface is character based to be used over an RS232 connection, a raw TCP/IP session over Ethernet or a set of HTTP requests over Ethernet.

The interface provides 3 types of operations:

1. Commands – sent by the user to control the lighting system
2. Queries – sent by the user to poll the status of the lighting systems
3. Events/Monitors – sent by the lighting system to report a change, e.g. button pressed, scene recalled. Event reporting can be turned on & off by the user system to suit their needs.

The interface has 3 categories of operation:

1. Administration level – establishing connection with the interface and configuring how the interface operates.
   * Sets and queries if message echo feature is enabled or disabled
   * Sets and queries if long or short message acknowledgment is used by the interface
   * Sets and queries which events are reported by the interface
   * Allows users to log in and out.
2. Channel level – direct control & monitoring of discrete lighting channels and input channels/buttons/contact closures.
   * At this level the user can have complete control over the lighting system without any local control by the eDIN system[[1]](#footnote-1) or can work along side the eDIN local control, monitoring it operation and overriding it as required.
   * System level features like scenes and rules are implemented in the user system or if working along side the eDIN local control by the eDIN system.
3. System level – control & monitoring of scenes (and possibly other features) of the local eDIN system.
   * At this level the user works along side the eDIN local control. The eDIN system holds the scenes and rules and the interface provides (restricted) access to these features.

# General Syntax and Operation

The syntax and operation is designed to be used either:

* manually connected to a terminal emulation app such as HyperTerm or PuTTY, or
* programmatically via user apps such as Savant or Roomie.

## Syntax

All messages conform to the following general syntax.

$<msgId>[,<parameter>,<parameter>,…]; for commands

?<msgId>[,<parameter>,<parameter>,…]; for queries

!<msgId>[,<parameter>,<parameter>,…]; for acknowledgements, replies and events

* All messages start with either $, ? or !.
* All messages end in a semicolon character ;.
* There are no space characters in a message (except if spaces exist in the text parameters).
* The <msgId> is the name of the command, query or event.
* Each <msgId> may have 0 or more <parameter> (usually numeric decimal) values separated by the comma character ,.
* Messages are case-insensitive (except the text parameters), i.e. any combination of upper and lower case letters are accepted.
* Any extra characters sent between the end ; of one message and the start of the next are ignored. Thus it is permissible to add <CR> and <LF> characters *after* the ; end of each message.

Commands and queries *sent to* the interface can vary in format:

* they are case insensitive,
* numeric parameters can be variable length with or without leading zeros,
* all extra characters between commands (such as <CR> and <LF>) are tolerated and ignored.

Acknowledgements, replies and events *received from* the interface always:

* are in upper case,
* have fixed length numeric parameters having leading zeros as necessary,
* have variable length text parameters,
* have <CR> and <LF> appended after each message.

This makes parsing a line at a time possible, using simple case-sensitive comparisons, and parameter values are at fixed locations within the reply/event message string.

## Connections

### Raw TCP/IP

When a new connection is made via raw TCP/IP default port 26 the interface sends out 2 messages:

!GATRDY;<CR><LF>  
!VERSION,<version>;<CR><LF>

The gateway interface *automatically* closes raw TCP/IP connections that are idle for 1 hour. The gateway interface is idle if it does not receive any commands or queries or has not sent any events. To keep a connection open send the null command ($OK;) at least once an hour.

See the Application Note on Troubleshooting for further details.

### RS232

The RS232 port needs to be configured for Gateway Interface operation. This is done on the NPU web pages Settings->RS232->RS232 port service->Gateway Control

When the NPU RS232 is ready for Gateway commands it sends out 2 messages:

!GATRDY;<CR><LF>  
!VERSION,<version>;<CR><LF>

It does this when the RS232 port is newly configured for Gateway Control and when the NPU powers up and is (has previously been) configured for Gateway Control.

The RS232 serial connection is left open unless the serial port is configured to be used by another service.

To check that the connection is OK, send the null command ($OK;). If the connection is OK, the system will respond !OK;<CR><LF>. See the Application Note on Troubleshooting for further details.

### HTTP

The http connection does not support long term sessions. Each http request/response stands by itself. As a consequence no event data is published on the http connections. Only commands and queries can be used.

The http connection uses AJAX http POST requests to the URL http://<npu\_ip\_address>/gateway? (Note the terminating ? is required) using the Content-Type: text/plain.

One or more command and/or query are assembled in to a plain text request block of the AJAX POST operation. The responses to the commands and queries are returned in the plain text response data block.

Example in JQuery:

$.ajax({

type: "POST",

url: 'http://'+ipaddr+'/gateway?',

contentType: "text/plain;",

dataType: "text",

data: "$scnRecall,1;",

})

See the Application Note on Using HTTP Connections for further details and about how to support User logins.

## Message acknowledgements

For every message sent to the interface, the interface always acknowledges the message with

* !OK message if it is a recognised command or query

or

* !BAD message if it is an unrecognised command or query

The general format of an acknowledge message is:

[<echochars>]!OK[,<msgId>,<parameter>,<parameter>,…];

or

[<echochars>]!BAD;

It is possible to configure the interface to format the generated acknowledge messages to contain more or less information:

* The characters of the message sent to the interface can be echoed back in the message acknowledge, except unrecognised characters are echoed back as the period character ‘.’
* The command or query msgId and parameters can be included in the !OK (long) acknowledgement message

The echo feature is to help troubleshoot !BAD messages, and is normally turned off. Use $DBGECHO command to turn this feature on & off. The feature is always off on a new connection and off by default for AJAX requests.

The long acknowledge message includes the details of what it is acknowledging and is normally turned on. It can be useful to the user for synchronisation, to ensure the acknowledge message relates to the correct message sent to the interface. It can also be used to confirm the gateway interface has seen the command/query and parameters that you have sent, and has not interpreted them differently. Use $DBGACK command to turn this feature on & off. The feature is always on for a new connection and on by default for AJAX requests.

Further details and examples for acknowledge features are in the Application Note on Troubleshooting.

# Commands

## Administration Level Interface

Administration level commands, queries and events control and report the operation of the interface itself.

The interface can be set to monitor the operation of the system and generate event messages when changes in the system are detected. There are a number of different event categories. It is possible to enable and disable each event category independently so that only messages of interest are generated. All events are disabled when a connection is opened. (Only changes in public channels and scenes are generated – see sections on channel level and system level interfaces)

To help with troubleshooting it is possible to configure the format of the acknowledge messages.

It is possible for users to log on and log off if users have been set up in the configuration. Users are configured to have different access levels to areas defined in the configuration. If a user does not have access to a particular channel or scene, the eDIN system will behave in the same way as if that resource does not exist.

### Commands

* Null Command  
  $OK;  
  *Long-ack Format*: !OK;  
  e.g.  
  $OK;  
  !OK;<CR><LF>
* Set Long Acknowledge Mode  
  $DBGACK,<on-off>;  
  *Long-ack Format*: !OK,DBGACK,0;  
  On-Off = 0 for short acknowledges, 1 for long acknowledges  
  e.g.  
  $dbgAck,1; $dbgAck,0;  
  !OK,DBGACK,1;<CR><LF>  
  !OK;<CR><LF>
* Set Echo Mode  
  $DBGECHO,<on-off>;  
  *Long-ack Format*: !OK,DBGECHO,0;  
  On-Off = 0 for no echo, 1 for echo  
  e.g.  
  $dbgEcho,1;$dbgEc**jl**,0;$dbgEcho,0;   
  !OK,DBGECHO,1;<CR><LF>$dbgEc....!BAD;<CR><LF>  
  $dbgEcho,0!OK,DBGECHO,0;<CR><LF>
* Log On as a particular user  
  $USER,<user-name>,<encrypted-password>;  
  *Long-ack Format*: !OK,USER,text,text;  
  user-name,encrypted-password = Username and Password as appearing on Configuration Options -> Users web page  
  e.g.  
  $User,Jeff,amVmZg;  
  !OK,USER,Jeff,amVmZg;<CR><LF>(**NOTE**: For HTTP connections username and password can also be sent in HTTP request header Basic authorisation. See Applications Notes for details.)
* Log Out from current user  
  $USER;  
  *Long-ack Format*: !OK,USER;  
  e.g.  
  $user;  
  !OK,USER;<CR><LF>
* Set Event Reporting  
  $EVENTS,<on-off>;  
  *Long-ack Format*: !OK,EVENTS,0;  
  On-Off = 0 to turn all event flags off, non-zero to turn all event flags on  
  e.g.  
  $Events,1;  
  !OK,EVENTS,1;<CR><LF>
* Set Output Event Reporting  
  $EVTOUT,<on-off>;  
  *Long-ack Format*: !OK,EVTOUT,0;  
  On-Off = 0 to turn flag off, non-zero to turn flag on  
  When the flag is on all changes to public outputs generate a channel level event message, including  
   !CHANFADE, !CHANSTATE, !CHANSTOP,  
   !DALIFADE, !DALISTOP.  
  e.g.  
  $EvtOut,0;  
  !OK,EVTOUT,0;<CR><LF>
* Set Display Event Reporting  
  $EVTDIS,<on-off>;  
  *Long-ack Format*: !OK,EVTDIS,0;  
  On-Off = 0 to turn flag off, non-zero to turn flag on  
  When the flag is on all changes to public outputs generate a display event message, including  
   !BTNCOLR, !BTNTEXT.  
  e.g.  
  $EvtDis,0;  
  !OK,EVTDIS,0;<CR><LF>
* Set Status Event Reporting  
  $EVTERR,<on-off>;  
  *Long-ack Format*: !OK,EVTERR,0;  
  On-Off = 0 to turn flag off, non-zero to turn flag on  
  When the flag is on all changes to any public channel status generates a channel level event message, including  
   !CHANERR, !DALIERR, !BTNERR, !INPERR.  
  e.g.  
  $EvtErr,1;  
  !OK,EVTERR,1;<CR><LF>  
  (**NOTE**: Channel status events are not currently implemented)
* Set Scene Event Reporting  
  $EVTSCN,<on-off>;  
  *Long-ack Format*: !OK,EVTSCN,0;  
  On-Off = 0 to turn flag off, non-zero to turn flag on  
  When the flag is on all scene actions for public scenes generate a system level event message, including  
   !SCNRECALL, !SCNRECALLX,  
   !SCNRAISE, !SCNLOWER, !SCNSTOP,  
   !SCNNUDGEUP, !SCNNUDGEDN, !SCNSAVE.  
  e.g.  
  $EvtScn,1;  
  !OK,EVTSCN,1;<CR><LF>
* Set Contact and PIR Input Event Reporting  
  $EVTCIN,<on-off>;  
  *Long-ack Format*: !OK,EVTCIN,0;  
  On-Off = 0 to turn flag off, non-zero to turn flag on  
  When the flag is on all changes to public button, contact input and PIR input states generate a channel level event message, including  
   !BTNSTATE, !INPSTATE, !INPPIR  
  e.g.  
  $EvtCIn,0;  
  !OK,EVTCIN,0;<CR><LF>
* Set Analogue Input Event Reporting  
  $EVTAIN,<on-off>;  
  *Long-ack Format*: !OK,EVTAIN,0;  
  On-Off = 0 to turn flag off, non-zero to turn flag on  
  When the flag is on all changes to public analogue input states generate a channel level event message, including  
   !INPLEVEL,  
  e.g.  
  $EvtAIn,0;  
  !OK,EVTAIN,0;<CR><LF>
* Set Advanced Event Reporting  
  $EVTADV,<on-off>;  
  *Long-ack Format*: !OK,EVTADV,0;  
  On-Off = 0 to turn flag off, non-zero to turn flag on  
  When the flag is on all advanced operation events are generated, allowing the operations to be tracked in real time, such as DALI repair and Emergency testing.  
  e.g.  
  $EvtAdv,0;  
  !OK,EVTADV,0;<CR><LF>

