



## OpenLCB Technical Note

### Unique Identifiers

December 13,  
2025

Draft

## 1 Introduction

This technical note contains informative discussion and background for the corresponding “OpenLCB Unique Identifiers Standard”. This explanation is not normative in any way.

## 2 Annotations to the Standard

- 5 This section provides background information on corresponding sections of the Standard document. It's expected that two documents will be read together.

### 2.1 Introduction

- 10 Originally, OpenLCB Unique Identifiers was defined in terms of 48-bit unique “Node IDs”. Other uses were found for these identifiers that didn't associate them with a specific node, which demonstrated the need for Unique Identifiers beyond just Node IDs. Therefore, the Standard and this Technical Note are generally written in terms of Unique Identifiers, which includes Node IDs usage.

### 2.2 Intended Use

- 15 The “globally unique” requirement only refers to the universe of connected nodes; nodes that never need to communicate with each other need not have separate Node IDs. In general, however, nodes can move: they can be sold or loaned for use on another layout, nodes on modular layouts can be connected to other arbitrary modules, and few assumptions can be made. It is best if nodes are given a completely unique identifier when manufactured, so there's no need to ever detect and resolve a conflict. If a Node ID conflict should occur on an OpenLCB
- 20 network, the Message Network Standard defines how the conflict should be handled. In general, OpenLCB standards place the effort and burden on preventing a conflict from occurring rather than resolving a conflict if and when it does occur.

### 2.3 References and Context

### 2.4 Format

- 25 The Standards don't require any particular human-readable format for input and output, but hex-pairs with separators (example: 01.AB.34.01.CD.E3) are recommended by the Common Information TN. If any other format is used, including decimal pairs, it's very important to make it clear how to interpret it.

- 30 There are many methods to store a Unique Identifier, and there is no constraint placed by the Unique Identifier Standard on how a Unique Identifier is stored. It could be stored in a non-volatile memory, as jumpers on a board, etc.

## 2.5 Allocation

### 2.5.1 Overview

Unique Identifiers are assigned via a delegation process. At the highest level, ranges are assigned to people and organizations, within which they are responsible for assigning Unique Identifiers to separate devices. These ranges can be subdivided and delegated further, as needed. Additional ranges can then be requested, which will eventually be recorded in the Unique Identifier Standard or the Unique Identifier Appendix as appropriate. The OpenLCB organization reserves the right to allocate Unique Identifiers within a separate database in real-time and periodically update the Unique Identifier Appendix document from this database. In the unlikely event that a conflict shall arise between OpenLCB managed real-time database and the Unique Identifiers Appendix, the Unique Identifiers Appendix, document represents the true allocation.

One of the reasons for having a long, 48-bit Unique Identifier space is to make it easier to use a delegation system like this. Because there are a lot of possible Unique Identifiers, large ranges can be delegated to groups without having to ensure that the range be efficiently used. For example, most NMRA members will not design their own OpenLCB nodes and need to assign Node IDs, but assigning a range to every member makes it easy for those who want to, at a total cost of less than 0.0016% of the available Unique Identifier space.

In the delegated assignments, the lower order byte(s) are self-assigned. Though not strictly required, by convention, the value of zero for the lower self-assigned byte(s) should be reserved, indicating that a number within the range hasn't been assigned.

The high byte of each range is different for each type of assignment, making it easy to determine the allocation pattern in use for a particular Unique Identifier.

Allocations are meant to be unique forever, so the standard requires that new allocation ranges not overlap existing ones, and allocation regions not be reused later.

### 2.5.2 Reserved Leading Zero

A Unique Identifier with a most significant value of 0x00 is never valid. This range is reserved. A message utilizing one of these values must never be sent on the OpenLCB bus. If a node receives a message utilizing a Unique Identifier from this range, it may ignore the message, or throw an error, however, it must never act on the message as if it was a valid Unique Identifier.