### Queries

* request gateway version  
  ?VERSION;  
  *Long-ack Format*: !OK,VERSION;  
  *Reply: ‘Report gateway version’ (see events)*e.g.  
  ?version;  
  !OK,VERSION;<CR><LF>  
  !VERSION,01.00;<CR><LF>
* Request current user  
  ?USER;  
  *Long-ack Format*: !OK,USER;  
  *Reply: Report current user*!USER,<username>;  
  Format:!USER,text;  
  e.g.  
  ?user;  
  !OK,USER;<CR><LF>  
  !USER,Jeff;<CR><LF>
* Request Event Reporting  
  ?EVENTS;  
  *Long-ack Format*: !OK,EVENTS;  
  *Reply: set of Event flags  
  a list of all ‘Report Event Flag’ (same as individual requests)*!EVTxxx,0;<CR><LF>  
  e.g.  
  ?eVents;  
  !OK,EVENTS;<CR><LF>  
  !EVTOUT,1;<CR><LF>!EVTDIS,1;<CR><LF>!EVTERR,0;<CR><LF>!EVTSCN,1;<CR><LF>!EVTCIN,0;<CR><LF>!EVTAIN,0;<CR><LF>!EVTADV,0;<CR><LF>
* Request Output Event Reporting  
  ?EVTOUT;  
  *Long-ack Format*: !OK,EVTOUT;  
  *Reply: ‘Report Output Event Flag’*  
  !EVTOUT,0;<CR><LF>  
  e.g.  
  ?EvtOut;  
  !OK,EVTOUT;<CR><LF>  
  !EVTOUT,1;<CR><LF>
* Request Display Event Reporting  
  ?EVTDIS;  
  *Long-ack Format*: !OK,EVTDIS;  
  *Reply: ‘Report Display Event Flag’*  
  !EVTDIS,0;<CR><LF>  
  e.g.  
  ?EvtDis;  
  !OK,EVTDIS;<CR><LF>  
  !EVTDIS,1;<CR><LF>
* Request Status Event Reporting  
  ?EVTERR;  
  *Long-ack Format*: !OK,EVTERR;  
  *Reply: ‘Report Status Event Flag’*  
  !EVTERR,0;<CR><LF>  
  e.g.  
  ?evtErr;  
  !OK,EVTERR;<CR><LF>  
  !EVTERR,1;<CR><LF>
* Request Scene Event Reporting  
  ?EVTSCN;  
  *Long-ack Format*: !OK,EVTSCN;  
  *Reply: ‘Report Scene Event Flag’*  
  !EVTSCN,0;<CR><LF>  
  e.g.  
  ?EvtScn;  
  !OK,EVTSCN;<CR><LF>  
  !EVTSCN,1;<CR><LF>
* Request Contact Input Event Reporting  
  ?EVTCIN;  
  *Long-ack Format*: !OK,EVTCIN;  
  *Reply: ‘Report Contact Input Event Flag’*  
  !EVTCIN,0;<CR><LF>  
  e.g.  
  ?EvtCIn;  
  !OK,EVTCIN;<CR><LF>  
  !EVTCIN,1;<CR><LF>
* Request Analogue Input Event Reporting  
  ?EVTAIN;  
  *Long-ack Format*: !OK,EVTAIN;  
  *Reply: ‘Report Analogue Input Event Flag’*  
  !EVTAIN,0;<CR><LF>  
  e.g.  
  ?EvtAIn;  
  !OK,EVTAIN;<CR><LF>  
  !EVTAIN,1;<CR><LF>
* Request Analogue Input Event Reporting  
  ?EVTADV;  
  *Long-ack Format*: !OK,EVTADV;  
  *Reply: ‘Report Advanced Event Flag’*  
  !EVTADV,0;<CR><LF>  
  e.g.  
  ?EvtAdv;  
  !OK,EVTADV;<CR><LF>  
  !EVTADV,1;<CR><LF>

### Events:

* *Event: ‘gateway ready’*!GATRDY;  
  Sent when initial connection is made or opened
* *Event: ‘Report gateway version’*  
  !VERSION,<version>;  
  Format: !VERSION,text;  
  Sent when initial connection is made or on a query

## Channel Level Interface

A *channel* is identified with 3 items: first by the module’s MBus *address* then the module’s Mode *device code* and then the *channel number* within the module (shown as <addr>,<devcode>,<chan-num> below).

There are a few exceptions. The first is lighting channels on an *addressable* DALI universe, where the Mode *DALI channel number* is used (shown as <addr>,<devcode>,<dali-num> below). Lighting ballasts on a DALI universe are assigned to a Mode DALI channel number during configuration/commissioning. It is this Mode DALI channel number that is used by this interface.

Note: parameters labelled as <dali-id> can use the <dali-num> values or advanced DALI Broadcast and Group Ids. See the Application Note on Advanced DALI Operation for further details.

Note: For modules that have DALI Broadcast output channels, you MUST use the standard CHANxxx commands. The DALIxxx commands are only for UBC modules with addressable DALI ouputs.

The second exception is DMX zones, where zone number rather than channel number is used. Users identify a DMX zone with 3 items: the module is identified similarly by its *address* and *device code* then *zone number* identifies the DMX zone (shown as <addr>,<devcode>,<zone-num> below). See the Application Note on DMX Operation for further details.

Note: the module’s DMX universe and specified DMX zone both must be enabled when using the DMX commands and queries. This is done during configuration/commissioning and cannot be controlled through the GATEWAY interface.

The third exception is plate buttons. Users identify a button with 3 items: the plate is identified similarly by its *address* and *device code* then *button number* identifies the button (shown as <addr>,<devcode>,<btn-num> below). With Button queries it is also possible to specify the <btn-num> as ALL, when all buttons on the plate will report. See the Application Note on Understanding Channel Types for details of button numbering.   
Plates also use *channel numbers* to identify the contact inputs on the plate.

Channels, zones and buttons have an *access level* that defines how they respond to gateway operations. The numeric access level value is a set of bit-field flags. For systems with an NPU this is set by the configuration and cannot be changes through the Gateway. For all other systems, all channels (that exist) are fully accessible.

|  |  |  |  |
| --- | --- | --- | --- |
| Bit | Numeric Value | Flag | Description |
| 0 | 1 | View | If set the channel can be queried. |
| 1 | 2 | Control | If set the channel can be controlled with commands. |
| 2 | 4 | Edit | n/a |

### Output Channel Commands

These commands control the brightness or output state of channels. Only channels with the Control access flag set will be affected by these commands.

* Set (output) channel to level with fade  
  $CHANFADE,<addr>,<devcode>,<chan-num>,<level>,<fadetime(ms)>;  
  *Long-ack Format*: !OK,CHANFADE,000,000,000,000,00000000;  
  e.g.  
  $ChanFade,1,12,2,30,3000;  
  !OK,CHANFADE,001,012,002,030,00003000;<CR><LF>
* Set (relay) channel to state  
  $CHANSTATE,<addr>,<devcode>,<chan-num>,<level>;  
  *Long-ack Format*: !OK,CHANSTATE,000,000,000,000;  
  e.g.  
  $ChanSTATe,1,12,2,255;  
  !OK,CHANSTATE,001,012,002,255;<CR><LF>
* Set DALI channel to level with fade  
  $DALIFADE,<addr>,<devcode>,<dali-id>,<level>,<fadetime(ms)>;  
  *Long-ack Format*: !OK,DALIFADE,000,000,000,000,00000000;  
  Dali Id: DALI channel number <dali-num> or advanced DALI id   
  e.g.  
  $dalifade,02,17,016,255,1000;  
  !OK,DALIFADE,002,017,016,255,00001000;<CR><LF>
* Set DMX zone brightness to level with fade  
  $DMXLEVEL,<addr>,<devcode>,<zone-num>,<level>,<fadetime(ms)>;  
  *Long-ack Format*: !OK,DMXLEVEL,000,000,000,000,00000000;  
  e.g.  
  $dmxlevel,02,15,03,255,1000;  
  !OK,DMXLEVEL,002,015,003,255,00001000;<CR><LF>
* Set button colour  
  $BTNCOLR,<addr>,<devcode>,<palette-colour>;  
  *Long-ack Format*: !OK,BTNCOLR,000,000,000,000;  
  e.g.  
  $BtnColr,08,2,4,3;  
  !OK,BTNCOLR,008,002,004,003;<CR><LF>
* Set button text  
  $BTNTEXT,<addr>,<devcode>,<btn-num>,<text>;  
  *Long-ack Format*: !OK,BTNTEXT,000,000,000,text;  
  e.g.  
  $btnTeXt,63,1,8,All On;  
  !OK,BTNTEXT,063,001,008,All On;<CR><LF>
* Stop (output) channel fade  
  $CHANSTOP,<addr>,<devcode>,<chan-num>;  
  *Long-ack Format*: !OK,CHANSTOP,000,000,000;  
  e.g.  
  $chanstop,26,16,4;  
  !OK,CHANSTOP,026,016,004;<CR><LF>
* Stop DALI channel fade  
  $DALISTOP,<addr>,<devcode>,<dali-id>;  
  *Long-ack Format*: !OK,DALISTOP,000,000,000;  
  Dali Id: DALI channel number <dali-num> or advanced DALI id   
  e.g.  
  $DALIstop,4,17,12;  
  !OK,DALISTOP,004,017,012;<CR><LF>
* Stop DMX zone fade  
  $DMXLEVELSTOP,<addr>,<devcode>,<zone-num>;  
  *Long-ack Format*: !OK,DMXLEVELSTOP,000,000,000;  
  e.g.  
  $DMXLevelStop,4,15,2;  
  !OK,DMXLEVELSTOP,004,015,002;<CR><LF>

### Input Channel Commands

These commands allow you to temporarily override an input channel. Only channels with the Control access flag set will be affected by these commands and not all channel types can be overridden.

It is possible to ‘inject’ a button event that will trigger the system to behave as if the real button had been press/held/released/etc. In effect this provides remote control of plates.

* Set button state  
  $BTNSTATE,<addr>,<devcode>,<btn-num>,<btn-state>;  
  *Long-ack Format*: !OK,BTNSTATE,000,000,000,000;  
  e.g.  
  $BtnState,08,2,4,1;  
  !OK,BTNSTATE,008,002,004,001;<CR><LF>

It is possible to force a PIR channel to a particular state. This allows the channel to be set so that it can timeout to an ‘absence’ event, triggering absent rules. It allows the channel to be cleared so it can see a new ‘occupied’ event.

* Set PIR Input  
  $INPPIR,<addr>,<devcode>,<chan-num>,<pir-state>;  
  *Long-ack Format*: !OK,INPPIR,000,000,000,000;  
  e.g.  
  $inpPir,02,15,15,3;  
  !OK,INPPIR,002,015,015,003;<CR><LF>

### DMX Colour Commands

These commands control the colour in DMX zones. Only DMX zones that are enabled and have their Control access flag set will be affected by these commands. See the Application Note on DMX Operation for further information.

* Set DMX zone colour mode with cross-fade  
  Notes:  
  - the <mode> value is one of the eDIN+ colour modes, including static colour washes and continually cross-fading colour sequences.   
  $DMXMODE,<addr>,<devcode>,<zone-num>,<mode>,<fadetime(ms)>;  
  *Long-ack Format*: !OK,DMXMODE,000,000,000,000,00000000;  
  See the Enumerations Appendix for DMX mode values  
  e.g.  
  $dmxMode,02,15,03,2,1000;  
  !OK,DMXMODE,002,015,003,002,00001000;<CR><LF>
* Set DMX zone raw colour  
  Notes:  
  - the specified colour is applied to all DMX fixtures in the specified zone  
  - the specified colour is applied instantly without cross-fade  
  - the <white> component may be ignored depending on DMX fixture type  
  $DMXCOLR,<addr>,<devcode>,<zone-num>,<red>,<grn>,<blu>,<wht>;  
  *Long-ack Format*: !OK,DMXCOLR,000,000,000,000,000,000,000;  
  e.g.  
  $dmxColr,02,15,01,128,64,0,0;  
  !OK,DMXCOLR,002,015,001,128,064,000,000;<CR><LF>

### Identify Channel Commands

These commands allow you to temporarily override an output channel. Only channels with the Control access flag set will be affected by these commands and not all channel types can be overridden.

The identify channel commands allow you to temporarily override an output channel to determine the locations of the channel’s fixtures. When sent to a channel, the channel will continually ramp from its minimum level to its maximum level and back to its minimum level, until the identify command is cancelled.

Notes:

* Only one channel per module can be identified at any one time. Identifying another channel on the module will automatically cancel the previous identify.
* A channel identify will automatically cancel itself after a built-in 300 seconds (5 mins) in case you forget to cancel it. Resending the command within the 5 mins will reset the time for a further 5 mins.
* Cancel identify channel on a module  
  $SHOWOFF,<addr>,<devcode>;  
  *Long-ack Format*: !OK,SHOWOFF,000,000;  
  e.g.  
  $showOff,08,2;  
  !OK,SHOWOFF,008,002;<CR><LF>
* Identify channel  
  $SHOWCHAN,<addr>,<devcode>,<chan-num>;  
  *Long-ack Format*: !OK,SHOWCHAN,000,000,000;  
  e.g.  
  $showChan,08,2,1;  
  !OK,SHOWCHAN,008,002,001;<CR><LF>
* Identify DALI channel or fixture  
  $SHOWDALI,<addr>,<devcode>,<dali-num>;  
  *Long-ack Format*: !OK,SHOWDALI,000,000,000;  
  e.g.  
  $showDali,05,17,3;  
  !OK,SHOWDALI,005,017,003;<CR><LF>  
  Note: There is also an ‘Identify DALI Fixture’ command – see 4.2.11 Advanced DALI – Repair

### Channel State Queries

Only channels with the View access flag set will reply to these queries.

* request (output or relay) channel status  
  ?CHAN,<addr>,<devcode>,<chan-num>;  
  *Long-ack Format*: !OK,CHAN,000,000,00;  
  *Reply: ‘Report channel status’*!CHAN,<addr>,<devcode>,<chan-num>,<status-code>,<level>,<power>,<wattage>;  
  Format:!CHAN,000,000,000,000,000,000,00000;  
  Status Code = 0 (OK). see Emulation Appendix for values  
  Power = percentage of maximum wattage (0-100)  
  Level = level (0-255) or state (0-255)  
  Wattage = wattage at maximum level/power in Watts  
  e.g.  
  ?chan,1,12,2;  
  !OK,CHAN,001,012,002;<CR><LF>  
  !CHAN,001,012,002,000,030,099,00100;<CR><LF>
* request DALI channel status  
  ?DALI,<addr>,<devcode>,<dali-num>;  
  *Long-ack Format*: !OK,DALI,000,000,00;  
  *Reply: ‘Report DALI channel status’*!DALI,<addr>,<devcode>,<dali-num>,<status-code>,<level>,<power>,<wattage>;  
  Format:!DALI,000,000,000,000,000,000,00000;  
  Status Code = 0 (OK). see Emulation Appendix for values  
  Power = percentage of maximum wattage (0-100)  
  Level = level (0-255) or state (0-255)  
  Wattage = total wattage of all ballasts in the DALI channel at  
   maximum level/power in Watts   
  e.g.  
  ?dali,1,17,2;  
  !OK,DALI,001,017,002;<CR><LF>  
  !DALI,001,017,002,000,254,099,02500;<CR><LF>
* request DMX zone status  
  ?DMX,<addr>,<devcode>,<zone-num>;  
  *Long-ack Format*: !OK,DMX,000,000,00;  
  *Reply: ‘Report DALI channel status’*!DMX,<addr>,<devcode>,<zone-num>,<status-code>,<mode>,<level>,<power>,  
   <wattage>;  
  Format:!DMX,000,000,000,000,000,000,000,00000;  
  Status Code = 0 (OK). see Emulation Appendix for values  
  Power = percentage of maximum wattage (0-100)  
  Mode = eDIN DMX mode (0-127) see Emulation Appendix  
  Level = brightness level (0-255)  
  Wattage = total wattage of all DMX fixtures in the zone at  
   maximum level/power in Watts  
  e.g.  
  ?dmx,1,15,2;  
  !OK,DMX,001,015,002;<CR><LF>  
  !DMX,001,015,002,000,005,255,033,02500;<CR><LF>
* request button status  
  ?BTN,<addr>,<devcode>,ALL;  
  ?BTN,<addr>,<devcode>,<btn-num>;  
  *Long-ack Format*: !OK,BTN,000,000,00;  
  *Reply: one or more ‘Report button status’*!BTN,<addr>,<devcode>,<btn-num>,<status-code>,<state>;  
  Format:!BTN,000,000,000,000,000;  
  Status Code = 0 (OK). see Emulation Appendix for values  
  State = 0 (Rel-off) or 1 (Press-on). see Emulation Appendix for values  
  e.g.  
  ?btn,15,1,2;  
  !OK,BTN,015,001,02;<CR><LF>  
  !BTN,015,001,002,000,000;<CR><LF>
* request button colour  
  ?BTNCOLR,<addr>,<devcode>,ALL  
  ?BTNCOLR,<addr>,<devcode>,<btn-num>;  
  *Long-ack Format*: !OK,BTNCOLR,000,000,000;  
  *Reply: one or more ‘Report button colour’*!BTNCOLR,<addr>,<devcode>,<btn-num>,<palette-colour>;  
  Format:!BTNCOLR,000,000,000,000;  
  Palette Colour = 0-255. See Emulation Appendix for values  
  e.g.  
  ?btncolr,15,1,2;  
  !OK,BTNCOLR,015,001,002;<CR><LF>  
  !BTNCOLR,015,001,002,003;<CR><LF>
* request button text  
  ?BTNTEXT,<addr>,<devcode>,ALL;  
  ?BTNTEXT,<addr>,<devcode>,<btn-num>;  
  *Long-ack Format*: !OK,BTNTEXT,000,000,000;  
  *Reply: one or more ‘Report button text’*!BTNTEXT,<addr>,<devcode>,<btn-num>,<text>;  
  Format:!BTNTEXT,000,000,000,text;   
  e.g.  
  ?btntext,15,1,2;  
  !OK,BTNTEXT,015,001,002;<CR><LF>  
  !BTNTEXT,015,001,002,All On;<CR><LF>
* request input channel status  
  ?INP,<addr>,<devcode>,<chan-num>  
  *Long-ack Format*: !OK,INP,000,000,000;  
  *Reply: ‘Report input channel status’*!INP,<addr>,<devcode>,<chan-num>,<status-code>,<level>;  
  Format:!INP,000,000,000,000,000;  
  Status Code = 0 (OK). see Emulation Appendix for values  
  Level = level (0-255) or state 0 (Rel-off) or 1 (Press-on), depending on channel type  
  e.g.  
  ?inp,15,21,2;  
  !OK,INP,015,021,002;<CR><LF>  
  !INP,015,021,002,000,000;<CR><LF>

### Channel Access Queries

These queries can be used to establish if a channel will respond to a command or query. Channels that do not exist will not reply to these queries, otherwise all channels will reply even if they are not set to reply to standard queries.

* request (output or relay) channel access  
  ?CHANACCESS,<addr>,<devcode>,<chan-num>;  
  *Long-ack Format*: !OK,CHANACCESS,000,000,00;  
  *Reply: ‘Report channel access’*!CHANACCESS,<addr>,<devcode>,<chan-num>,<access>;  
  Format:!CHANACCESS,000,000,000,00;  
  Access = bit-field flags 1 (viewable) + 2 (controllable) + 4 (editable).  
  e.g.  
  ?chanAccess,1,12,2;  
  !OK,CHANACCESS,001,012,002;<CR><LF>  
  !CHANACCESS,001,012,002,07;<CR><LF>
* request DALI channel access  
  ?DALIACCESS,<addr>,<devcode>,<dali-num>;  
  *Long-ack Format*: !OK,DALIACCESS,000,000,00;  
  *Reply: ‘Report DALI channel access’*!DALIACCESS,<addr>,<devcode>,<dali-num>,<access>;  
  Format:!DALIACCESS,000,000,000,00;  
  Access = bit-field flags 1 (viewable) + 2 (controllable) + 4 (editable).  
  e.g.  
  ?DALIaccess,01,0017,2;  
  !OK,DALIACCESS,001,017,002;<CR><LF>  
  !DALIACCESS,001,017,002,07;<CR><LF>
* request DMX zone access  
  ?DMXACCESS,<addr>,<devcode>,<zone-num>;  
  *Long-ack Format*: !OK,DMXACCESS,000,000,00;  
  *Reply: ‘Report DMX zone access’*!DMXACCESS,<addr>,<devcode>,<zone-num>,<access>;  
  Format:!DMXACCESS,000,000,000,00;  
  Access = bit-field flags 1 (viewable) + 2 (controllable) + 4 (editable).  
  e.g.  
  ?DMXaccess,01,0015,2;  
  !OK,DMXACCESS,001,015,002;<CR><LF>  
  !DMXACCESS,001,015,002,07;<CR><LF>
* request button access  
  ?BTNACCESS,<addr>,<devcode>,<btn-num>;  
  *Long-ack Format*: !OK,BTNACCESS,000,000,000;  
  *Reply: ‘Report DALI channel access’*!BTNACCESS,<addr>,<devcode>,<btn-num>,<access>;  
  Format:!BTNACCESS,000,000,000,00;  
  Access = bit-field flags 1 (viewable) + 2 (controllable) + 4 (editable).  
  e.g.  
  ?BtnAccess,01,2,2;  
  !OK,BTNACCESS,001,002,002;<CR><LF>  
  !BTNACCESS,001,002,002,07;<CR><LF>
* request input channel access  
  ?INPACCESS,<addr>,<devcode>,<chan-num>;  
  *Long-ack Format*: !OK,INPACCESS,000,000,000;  
  *Reply: ‘Report input channel access’*!INPACCESS,<addr>,<devcode>,<chan-num>,<access>;  
  Format:!INPACCESS,000,000,00,000;  
  Access = bit-field flags 1 (viewable) + 2 (controllable) + 4 (editable).  
  e.g.  
  ?inpAccess,01,2,2;  
  !OK,INPACCESS,001,002,002;<CR><LF>  
  !INPACCESS,001,002,002,07;<CR><LF>

### General Channel Events

Only channels with View access flag set will have these events reported on the Gateway interface. Note: Event reporting is also controlled via Administration Level commands – see Section 4.1 Administration Level Interface.