The fact that the Unique Identifiers in this range are invalid may be, and often is, exploited within the internal software of a node in order to mark an unused, or invalid, Node ID. This use is private and internal to the node and as such is not subject to scrutiny by the OpenLCB standards as long as it is not exposed external to the node.

### 2.5.3 Well-Known Global Identifiers

In addition to use as Node Identifiers (Node IDs), OpenLCB Unique Identifiers are used to ensure uniqueness of specific global event identifiers and for other purposes. These numbers must be allocated so that they are kept unique. The identifiers specified in this section are of that type.

Note that the detailed use of these identifiers is specified elsewhere. In some cases, the protocols are still being developed, and the entry here is just reserving a range for a specific future use.

## 2.5.4 Manufacturer Specific

This group of Unique Identifiers is reserved specifically for manufacturers. Manufacturers may request a range of Unique Identifiers for use within their products. A Manufacturer may assign the Unique Identifiers within their allocated range at their own discretion so long as every assignment is unique.

- 75 A manufacturer is not required to assign Unique Identifiers sequentially. They may choose their own arbitrary scheme that could be based on product type, manufacture date, or some other method of their choosing. Though it is strongly recommended for a manufacturer to make reasonably efficient use of their self-assigned space, should a manufacturer require additional space, it may be requested, and granted, without the requirement of having used every last available Unique Identifier within a previously assigned space. Manufacturers are not required to publicly disclose their allocation scheme for Unique Identifiers.

- 85 In order to encourage existing manufacturers to participate in developing OpenLCB products, every DCC manufacturer has already been allocated a 24-bit region within which to self-assign Unique Identifiers from. The DIY, JMRI, MERG, and NMRA Reserved spaces are sub-sets of the DCC manufacturer space and are consistent with the DCC ID's already assigned for these purposes in the NMRA standards doc "S-9.2.2 Appendix A, DCC Manufacturer ID Codes". They are singled out only for the purpose of drawing attention to the fact that they exist and not for any other purpose.

- 90 Only those DCC manufacturers assigned a short (8-bit) DCC manufacturer ID have allocations in this region. For those manufacturers that obtain a long (16-bit) DCC manufacture ID, please see section 2.5.11. The result is that those manufacturers who happen to be earlier assignees of a unique DCC manufacture ID have a slight advantage in that they have two 24-bit regions assigned to them without having to explicit request assignment of a 2<sup>nd</sup> 24-bit region. The reason for this is accidental. It was not known to the OpenLCB developers at the time of the first DCC manufacture ID based allocation what the scheme for long (16-bit) DCC manufacture ID allocation would be. When the long (16-bit) DCC manufacture allocation scheme became known, it also became clear that this scheme would not fit within the unassigned space remaining in the Manufacture Specific region.

The advantage posed by having two automatically allocated 24-bit DCC manufacture ID based regions is negligible. As stated in section 2.5.1, there is no shortage of 24-bit regions that can be assigned, and another 24-bit region can be assigned to any manufacture upon request.

## 100 2.5.5 Self-Assigning Groups

- 105 MERG kit builders and others would like to assign their own identifiers without going through a complicated process. To make this possible without any interaction with anybody, these groups are assigned identifier ranges that involve their member number in the organization. A member may assign any identifier from this range to the node(s) they produce, provided that each identifier is assigned to at most one node. A range of 255 identifiers per member is typically sufficient for hobby usage. Should a hobbyist exhaust their assigned range, the hobbyist can get another, larger assignment. It's also convenient to give hobbyists a byte as their range.

- 110 Each organization is assigned a unique high-order two bytes. The organization member number is given 24 bits. Byte 2 of the Unique Identifier advances by 4 between groups (NMRA is 0x00, MERG is 0x04, etc) to allow a little more headroom on group membership numbers; this space can be reclaimed later if needed.

Other groups have defined mechanisms to ensure that their node numbers or equivalent constructs are uniquely assigned. They may have non-technical reasons for wishing to use those same mechanisms to assign OpenLCB unique identifiers. Ranges of OpenLCB Unique Identifiers can be assigned to these groups, so that members may then use their group's mechanism to select a value within that range, the result will be a properly unique OpenLCB Unique Identifier.

The first example of this is MERG CBUS developers. MERG CBUS has defined a “no cost” way of identifying unique 16-bit Node Number (NN) for CBUS use, perhaps with an optional 16-bit Layout Number (LN). People who wish to use this mechanism to allocate unique OpenLCB Node ID identifiers can, without having to consult anybody, generate an OpenLCB Node ID from the unique CBUS number(s) as described in the Standard.

If the user is involved in determining the Unique Identifier for a node (the Node ID) by setting switches, the possibility of duplicated Node IDs must be considered. Users make mistakes. To reduce user frustration, the node should provide a user-visible way to indicating a duplicate has been seen, and should fully implement the relevant wire-protocol-specific methods for detecting duplicate Node IDs.

### 2.5.6 Assigned by Software at Runtime

Programs that act as one or more OpenLCB nodes need to associate unique identifiers with them. For licensed software, where a unique key can be associated with each instance of the program, this is easy: Use the manufacturer space defined above, and generate the lower bits of a specific ID from the license key.

Free, open and unlicensed software can't use a license-key-based method. Unfortunately, the 48-bit address space is too small to use the IP-address-plus-signature GIDs that would otherwise make this a simple problem, or the even larger MAC-address-plus-signature GIDs.

Initial experiments were done using 32-bit IPv4 addresses as components of Unique Identifiers, but this is no longer recommended for several reasons:

- Not all IPv4 addresses are globally unique. Some IP addresses correspond to “private networks”, which are only locally unique. See RFC 1918 and RFC 3330 for more information. In addition, Microsoft defined a non-IETF “Automatic Private IP Addressing” mechanism for providing non-global IP addresses.
- A single computer may run several programs, so there still needs to be separate mechanism to provide a unique value for the lowest bytes of the ID. That involves a level of coordination across multiple software vendors that is hard to imagine.
- IPv6 is coming. It provides addresses that are too large to use directly. Even before that happens, the various issues of IPv4 to IPv6 mapping raise all sorts of questions about uniqueness of IPv4 addresses.
- Even globally routable IPv4 addresses may not be unique over time. For example, DHCP may assign the same address to multiple computers sequentially. This is particularly an issue with wireless access at e.g. clubs and shows.

Computers that have global Internet access, even if they don't have a permanent and unique IP address, can still get a Unique Identifier from an openlcb.org-provided service. These Unique Identifiers are provided from a specified range to ensure that they are unique when created. Each identifier is only

provided once to ensure that it remains unique. Programs using this facility should permanently remember Unique Identifiers obtained this way, because they won't get the same one on a later request.

155 Other organizations can also distribute Unique Identifiers from within their allocated blocks. For example, a Unique Identifier could be provided when a free-software program is downloaded, perhaps as part of the download package or even as part of its filename.

Programs without access to an ID-providing service must use some other mechanism, which may result in prompting the user for a Unique Identifier assigned by one of the other mechanisms.

### 2.5.7 Specifically Assigned by Request

160 Users can request blocks of Unique Identifiers of various sizes. The small (256) and medium (65536) blocks are not scarce resources. Requests for these blocks should be routinely granted once the requester has been identified. The 24-bit blocks are slightly scarcer, but there are still almost  $2^{16}$  of them available by using additional values for byte 2.

165 An automated system for requesting and obtaining unique ID ranges is available at <http://registry.openlcb.org> and subsequent pages.

### 2.5.8 Locomotive Control Systems

170 Though locomotive control was initially beyond the scope of OpenLCB development, later work has defined OpenLCB methods for working with existing locomotive control systems. This section specifies ranges of Unique Identifiers that are reserved for the purpose of interfacing with existing locomotive control, including how a given Unique Identifier maps into the address space of the existing locomotive control system. Additional details of how these Unique Identifiers are to be used are specified elsewhere, but sufficient range has been reserved to allow providing existing locomotive control systems with Unique Identifiers.