* *Event: ‘Change in (output) channel level with fade’*!CHANFADE,<addr>,<devcode>,<chan-num>,<level>,<fadetime(ms)>;  
  Format:!CHANFADE,000,000,000,000,00000000;  
  e.g.  
  !CHANFADE,002,021,006,255,00003000;<CR><LF>
* *Event: ‘Change in (relay) channel state’*!CHANSTATE,<addr>,<devcode>,<chan-num>,<state>;  
  Format:!CHANSTATE,000,000,000,000;  
  e.g.  
  !CHANSTATE,002,021,006,255;<CR><LF>
* *Event: ‘Stop in (output) channel’*!CHANSTOP,<addr>,<devcode>,<chan-num>;  
  Format:!CHANSTOP,000,000,000;  
  e.g.  
  !CHANSTOP,002,021,006;<CR><LF>
* *Event: ‘Change in DALI channel level with fade’*!DALIFADE,<addr>,<devcode>,<dali-id>,<level>,<fadetime(ms)>;  
  Dali Id: DALI channel number <dali-num> or advanced DALI id   
  Format:!DALIFADE,000,000,000,000,00000000;  
  e.g.  
  !DALIFADE,002,017,016,255,00001000;<CR><LF>
* *Event: ‘Stop in DALI channel’*!DALISTOP,<addr>,<devcode>,<dali-id>;  
  Dali Id: DALI channel number <dali-num> or advanced DALI id   
  Format:!DALISTOP,000,000,000;  
  e.g.  
  !DALISTOP,002,017,016;<CR><LF>
* *Event: ‘Change in DMX zone brightness’*!DMXLEVEL,<addr>,<devcode>,<zone-num>,<level>,<fadetime(ms)>;  
  Format:!DMXLEVEL,000,000,000,000,00000000;  
  e.g.  
  !DMXLEVEL,002,015,006,255,00005000;<CR><LF>
* *Event: ‘Stop DMX zone brightness Fade’*!DMXLEVELSTOP,<addr>,<devcode>,<chan-num>;  
  Format:!DMXLEVELSTOP,000,000,000;  
  e.g.  
  !DMXLEVELSTOP,002,015,006;<CR><LF>
* *Event: ‘Change in Button colour’*!BTNCOLR,<addr>,<devcode>,<btn-num>,<palette-colour>;Format:!BTNCOLR,000,000,000,000;  
  e.g.  
  !BTNCOLR,063,001,008,015;<CR><LF>
* *Event: ‘Change in Button text’*!BTNTEXT,<addr>,<devcode>,<btn-num>,<text>;Format:!BTNTEXT,000,000,000,text;  
  e.g.  
  !BTNTEXT,063,001,008,All On;<CR><LF>
* *Event: ‘Change in Button state’*!BTNSTATE,<addr>,<devcode>,<btn-num>,<new-state>;Format:!BTNSTATE,000,000,000,000;  
  e.g.  
  !BTNSTATE,063,001,008,000;<CR><LF>
* *Event: ‘Change in PIR input state’*!INPPIR,<addr>,<devcode>,<chan-num>,<new-state>;Format:!INPPIR,000,000,000,000;  
  e.g.  
  !INPSTATE,005,021,008,000;<CR><LF>
* *Event: ‘Change in switched input’*!INPSTATE,<addr>,<devcode>,<chan-num>,<new-state>;Format:!INPSTATE,000,000,000,000;  
  e.g.  
  !INPSTATE,005,021,008,000;<CR><LF>
* *Event: ‘Change in analogue input’*!INPLEVEL,<addr>,<devcode>,<chan-num>,<new-level>;Format:!INPLEVEL,000,000,000,000;  
  e.g.  
  !INPLEVEL,005,021,008,255;<CR><LF>

### DMX Colour Events

Only zones with View access flag set will have these events reported on the Gateway interface. Note: Event reporting is also controlled via Administration Level commands – see Section 4.1 Administration Level Interface.

* *Event: ‘Change in DMX zone colour mode’*  
  !DMXMODE,<addr>,<devcode>,<zone-num>,<colour-mode>;  
  Format:!DMXMODE,000,000,000,000;  
  e.g.  
  !DMXMODE,002,015,006,001;<CR><LF>
* *Event: ‘Change in DMX zone raw colour’*!DMXCOLR,<addr>,<devcode>,<zone-num>,<red>,<grn>,<blu>,<wht>;Format:!DMXCOLR,000,000,000,000,000,000,000;  
  e.g.  
  !DMXCOLR,008,015,004,255,255,255,0;<CR><LF>

### Channel Status Events

Only channels with View access flag set will have these events reported on the Gateway interface. Note: Event reporting is also controlled via Administration Level commands – see Section 4.1 Administration Level Interface.

Status codes values are given in the Enumerations Appendix section.

* *Event: ‘Change in Module status’*Notes:  
  - module errors are also reported on individual channels of the module  
  !MODUERR,<addr>,<devcode>,<new status-code>;  
  Format:!MODUERR,000,000,000;  
  e.g.  
  !MODUERR,002,021,001;<CR><LF>
* *Event: ‘Change in (output or relay) channel status’*  
  !CHANERR,<addr>,<devcode>,<chan-num>,<new status-code>;  
  Format:!CHANERR,000,000,000,000;  
  e.g.  
  !CHANERR,002,021,006,001;<CR><LF>
* *Event: ‘Change in DALI channel status’*  
  !DALIERR,<addr>,<devcode>,<dali-num>,<new status-code>;  
  Format:!DALIERR,000,000,000,000;  
  e.g.  
  !DALIERR,002,017,016,002;<CR><LF>
* *Event: ‘Change in DALI fixture status’*  
  !DALIERR,<addr>,<devcode>,<dali-fixture>,<new status-code>;  
  Format:!DALIERR,000,000,F00,000;See Advanced DALI section below  
  e.g.  
  !DALIERR,002,017,F16,025;<CR><LF>
* *Event: ‘Change in DMX zone status’*  
  !DMXERR,<addr>,<devcode>,<zone-num>,<new status-code>;  
  Format:!DMXERR,000,000,000,000;  
  e.g.  
  !DMXERR,002,015,001,002;<CR><LF>
* *Event: ‘Change in button status’*  
  !BTNERR,<addr>,<devcode>,<btn-num>,<new status-code>;Format:!BTNERR,000,000,000,000;  
  e.g.  
  !BTNERR,063,001,008,000;<CR><LF>
* *Event: ‘Change in input channel status’*  
  !INPERR,<addr>,<devcode>,<chan-num>,<new-status>;Format:!INPERR,000,000,000,000;  
  e.g.  
  !INPERR,005,021,008,000;<CR><LF>

### Advanced DALI

See the Application Note on Advanced Dali Operation for further information.

**<dali-num> Parameters**

Advanced DALI identifiers cannot be used with this parameter. It must be the usual Mode DALI channel number.

**<dali-id> Parameters**

Any Mode DALI channel number can be replaced in commands, queries and events with an advanced DALI identifier:

* BST – DALI Broadcast – all channels
* G00 to G15 – DALI Group – commissioned DALI groups
* F00 to F63 – DALI Fixtures (short address)

Note the behaviour when using the advanced identifiers depends on the operating mode and the command/query/event.

* F00 to F63 only work when in Channel mode
* G00 to G15 only work when in Group and Channel modes
* BST, Gxx & Fxx identifiers will be acknowledged as BST, Gxx & Fxx, but converted to a Mode DALI channel number in replies and events where appropriate:
  + In Broadcast mode, BST -> Mode DALI channel 001.
  + In Group mode, G00 to G15 -> Mode DALI channels 001 to 016.
  + In Channel mode, F00 to F63 -> Mode DALI channel 001 to 064.
* The following queries only reply to Mode DALI channels. For example, using BST in Group or Channel modes will not generate a reply for these queries.
  + ?DALI, ?DALIACCESS

For example,

* In Broadcast mode  
  $dalifade,02,17,BST,255,1000;  
  !OK,DALIFADE,002,17,BST,255,00001000;<CR><LF>  
  !DALIFADE,002,17,**001**,255,00001000;<CR><LF>
* In Group mode  
  $daliStop,02,17,BST;  
  !OK,DALISTOP,002,17,BST;<CR><LF>  
  !DALISTOP,002,17,**BST**;<CR><LF>  
    
  $dalifade,02,17,G00,255,1000;  
  !OK,DALIFADE,002,17,G00,255,00001000;<CR><LF>  
  !DALIFADE,002,17,**001**,255,00001000;<CR><LF>  
    
  ?DALI,002,17,BST;  
  !OK,DALI,002,17,BST;<CR><LF> (to acknowledge the query)  
  no response to query

**<dali-adv> Parameters**

DALI Emergency Lighting support commands use the <dali-adv> parameter. This means that the advanced DALI identifier BST, Gxx & Fxx are used. For these advanced commands the usual Mode DALI channel number has no meaning and so is not allowed.

**<dali-fixture> Parameters**

Some Advanced DALI commands, query and events use the <dali-fixture> parameter. This means that the advanced DALI identifier Fxx must be used to specify a specific fixture. All other identifiers do not have meaning in this context and so are not allowed.

### Advanced DALI – Errors & Repair

See the Application Note on Advanced Dali Operation for further information.

This query is used to determine the fixture status of a faulty DALI fixture.

* request DALI channel fixture status  
  ?DALIERR,<addr>,<devcode>,<dali-fixture>;  
  *Long-ack Format*: !OK,DALIERR,000,F00,000;  
  *Reply: ‘Report DALI channel fixture status’*!DALIERR,<addr>,<devcode>,<fixture>,<new-status>;  
  Format:!DALIERR,000,000,F00,000;  
  Status Code = 0 (OK); 25 (lamp failure), 26 (missing fixture) – see Enumeration Appendix Section for full code list  
  e.g.  
  ?daliErr,1,17,f1;  
  !OK,DALIERR,001,017,F01;<CR><LF>  
  !DALIERR,001,017,F01,025;<CR><LF>

These commands are used to repair a DALI universe when fixtures have been replaced and the module is reporting status error 22 - DALI Commissioning problem

* Start a search to find existing and missing DALI fixtures  
  $DALISCAN,<addr>,<devcode>;  
  *Long-ack Format*: !OK,DALISCAN,000,000;  
  e.g.  
  $daliScan,1,17;  
  !OK,DALISCAN,001,017;<CR><LF>
* Query the search status. Use this to check on the progress of the scan.  
  ?DALISCAN,<addr>,<devcode>;  
  *Long-ack Format*: !OK,DALISCAN,000,000;  
  *Reply: ‘Report DALI scan status’*!DALISCAN,<addr>,<devcode>,<scan-status>,<num-of-fixtures>;  
  Format:!DALISCAN,000,000,00,000;  
  Status = 0 (idle), 1 (search done), 2 (searching), 3 (programming), 4 (error)  
  e.g.  
  ?daliScan,1,17;  
  !OK,DALISCAN,001,017;<CR><LF>  
  !DALISCAN,001,017,1,24;<CR><LF>
* End a repair session. This must be sent as the last command of a repair session, when no more scan or repair commands and queries are going to be sent. This resets the DALI universe status and restarts the runtime checking.  
  $DALIDONE;  
  *Long-ack Format*: !OK,DALIDONE;  
  e.g.  
  $daliDone;  
  !OK,DALIDONE;<CR><LF>
* Query the DALI fixtures. Use this when a scan is complete to get the results of the scan, and to find out what to repair.  
  ?DALIFIX,<addr>,<devcode>;  
  *Long-ack Format*: !OK,DALIFIX,000,000;  
  *Reply: list of fixtures in the specified DALI universe  
  zero or more ‘Report DALI Fixture’*!DALIFIX,<addr>,<devcode>,<fixture>,<long-addr>,<groups>,  
   <device-type>,<fixture-status>;  
  !DALIEND,<addr>,<devcode>;  
  Format:!DALIFIX,000,000,F00,00000000,00000,000,0;  
  Format:!DALIEND,000,000;  
  fixture = DALI fixture short address 0-63  
  long-addr = DALI fixture 24-bit long or random address  
  groups = DALI fixture 16-bit groups bit-field  
  device-type = DALI fixture 8-bit device type  
  fixture-status = 0 = OK, 2 = missing, 5 = new – see Section 5 for full code list  
  e.g.  
  $daliFix,1,17;  
  !OK,DALIFIX,001,017;<CR><LF>  
  !DALIFIX,001,017,F00,04905615,00064,000,0;<CR><LF>  
  !DALIFIX,001,017,F01,53195828,00032,000,1;<CR><LF>  
  !DALIFIX,001,017,F01,14096720,00002,000,5;<CR><LF>  
  !DALIEND,001,017;<CR><LF>
* Repair a DALI fixture. This reprograms a new fixture to replace a missing fixture, and updates the list of known fixtures in the configuration.  
  $DALIREPAIR,<addr>,<devcode>,<missing-fixture>,<new-fixture>;  
  *Long-ack Format*: !OK,DALIREPAIR,000,000,F00,F00;  
  *Reply: list of fixtures that have been changed during the repair  
  zero, one or two ‘Report DALI Fixture’*!DALIFIX,<addr>,<devcode>,<fixture>,<long-addr>,<groups>,  
   <device-type>,<fixture-status>;  
  Format: see above  
  e.g.  
  $dalirepair,1,17,f0,f01;  
  !OK,DALIREPAIR,001,017,F00,F01;<CR><LF>  
  !DALIFIX,001,017,F00,14096720,00064,000,0;<CR><LF>  
  !DALIFIX,001,017,F01,04905615,00000,000,0;<CR><LF>
* Import a DALI fixture. This adds a new fixture in to the list of known fixtures in the configuration, or removes a missing fixture from the known list. This command does not reprogram the actual fixture.  
  $DALIACCEPT,<addr>,<devcode>,<new or missing fixture>;  
  *Long-ack Format*: !OK,DALIACCEPT,000,000,F00;  
  *Reply: fixture that has been imported in to the configuration  
  zero or one ‘Report DALI Fixture’*!DALIFIX,<addr>,<devcode>,<fixture>,<long-addr>,<groups>,  
   <device-type>,<fixture-status>;  
  Format: see above  
  e.g.  
  $daliaccept,1,17,f1;  
  !OK,DALIACCEPT,001,017,F01;<CR><LF>  
  !DALIFIX,001,017,F01,14096720,00002,000,0;<CR><LF>

See 4.2.4 Identify Channel Commands for further information on these Identifyxxx commands

* Identify DALI fixture. Flash an individual DALI fixture to find out its physical location.  
  $SHOWDALI,<addr>,<devcode>,<fixture>;  
  *Long-ack Format*: !OK,SHOWDALI,000,000,F00;  
  e.g.  
  $showDali,05,17,F3;  
  !OK,SHOWDALI,005,017,F03;<CR><LF>
* Cancel identify DALI fixture on a module. Stop flashing the fixture.  
  $SHOWOFF,<addr>,<devcode>;  
  *Long-ack Format*: !OK,SHOWOFF,000,000;  
  e.g.  
  $showOff,08,2;  
  !OK,SHOWOFF,008,002;<CR><LF>

### Advanced DALI – Emergency Lighting

See the Application Note on Advanced Dali Operation and to the DALI standard for further information.

These commands are used to perform DALI Emergency Lighting Fixture testing.

* Start a functional test (dali cmd 227)  
  $EMFUNC,<addr>,<devcode>,<dali-fixture>;  
  *Long-ack Format*: !OK,EMFUNC,000,000,BST/G00/F00;  
  e.g.  
  $emFunc,1,17,g0;  
  !OK,EMFUNC,001,017,G00;<CR><LF>
* Start a duration test (dali cmd 228)  
  $EMDURA,<addr>,<devcode>,<dali-fixture>;  
  *Long-ack Format*: !OK,EMDURA,000,000,BST/G00/F00;  
  e.g.  
  $emDura,1,17,f23;  
  !OK,EMDURA,001,017,F23;<CR><LF>
* Stop a test (dali cmd 229)  
  $EMSTOP,<addr>,<devcode>,<dali-fixture>;  
  *Long-ack Format*: !OK,EMSTOP,000,000,BST/G00/F00;  
  e.g.  
  $emStop,1,17,bst;  
  !OK,EMSTOP,001,017,BST;<CR><LF>
* Put fixture in to or out of Inhibit mode (dali cmd 225 / 226)  
  $EMINHIBIT,<addr>,<devcode>,<dali-fixture>,<on-off>;  
  *Long-ack Format*: !OK,EMINHIBIT,000,000,BST/G00/F00,ON/OFF;  
  e.g.  
  $emInhibit,1,17,g15,on;  
  !OK,EMDURA,001,017,G15,ON;<CR><LF>  
  $emInhibit,1,17,g15,off;  
  !OK,EMDURA,001,017,G15,OFF;<CR><LF>
* Put fixture in to Rest mode (dali cmd 224)  
  $EMREST,<addr>,<devcode>,<dali-fixture>;  
  *Long-ack Format*: !OK,EMDURA,000,000,BST/G00/F00;  
  e.g.  
  $emDura,1,17,f23;  
  !OK,EMDURA,001,017,F23;<CR><LF>
* Identify a DALI EM fixture (dali cmd 240)  
  $EMSHOW,<addr>,<devcode>,<dali-fixture>;  
  *Long-ack Format*: !OK,EMSHOW,000,000,BST/G00/F00;  
  e.g.  
  $emShow,1,17,bst;  
  !OK,EMSHOW,001,017,BST;<CR><LF>  
  Note: This is not the same as the ‘Identify DALI Fixture’ command. This command asks the EM fixture to identify itself for 10 secs. It does not need a $SHOWOFF command or equivalent.
* Query a fixture for test results (dali cmd 253 / 252 / 243)  
  ?EMTEST,<addr>,<devcode>,<dali-fixture>;  
  *Long-ack Format*: !OK,EMTEST,000,000,BST/G00/F00;  
  *Reply: ‘Report EM Test Results*!EMTEST,<addr>,<devcode>,<fixture>,<em-status>,<em-failure>,  
   <em-duration>;  
  Format:!EMTEST,000,000,F00,000,000,000;  
  em-status = 8-bit result of DALI ‘QUERY EM STATUS' (DALI cmd 253)  
  em-failure = 8-bit result of DALI ‘QUERY FAILURE STATUS' (DALI cmd 252)  
  em-duration = 8-bit result of DALI ‘QUERY DURATION TEST RESULT' (DALI cmd 243)  
  e.g.  
  ?emTest,1,17,f23;  
  !OK,EMTEST,001,017,F23;<CR><LF>  
  !EMTEST,001,017,F23,000,004,000;<CR><LF>

## System Level Interface

This interface is only active if there is a local controller holding the scene information. Users identify scenes with its *scene number*.