175 The expected use case is for a physical OpenLCB device to provide virtual Node representations with Unique Identifiers defined within this space. The virtual Node representations may be static (created automatically upon startup of the OpenLCB device) or dynamic (created on demand by mechanisms defined elsewhere).

180 It is the responsibility of the user to ensure that only one OpenLCB device can exist in the network that provides a virtual Node representation of a given Unique Identifier in this space. These Unique Identifiers are not expected to be globally unique among all OpenLCB networks, but are expected to be unique within the confines of a single OpenLCB network. It is for this reason that assignment of Unique Event Identifiers out of this range are not allowed. Manufacturers are encouraged to provide guidance, in the form of user facing documentation, to ensure that these criteria are met.

185 While a Unique Identifier space has been reserved for the MTH DCS system, the details of this system are not well known enough at this time to provide greater specificity. Public information about the MTH DCS protocol is limited. Manufacturers interested in implementing the MTH DCS system are encouraged to share their knowledge of the MTH DCS system and its supported addressing schemes so that this Unique Identifier space may become better defined in the future.

## 2.5.9 RFID and NFC

190 The RFID and NFC Unique Identifiers space is reserved to be used in future standards to be defined elsewhere.

## 2.5.10 Temporary Assigned by Software at Runtime

195 As described in the section 2.5.6 Assigned by Software at Runtime above, there are many reasons for which a node may want to be assigned a Unique Identifier at runtime. This Unique Identifier space is specifically designated for temporary (leased) assignment of these Unique Identifiers. There is a contract implied with the assignment between the assigning server and client node that is valid for a time period defined by the server. Once the prescribed time period expires, the client node must cease usage of the previously assigned Unique Identifier.

200 It is left up to the user to guarantee that there are not two or more servers with the potential to assign the same Unique Identifier within a single OpenLCB network.

205 The implementation of the client server protocol is not explicitly defined, and currently there is no OpenLCB protocol that allows this client/server process to run over an OpenLCB network. Other mechanisms, such as a TCP connection over the Internet may be of use in implementation. The following list suggests implementation details that are not required, but might be considered when designing such a client server pair:

- Lease time can be extended through a renewal process.
- The server does not reassign a previously assigned and currently expired Unique Identifier until it has run through the entire pool allocated to it.
- The server allows the client to suggest a Unique Identifier that it would like from the server's pool.
- The server have persistent storage of current leases that it can consult in case of a restart.
- The client remembers its last Unique Identifier assignment and suggests to the server to be re-assigned this Unique Identifier upon reset, power failure, or reconnection to the server following a previous disconnect.

215 Use this pool with caution. Implementation mechanisms that make use of this pool are currently experimental. Because Unique Event Identifiers assigned out of this range could be captured and disseminated into use by nodes that could become unaware of a lease expiration and reassignment, Unique Event Identifiers are not to be assigned out of this range.

## 2.5.11 Long (1246-bit) NMRA DCC Manufacture Specific

220 When it became known that the long (1246-bit) NMRA DCC manufacture ID space would not fit within the unassigned space of the Manufacture Specific region described in section 2.5.4, this region was reserved to be assigned to DCC manufacturers with a long (1246-bit) DCC manufacture ID. Because all short (8-bit) DCC manufacturer IDs can all be represented as valid long (1246-bit) DCC manufacture IDs by having the most significant of the long (1246-bit) DCC manufacture ID set to 0x00, those manufacturers with a short (8-bit) DCC manufacture ID have two 24-bit allocations assigned to them without having to explicitly request any additional allocation.

### **2.5.12 Locally-Allocated Identifiers**

It is strongly recommended that Unique identifiers be constructed from ranges allocated to a particular user or manufacturer (Sections 5.4, 5.5, 5.7, and 5.11), or those dedicated to specific purposes by the Unique Identifiers Standard (Sections 5.3, 5.6, 5.8, and 5.10). This guarantees that there will be no conflicts with other equipment currently part of an LCC network or added in the future.