Scenes have an *access level* that defines how they respond to gateway operations. The numeric access level value is a set of bit-field flags. This is fixed by the configuration and cannot be changes through the Gateway.

|  |  |  |  |
| --- | --- | --- | --- |
| Bit | Numeric Value | Flag | Description |
| 0 | 1 | View | If set the scene can be queried. |
| 1 | 2 | Control | If set the scene can be controlled with commands. |
| 2 | 4 | Edit | If set the scene can be edited / adjusted with scene setting operations. |

### Scene Recall Commands

* Recall scene action.  
  Make the scene active at maximum level and set the channels to the levels defined in the scene, using the scene’s fade time defined in the configuration.  
  $SCNRECALL,<scn-num>;  
  *Long-ack Format*: !OK,SCNRECALL,00000;  
  e.g.  
  $scnrecall,8;  
  !OK,SCNRECALL,00008;<CR><LF>
* Recall scene to off (level = 0) action.  
  This makes the scene inactive, sets the scene level to zero and set the channels defined in the scene to level = 0, using the scene’s fade time defined in the configuration.  
  $SCNOFF,<scn-num>;  
  *Long-ack Format*: !OK,SCNOFF,00000;  
  e.g.  
  $scnOff,8;  
  !OK,SCNOFF,00008;<CR><LF>
* Recall scene action with explicit values.  
  Perform a scene recall action using the explicit parameters, rather than the default values in the configuration.  
  Note: recalling the scene to level = 0 is the same as turning the scene off and making it inactive.  
  Note: this command specifies the scene level not channel levels. If a channel is defined in the scene at say 70%, and you recall the scene at 50% (level=128), then the channel will be set to 70% x 50% = 35%.  
  $SCNRECALLX,<scn-num>,<scn-level>,<fadetime(ms)>;  
  *Long-ack Format*: !OK,SCNRECALLX,00000,000,00000000;  
  Scene Level: 0-255. 0 = off, 255 = max (100%).  
  e.g.  
  $scnrecallx,8,255,60000;  
  !OK,SCNRECALLX,00008,255,00060000;<CR><LF>
* Recall scene with *fast* recall time action.  
  This recalls the scene at maximum level using the *fast* fade time. The *fast* fade time is defined in the configuration as a global setting.  
  This command is used to immediately turn a scene on that has a very long fade time. For example, a scene may be set to fade slowly over half an hour, but occasionally you need the lights to come on straight away, i.e. fade to max over a second or so.  
  $SCNFAST,<scn-num>;  
  *Long-ack Format*: !OK,SCNFAST,00000;  
  e.g.  
  $scnFast,8;  
  !OK,SCNFAST,00008;<CR><LF>
* Recall On/Off scene action.  
  (Toggle between $SCNRECALL and $SCNOFF actions).  
  This command toggles a scene on and off. It resets channel levels that may be been adjusted with e.g. SCNRAISE/SCNLOWER/SCNRAMP, SCNNUDGEUP/SCNNUDGEDN or SCNRECALLX commands back to the scene definition. In effect, it makes scene and channel adjustments temporary. The scene is set off if the scene is active, or is recalled to max if the scene is inactive.  
  $SCNONOFF,<scn-num>;  
  *Long-ack Format*: !OK,SCNONOFF,00000;  
  e.g.  
  $scnOnOff,8;  
  !OK,SCNONOFF,00008;<CR><LF>
* Recall scene with toggle action.  
  (Toggles between $SCNBACKON and $SCNOFF actions).  
  This command is similar to $ScnOnOff but saves and restores scene and channel levels. This makes scene and channel levels *semi*-permanent when they have been adjusted via SCNRAISE/SCNLOWER/SCNRAMP, SCNNUDGEUP/SCNNUDGEDN or SCNRECALLX commands.  
  $SCNTOGGLE,<scn-num>;  
  *Long-ack Format*: !OK,SCNTOGGLE,00000;  
  e.g.  
  $scntoggle,8;  
  !OK,SCNTOGGLE,00008;<CR><LF>
* Recall scene with previous (toggle) level.  
  This is a scene recall action, but recalls the scene and channels to their previously saved level. Current scene and channel levels are always saved during every SCNOFF or any *recall-to-level-zero* action. This command will restore them and is half of the SCNTOGGLE command.  
  $SCNBACKON,<scn-num>;  
  *Long-ack Format*: !OK,SCNBACKON,00000;  
  e.g.  
  $scnBackOn,8;  
  !OK,SCNBACKON,00008;<CR><LF>

### Scene Adjustment Commands

These commands adjust the scene level and the values of the channels and objects defined in the scene. For the most part, these are simple channel levels, but a scene may contain other objects that are invisible to the GATEWAY interface, but are still affected by these commands. In other words these commands may have unintended side effects. This will depend on how the configuration is built.

Note, also, that the effect that these commands have on scene state (where it is active or inactive) is also complicated and also depends on how the scene is configured.

If you simply want to change channel & scene levels, then use the SCNRECALLX command. If you want to mimic the action of a plate button, then these commands may be useful.

* Raise scene action – start dimming up.  
  This starts a dim-up operation. Channels defined in the scene will start to increase their channel levels. Channels that are already off (level=0) are unaffected. The rate that the levels increase is set by a global configuration setting.  
  $SCNRAISE,<scn-num>;  
  *Long-ack Format*: !OK,SCNRAISE,00000;  
  e.g.  
  $scnraise,8;  
  !OK,SCNRAISE,00008;<CR><LF>
* Lower scene action– start dimming down  
  This starts a dim-down operation. Channels defined in the scene will start to decrease their channel levels, to a minimum level that is set by a global configuration setting. Channels that are already off (level=0) are unaffected. The rate that the levels decrease is set by the global configuration setting that determines the SCNRAISE rate.  
  $SCNLOWER,<scn-num>;  
  *Long-ack Format*: !OK,SCNLOWER,00000;  
  e.g.  
  $scnlower,8;  
  !OK,SCNLOWER,00008;<CR><LF>
* Stop scene dimming - stop raise, lower and ramp actions  
  This stops any dimming action on all channels defined in the scene. $SCNSTOP,<scn-num>;  
  *Long-ack Format*: !OK,SCNSTOP,00000;  
  e.g.  
  $scnSTOp,8;  
  !OK,SCNSTOP,00008;<CR><LF>
* Ramp scene action – start dimming Toggle between Raise and Lower actions  
  This command will alternate between performing a SCNLOWER action and a SCNRAISE action, depending on the current scene state.  
  $SCNRAMP,<scn-num>;  
  *Long-ack Format*: !OK,SCNRAMP,00000;  
  e.g.  
  $scnramp,8;  
  !OK,SCNRAMP,00008;<CR><LF>
* Nudge scene up action - raise scene level by small amount  
  This dims-up by a fixed step size. Channels defined in the scene increase their channel levels by the fixed step. Channels that are already off (level=0) are unaffected. The step size is defined by a global configuration setting. The rate that the levels increase is set by the global configuration setting that defines the SCNRAISE rate.  
  $SCNNUDGEUP,<scn-num>;  
  *Long-ack Format*: !OK,SCNNUDGEUP,00000;  
  e.g.  
  $scnNudgeUp,8;  
  !OK,SCNNUDGEUP,00008;<CR><LF>
* Nudge scene down action - lower scene level by small amount  
  This dims-downs by a fixed step size. Channels defined in the scene decrease their channel levels by the fixed step, but are cupped to a minimum level that is set by the global configuration setting that sets the SCNLOWER minimum. Channels that are already off (level=0) are unaffected. The step size and rate of change uses the same global configuration settings as SCNNUDGEUP.  
  $SCNNUDGEDN,<scn-num>;  
  *Long-ack Format*: !OK,SCNNUDGEDN,00000;  
  e.g.  
  $scnNudgeDn,8;  
  !OK,SCNNUDGEDN,00008;<CR><LF>

### Scene Set Commands

There are 2 ways to permanently change the channel levels defined in a scene: the *live* method using SCNSAVE and the *offline* method using SCNSET. Further details can be found in the Application note on Scene Setting.

* Save scene levels action – permanently reprogram the channel levels of a scene to the current channel levels   
  $SCNSAVE,<scn-num>;  
  *Long-ack Format*: !OK,SCNSAVE,00000;  
  e.g.  
  $scnSave,8;  
  !OK,SCNSAVE,00008;<CR><LF>
* Define a scene (time limited)  
  $SCNSET,<scn-num>;  
  *Long-ack Format*: !OK,SCNSET,00000;  
    
  *followed by zero or more ‘Set Scene Item’  
    
  (optional)*$SCNFADE,<scn-num>,<fadetime(ms)>; or  
  *Long-ack Format*: !OK,SCNFADE,00000,00000000;  
    
  $SCNCHAN,<scn-num>,<addr>,<devcode>,<chan-num>,<level>; or  
  *Long-ack Format*: !OK,SCNCHAN,00000,000,00,000,000;  
    
  $SCNDALI,<scn-num>,<addr>,<devcode>,<chan-num>,<level>; or  
  *Long-ack Format*: !OK,SCNDALI,00000,000,00,000,000;  
    
  $SCNDMX,<scn-num>,<addr>,<devcode>,<zone-num>,<level>; or  
  *Long-ack Format*: !OK,SCNDMX,00000,000,00,000,000;  
    
  *followed by ‘End of scene definition’*  
  $SCNEND,<scn-num>; *Long-ack Format*: !OK,SCNEND,00000;  
  Only editable scenes (Edit access bit set) can be defined.  
  Only controllable channels (Control access bit set) can be included in the scene definition. View-only channels are left unaffected by the scene set operation.Each message must be sent within 30 seconds of the previous message. Otherwise the whole scene definition process is aborted. (See section on Scene setting.)  
  e.g.  
  $scnSET,3;  
  !OK,SCNSET,00003;<CR><LF>  
  $scnFade,3,10000;  
  !OK,SCNFADE,00003,00010000;<CR><LF>  
  $scnChan,3,2,21,5,255;  
  !OK,SCNCHAN,00003,002,21,005,255;<CR><LF>  
  $scnDALI,3,3,17,12,200;  
  !OK,SCNDALI,00003,003,17,012,200;<CR><LF>  
  $scnDMXLevel,3,4,15,1,64;  
  !OK,SCNDMXLevel,00003,004,15,001,064;<CR><LF>  
  $scnEnd,3;  
  !OK,SCNEND,00003;<CR><LF>  
  Note: a scene setting operation can be aborted by sending *‘Abort scene definition’*  
  $SCNABORT; *Long-ack Format*: !OK,SCNEND;  
  !OK,SCNABORT;

### DMX Set Commands

This enables the DMX colour palette to be redefined.

* Set DMX palette colour  
  $DMXPALETTE,<addr>,<devcode>,<palette-colour>,<red>,<grn>,<blu>,  
   <wht>;  
  *Long-ack Format*: !OK,DMXPALETTE,000,000,000,000,000,000,000;  
  e.g.  
  $dmxPalette,02,15,15,255,128,64,0;  
  !OK,DMXPALETTE,002,015,015,255,128,064,000;<CR><LF>  
  Notes:  
  - the DMX colour palette applies to the whole DMX universe and all zones in that universe (hence no zone number required).  
  - each DMX universe (module) has its own palette  
  - the DMX palettes are separate from the plate button colour palette.  
  - the <white> component may be ignored depending on DMX fixture type

### Configuration Discovery Queries

A limited amount of information can be found about the configuration through the discovery queries. However, these commands only allow you to view the configuration, not change it, and they do not provide full access to the configuration.

The queries give you extra information about certain scenes that you can control using the scene commands. The information given is static, for example what the scene is called or which area the scene is in, so these queries can be performed once, unless the configuration is changed. The scenes available are set in the configuration.

* Request system ID  
  This is used to check if the configuration has changed since that last time you checked, i.e. do you need to re-query the configuration or is the extra information from previous discovery queries still valid? If either stamp value has changed then the configuration has been altered in some way.  
  ?SYSTEMID;  
  *Long-ack Format*: !OK,SYSTEMID;  
  *Reply: ID stamps that uniquely define a configuration state*!SYSTEMID,<serial-num>,<edit-stamp>,<adjust-stamp>;  
  Format: !SYSTEMID,00000000,00000000,00000000;  
  Serial number = serial number of NPU holding the configuration.   
  Edit stamp = 8-digit number representing when the last edit was saved  
  Adjust stamp = 8-digit number representing when the last adjustment was saved  
  e.g.  
  ?systemid;  
  !OK,SYSTEMID;<CR><LF>  
  !SYSTEMID,0002016D,1CC12058,1CC12058;<CR><LF>
* Request scene areas  
  This returns a list of available areas in the configuration. As part of the configuration, every scene is put in to an area. Each area has a number and a name.  
  ?AREANAMES;  
  *Long-ack Format*: !OK,AREANAMES;  
  *Reply: set of area names  
  zero or more ‘Report Area Name’*!AREANAME,<area-num>,<access>,<content>,<area-name>;  
  Format: !AREANAME,00000,00,000,text;  
  Access = not used;   
  Content bitfield = 1 (has channels) + 2 (has scenes) + 4 (has plates).  
  (There is no end-of-list message as in some systems there is not a single controller that knows where the end of a list is).  
  e.g.  
  ?areaNames;  
  !OK,AREANAMES;<CR><LF>  
  !AREANAME,00001,00,003,All;<CR><LF>  
  !AREANAME,00002,00,001,DALI;<CR><LF>
* Request scene names  
  This returns a list of scenes, the area they are in and their name. You can query for all available scenes, or just scenes in a particular area.  
  ?SCNNAMES;  
  ?SCNNAMES,<area-num>;  
  *Long-ack Format*: !OK,SCNNAMES; or !OK,SCNNAMES,00000;  
  *Reply: set of scene names  
  zero or more ‘Report Scene Name’*!SCNNAME,<scn-num>,<access>,<area-num>,<scn-name>;  
  Format: !SCNNAME,00000,00,00000,text;  
  Access = bit-field flags 1 (viewable) + 2 (controllable) + 4 (editable).  
  If the area number is not given then all scenes are reported.   
  (There is no end-of-list message as in some systems there is not a single controller that knows where the end of a list is).  
  e.g.  
  ?SCNNAMES,1;  
  !OK,SCNNAMES,00001;  
  !SCNNAME,00003,07,00001,Off;<CR><LF>  
  !SCNNAME,00004,07,00001,On;<CR><LF>  
  !SCNNAME,00005,07,00001,Kitchen;<CR><LF>
* Request plate names  
  This returns a list of wall plates, the type of plate they are, the area they are in and their name. You can query for all available plates, or just plates in a particular area.  
  ?PLATENAMES;  
  ?PLATENAMES,<area-num>;  
  *Long-ack Format*: !OK,PLATENAMES; or !OK,PLATENAMES,00000;  
  *Reply: set of plate names  
  zero or more ‘Report Plate Name’*!PLATENAME,<addr>,<devcode>,<style>,<access>,<area-num>,<plate-name>;Format: !PLATENAME,000,000,00,00,00000,text;  
  Style = 1 SGP style, 2 Coolbrium style, 3 ICON style  
  If the area number is not given then all plates are reported.   
  (There is no end-of-list message as in some systems there is not a single controller that knows where the end of a list is).  
  e.g.  
  ?PLATENAMES,1;  
  !OK,PLATENAMES,00001;  
  !PLATENAME,001,002,03,07,00001,Main;<CR><LF>  
  !PLATENAME,002,002,01,07,00001,Bedside;<CR><LF>
* Request button styles  
  This returns a list of buttons on a particular wall plate. Each button specifies the type of button they are and their name.  
  ?BTNSTYLE,<addr>,<devcode>,ALL;  
  ?BTNSTYLE,<addr>,<devcode>,<btn-num>;  
  *Long-ack Format*: !OK,BTNSTYLE,000,000,ALL; or !OK,BTNSTYLE,000,00,000;  
  *Reply: button style for plate buttons or single button  
  zero or more ‘Report Button Style*!BTNSTYLE,<addr>,<devcode>,<btn-num>,<icon>,<style>,<btn-name>;  
  Format: !BTNSTYLE,000,000,000,000,000,text;  
  Icon = reference number for button Icon  
  Style = not used  
  e.g.  
  ?BTNSTYLE,8,2,1;  
  !OK,BTNSTYLE,008,002,001;  
  !BTNSTYLE,008,002,001,001,000,;<CR><LF>
* Request channel names  
  This is primarily used for scene setting operations. It lists the available channels in a particular scene, and gives the name and area of each channel.  
  ?CHANNAMES,<scn-num>;  
  *Long-ack Format*: !OK,CHANNAMES,00000;  
  *Reply: set of channel names in the specified scene  
  zero or more ‘Report Channel Name’*!CHANNAME,<addr>,<devcode>,<chan-num>,<access>,<area-num>,<chan-name>;  
  !DALINAME,<addr>,<devcode>,<dali-num>,<access>,<area-num>,<chan-name>;  
  !DMXNAME,<addr>,<devcode>,<zone-num>,<access>,<area-num>,<zone-name>;  
  Format: !CHANNAME,000,000,000,00,00000,text;  
  Format: !DALINAME,000,000,000,00,00000,text;  
  Format: !DMXNAME,000,000,000,00,00000,text;  
  Access = bit-field flags 1 (viewable) + 2 (controllable) + 4 (editable).  
  (There is no end-of-list message as in some systems there is not a single controller that knows where the end of a list is).  
  e.g.  
  ?CHANNAMES,1;  
  !OK,CHANNAMES,00001;  
  !CHANNAME,002,021,003,03,00001,Downlight;<CR><LF>  
  !DALINAME,003,017,004,03,00001,Main Ceiling;<CR><LF>  
  !DMXNAME,004,015,001,03,00001,Features;<CR><LF>

### System Queries

* Request system error list  
  This returns a list of all current channel errors. Note: this is not a log of any error that has occurred; it only reports errors that are current or live. If an error is intermittent, it may not be shown in the list. It does not report all channels with no errors, i.e. channels that have the status code 0 Status Ok.  
  ?ERRORS;  
  *Long-ack Format*: !OK,ERRORS;  
  *Reply: set of errors  
  zero or more Channel Status Events*!MODUERR,<addr>,<devcode>,<status-code>;  
  !CHANERR,<addr>,<devcode>,<chan-num>,<status-code>;  
  !DALIERR,<addr>,<devcode>,<dali-num>,<status-code>;  
  !DALIERR,<addr>,<devcode>,<dali-fixture>,<status-code>;  
  !DMXERR,<addr>,<devcode>,<zone-num>,<status-code>;  
  !BTNERR,<addr>,<devcode>,<btn-num>,<status-code>;!INPERR,<addr>,<devcode>,<chan-num>,<status-code>;Format: see Channel Status Events section;  
  e.g.  
  ?errors;  
  !OK,ERRORS;<CR><LF>  
  !MODUERR,003,015,002;<CR><LF>  
  !DALIERR,001,017,001,025;<CR><LF>  
  !DALIERR,001,017,F03,025;<CR><LF>

### Scene Queries

* Request scene status  
  This is used to find out if a scene is active or inactive. It also tells you what ‘active’ means in terms of the state of its channels. You can query all scenes, scenes in a particular area or for a single scene.   
  ?SCNS;  
  ?SCNS,<area-num>;  
  ?SCN,<scn-num>;  
  *Long-ack Format*: !OK,SCNS; or !OK,SCNS,00000; or !OK,SCN,00000;  
  *Reply: zero or more ‘Report scene status’*!SCN,<scn-num>,<mode>,<flags>,<status>;Format: !SCN,00000,00,00,000;  
  Mode = 1 Scene, 2 Channel.  
  Flags = bit-field flags 1 (off scene) + 2 (strict).  
  If the area number is not given then all scenes are reported.   
  e.g.  
  ?scns,1;  
  !OK,SCNS,00001;  
  !SCN,00001,01,02,001;<CR><LF>  
  !SCN,00002,02,03,000;<CR><LF>
* Request scene channel status (request channel status for channels in scene)  
  This is a convenience method to poll for channel status. It returns a list of channels and their current status for all channels in a particular scene. This is primarily used during scene setting operations.  
  Note: Only available on NPU based systems.  
  ?SCNCHANS,<scn-num>;  
  *Long-ack Format*: !OK,SCNCHANS,00000;  
  *Reply: set of item statuses in scene  
  zero or more ‘Report (item) status’*!CHAN,<addr>,<devcode>,<chan-num>,<status>,<level>,<power>,  
  <wattage>;  
  !DALI,<addr>,<devcode>,<dali-num>,<status>,<level>,<power>,  
  <wattage>;  
  !DMX,<addr>,<devcode>,<zone-num>,<status>,<mode>,<level>,<power>,  
   <wattage>;  
  Only viewable scenes will report their channels. The ‘Report item status’ messages are the same as the replies from channel status queries.  
  Only viewable channels are reported.  
  (There is no end-of-list message as in some systems there is not a single controller that knows where the end of a list is).  
  e.g.  
  ?SCNCHANS,4;  
  !OK,SCNCHANS,00004;<CR><LF>  
  !CHAN,003,021,001,000,255,100,00100;<CR><LF>  
  !CHAN,003,021,002,000,255,100,00100;<CR><LF>  
  !CHAN,004,021,001,000,255,100,00100;<CR><LF>  
  !DALI,004,017,012,000,255,100,02500;<CR><LF>  
  !DMX,001,15,002,000,005,255,033,02500;<CR><LF>

### Scene Set Queries

* Request scene definition  
  This is used to support scene setting operations. It returns a list of channels that are in the scene and the level that they are set to by the scene. It also return the scene’s fade time.  
  ?SCNSET,<scn-num>;  
  *Long-ack Format*: !OK,SCNSET,00000;  
  *Reply: start of scene definition*!SCNSET,<scn-num>; *followed by zero or more ‘Report Scene Item’*!SCNFADE,<scn-num>,<fadetime(ms)>; or  
  !SCNCHAN,<scn-num>,<addr>,<devcode>,<chan-num>,<level>; or  
  !SCNDALI,<scn-num>,<addr>,<devcode>,<dali-num>,<level>; or  
  !SCNDMXLEVEL,<scn-num>,<addr>,<devcode>,<zone-num>,<level>; or  
  *followed by ‘End of scene items’*  
  !SCNEND,<scn-num>;  
  Format:!SCNSET,00000;  
  Format:!SCNFADE,00000,00000000;  
  Format:!SCNCHAN,00000,000,000,000,000;  
  Format:!SCNDALI,00000,000,000,000,000;  
  Format:!SCNDMXLEVEL,00000,000,000,000,000;  
  Format:!SCNEND,00000;  
  Only editable scenes will report their items.Only controllable channels are reported.   
  e.g.  
  ?scnSET,3;  
  !OK,SCNSET,00004;<CR><LF>  
  !SCNSET,00003;<CR><LF>  
  !SCNFADE,00003,00010000;<CR><LF>  
  !SCNCHAN,00003,002,021,005,255;<CR><LF>  
  !SCNDALI,00003,003,017,012,200;<CR><LF>  
  !SCNDMXLEVEL,00003,004,015,0012,064;<CR><LF>  
  !SCNEND,00003;<CR><LF>

### System Events

* *Event: ‘scene default recall occurred’*!SCNRECALL,<scn-num>;   
  Format:!SCNRECALL,00000;  
  e.g.  
  !SCNRECALL,00008;<CR><LF>
* *Event: ‘scene recall occurred’*!SCNRECALLX,<scn-num>,<level>,<fadetime(ms)>;   
  Format:!SCNRECALLX,00000,000,00000000;  
  e.g.  
  !SCNRECALLX,00008,255,00001000;<CR><LF>
* *Event: ‘scene recall off occurred’*!SCNOFF,<scn-num>;   
  Format:!SCNOFF,00000;  
  e.g.  
  !SCNOFF,00008;<CR><LF>
* *Event: ‘scene fast recall occurred’*!SCNFAST,<scn-num>;   
  Format:!SCNFAST,00000;  
  e.g.  
  !SCNFAST,00008;<CR><LF>
* *Event: ‘scene recall back to toggle level occurred’*!SCNBACKON,<scn-num>;   
  Format:!SCNBACKON,00000;  
  e.g.  
  !SCNBACKON,00008;<CR><LF>
* *Event: ‘scene raise occurred’*  
  !SCNRAISE,<scn-num>;  
  Format: !SCNRAISE,00000;  
  e.g.  
  !SCNRAISE,00008;<CR><LF>
* *Event: ‘scene lower occurred’*  
  !SCNLOWER,<scn-num>;   
  Format: !SCNLOWER,00000;  
  e.g.  
  !SCNLOWER,00008;<CR><LF>
* *Event: ‘scene ramp occurred’*  
  !SCNRAMP,<scn-num>;   
  Format: !SCNRAMP,00000;  
  e.g.  
  !SCNRAMP,00008;<CR><LF>
* *Event: ‘scene stop occurred’*  
  !SCNSTOP,<scn-num>;   
  Format: !SCNSTOP,00000;  
  e.g.  
  !SCNSTOP,00008;<CR><LF>
* *Event: ‘scene nudge up occurred’*  
  !SCNNUDGEUP,<scn-num>;   
  Format: !SCNNUDGEUP,00000;  
  e.g.  
  !SCNNUDGEUP,00008;<CR><LF>
* *Event: ‘scene nudge down occurred’*  
  !SCNNUDGEDN,<scn-num>;   
  Format: !SCNNUDGEDN,00000;  
  e.g.  
  !SCNNUDGEDN,00008;<CR><LF>
* *Event: ‘scene recall on/off occurred’*  
  !SCNONOFF,<scn-num>;   
  Format: !SCNONOFF,00000;  
  e.g.  
  !SCNONOFF,00008;<CR><LF>
* *Event: ‘scene toggle recall occurred’*  
  !SCNTOGGLE,<scn-num>;   
  Format: !SCNTOGGLE,00000;  
  e.g.  
  !SCNTOGGLE,00008;<CR><LF>
* *Event: ‘scene save occurred’*  
  !SCNSAVE,<scn-num>;   
  Format: !SCNSAVE,00000;  
  e.g.  
  !SCNSAVE,00008;<CR><LF>
* *Event: ‘Change in DMX Palette’*!DMXPALETTE,<addr>,<devcode>,<code>,<red>,<grn>,<blu>,<wht>;Format:!DMXPALETTE,000,000,000,000,000,000,000;  
  e.g.  
  !DMXPALETTE,008,015,004,255,255,255,0;<CR><LF>

# Enumerations

This section defines the values used by various commands and queries.

* Device Codes

|  |  |  |
| --- | --- | --- |
| devCode | Product Code | Products |
| 01 | EVO-LCD-55 | LCD Wall Plate |
| 02 | EVO-SGP-xx | 2, 5 and 10 button Wall Plates,  Coolbrium & Icon plates |
| 04 | EVO-RP-03-02 | Evo 2-channel Relay Module |
| 08 | EVS-xxx | All Evo Slave Packs |
| 09 | EVO-INT\_CI\_xx | Evo 4 & 8 channel Contact Input modules |
| 12 | DIN-02-08 | eDIN 2A 8 channel dimmer module |
| 14 | DIN-03-04 | eDIN 3A 4 channel dimmer module |
| 15 | DIN-INT-00-08 | eDIN 8 channel IO module |
| 16 | DIN-RP-05-04 | eDIN 5A 4 channel relay module |
| 17 | DIN-UBC-01-05 | eDIN Universal Ballast Control module |
| 18 | DIN-COUT-00-08 | eDIN 8 channel Configurable Output module |
| 24 | ECO\_MULTISENSOR | eDIN Mk 1 Multisensor |
| 144 | DIN-RP-05-04 | eDIN 5A 4 channel mains sync relay module |
| 145 | DIN-UBC-01-05 | eDIN Universal Ballast Control 2 module |

* Contact Input and Button State

|  |  |  |
| --- | --- | --- |
|  | value | state |
|  | 0 | Release-off |
|  | 1 | Press-on |
|  | 2 | Hold-on |
|  | 5 | Short-press |
|  | 6 | Hold-off |

* PIR Input State

|  |  |  |
| --- | --- | --- |
|  | value | state |
|  | 0 | Empty |
|  | 1 | Triggered |
|  | 2 | Timeout |
|  | 3 | Set |
|  | 4 | Hold |
|  | 5 | Clear |

* Plate Styles

|  |  |  |
| --- | --- | --- |
|  | value | Products |
|  | 1 | 2, 5 and 10 button SGP Wall Plates |
|  | 2 | Coolbrium plates |
|  | 3 | Icon plates |

* eDIN Button Palette Colours

|  |  |  |
| --- | --- | --- |
|  | value | Name |
|  | 0 | Black |
|  | 1 | White |
|  | 2 | Red |
|  | 3 | Green |
|  | 4 | Blue |
|  | 5 | Orange |
|  | 6 | Cyan |
|  | 7 | Magenta |
|  | 8 | Yellow |
|  | 9 | DimWhite |
|  | 10 | DimRed |
|  | 11 | DimGreen |
|  | 12 | DimBlue |
|  | 13 | DimOrange |
|  | 14 | DimCyan |
|  | 15 | DimMagenta |
|  | 16 | DimYellow |

|  |  |  |
| --- | --- | --- |
|  | value | Name |
|  | 128 | User1 |
|  | 129 | User2 |
|  | 130 | User3 |
|  | 131 | User4 |
|  | 132 | User5 |
|  | 133 | User6 |
|  | 134 | User7 |
|  | 135 | User8 |
|  | 136 | User9 |
|  | 137 | User10 |
|  | 138 | User11 |
|  | 139 | User12 |
|  | 140 | User13 |
|  | 141 | User14 |
|  | 142 | User15 |
|  | 143 | User16 |

* eDIN DMX Palette and Modes

|  |  |  |
| --- | --- | --- |
|  | value | Mode |
|  | 1-15 | Static Colour |
|  | Palette code | Static wash using palette colour |
|  | 64-68 | Solid Colour Sequence |
|  | 64 | Long Rainbow Solid |
|  | 65 | Short Rainbow Solid |
|  | 66 | Hot Colours Solid |
|  | 67 | Cold Colours Solid |
|  | 68 | User Colours Solid |
|  | 96-100 | Ripple Colour Sequence |
|  | 96 | Long Rainbow Ripple |
|  | 97 | Short Rainbow Ripple |
|  | 98 | Hot Colours Ripple |
|  | 99 | Cold Colours Ripple |
|  | 100 | User Colours Ripple |

|  |  |  |
| --- | --- | --- |
|  | Palette code | Default Name |
|  | 1 | Red |
|  | 2 | Orange |
|  | 3 | Yellow |
|  | 4 | LawnGreen |
|  | 5 | Green |
|  | 6 | Mint |
|  | 7 | Cyan |
|  | 8 | DeepSkyBlue |
|  | 9 | Blue |
|  | 10 | Purple |
|  | 11 | Magenta |
|  | 12 | DeepPink |
|  | 13 | User1 |
|  | 14 | User2 |
|  | 15 | User3 |

* eDIN Module Palette and Modes

|  |  |  |
| --- | --- | --- |
|  | value | Mode |
|  | 1-15 | Static Colour |
|  | Palette code | Static wash using palette colour |

|  |  |  |
| --- | --- | --- |
|  | Palette code | Default Name |
|  | 1 | Red |
|  | 2 | Yellow |
|  | 3 | Green |
|  | 4 | Cyan |
|  | 5 | Blue |
|  | 6 | Magenta |
|  | 7 | UserColour |
|  | 8 | Candlelight |
|  | 9 | SoftWhite |
|  | 10 | WarmWhite |
|  | 11 | White |
|  | 12 | CoolWhite |
|  | 13 | BrightWhite |
|  | 14 | Daylight |
|  | 15 | UserTemperature |

* Standard Status Codes

|  |  |  |  |
| --- | --- | --- | --- |
|  | value | Error | Description |
|  | 0 | Status Ok | No Errors |
|  | 2 | Device missing | Device or Module is not responding to MBus messages. |
|  | 4 | Bad Device Firmware | System is configured to use features that are not present in current module firmware. |
|  | 5 | No AC | Module uses mains AC and it does not detect any main AC power |
|  | 6 | Too Hot | The module has detected that its internal temperature is above its maximum rated operating temperature. |
|  | 10 | Channel Load Failure | The module has detected there is a problem with the external load a channel is driving |
|  | 20 | No DALI PSU | The module has detected that there is no PSU on its DALI bus. |
|  | 21 | No DALI  Commissioning Data | The DALI universe on this module does not contain any commissioning data. |
|  | 22 | DALI Commissioning problem | The module has detected that the actual DALI fixtures detected do not match with the commissioning data |
|  | 25 | DALI Lamp failure | A DALI fixture on this channel is indicating a lamp failure condition |
|  | 26 | DALI missing ballast | A DALI fixture that is in the commissioning data is not present (is not responding). |

* DALI Scan Status Codes

|  |  |  |  |
| --- | --- | --- | --- |
|  | value | Name | Description |
|  | 0 | Idle | No Error with fixture |
|  | 1 | Done | A Search or programming operation has completed successfully or been aborted. |
|  | 2 | Searching | A search is in progress. |
|  | 3 | Programming | A repair operation is in progress |
|  | 4 | Error | A search or programming operation has ended in an error |

* DALI Fixture Status Codes

|  |  |  |  |
| --- | --- | --- | --- |
|  | value | Error | Description |
|  | 0 | Ok | No Error with fixture |
|  | 1 | Lamp Failure | Fixture is reporting Lamp Failure |
|  | 2 | Missing | Fixture is not responding to messages on DALI bus. |
|  | 5 | New | Fixture is present but does not match commissioned (expected) fixture list |
|  | 8 | Address Clash | There is more than one fixture with the same short address. (Commissioning problem) |
|  | 9 | Unassigned | Fixture does not exist or does not have a valid DALI short address. |

1. Currently the Gateway Interface is only implemented via an NPU connection and requires a minimal configuration that contains the modules in the lighting system [↑](#footnote-ref-1)