Never-the-less, some constructors of LCC networks want to allocate bits in an Event ID<sup>1</sup> to code for various things, such as allocating bit fields to carry the function, type and location of the device producing or consuming them. The large address space of LCC's Event IDs makes this easier to do than with other systems which typically have 11 or 12 bits available.

Assigning Event IDs using a custom allocation system, without properly allocating the underlying Unique ID, will cause conflicts to occur. One obvious case is when a node is taken from a layout with one custom assignment to a layout with another, i.e. as part of a modular setup. But conflicts can arise even if the nodes aren't moved to another layout. For example, when installing a new node that uses properly allocated Unique IDs and Event IDs, those might conflict with the custom allocation system. Or when a throttle or train is brought to run on the layout, that might cause a conflict. These can be difficult to resolve.

The 0x0A Unique Identifier range is meant to reduce the probability of conflicts when doing local allocation of Unique IDs and Event IDs.

By allocating Unique Ids and Event Ids that start with a 0x0A byte, the implementor can be assured that no new node brought to the layout with properly allocated Unique IDs and Event IDs will conflict: No new node will have IDs that start with 0x0A.

An example club coding might be:

- Use byte 2 to code a location on the layout
- Use byte 3 to code a sub-location such as the west-bound end of the location
- Use byte 4 to code the particular device type, such as signal, turnout, etc.
- Use byte 5 to code the specific device, such as the signal number within the location

Note that this is not a recommendation for allocating unique IDs. The recommended allocation system is defined in other sections of the Standard and Technical note.

Use of the 0x0A allocation does not resolve conflicts with a locally-allocated node is taken to another layout with locally-allocated Unique Ids and Event IDs. The 0x0A prefix is not intended for e.g. modular layout groups. It is intended for independent clubs and home layouts that want to create their own allocation system for use only on their layout.

### **2.5.13 Devices originating RailCom© and Bi-Directional Communications<sup>2</sup>**

OpenLCB is reserving the allocations from 0x12.\*.\*.\*.\* to 0x1F.\*.\*.\*.\* so that they can be reclaimed for other purposes should they not be needed for manufacturer IDs at some point in the

<sup>1</sup>See the Event Identifiers Standard and Technical Note

<sup>2</sup>The NMRA uses “Bid-Directional Communications” and the RailCommunity uses RailCom to refer to the ability of a DCC decoder to return communications to a detector. Note that RailCom is the registered trademark of Lenx GmbH in Germany.



future. The NMRA and RailCommunity are doing the first allocations of manufacturer IDs with the top 4 bits of zero. Discussions are underway (2025) to request that they use a one in the top four bits for when they need to extend that range.

## 265 | 2.5.14 Reserved Unique Identifiers

For error detection, we permanently reserve all identifiers that start with either a 0x00 or 0xFF value. OpenLCB implementations should, but are not required to, treat it as an error when any of those are encountered.

## 3 Implementation Information

270 Specific Unique Identifier assignments are stored in a MySQL database, and the full list of assignments, including overall ranges from the standard and ranges assigned for specific purposes and users can be found on the <http://registry.openlcb.org> website.

Automated allocation systems can be abused, and we don't want to give away large chunks of address space to automated requesters. All available information about requests is logged. Users are asked for their name and contact information at the time of the request, which is also logged. Depending on  
275 experience with requests, an email challenge-response or other mechanism to ensure only valid requests get allocations may need to be added in the future.

Because of the inherent difficulties in protecting an SQL or other Internet accessible databases from abuse, both intentionally malicious and unintentional, a Unique Identifier Appendix document will  
280 periodically be generated for the purposes of archival to a higher integrity revision control system. The Unique Identifier Appendix document is the final authority on assignments should the SQL database become corrupted creating a discrepancy between the two.



## Table of Contents

1 Introduction.....	1
2 Annotations to the Standard.....	1
2.1 Introduction.....	1
2.2 Intended Use.....	1
2.3 References and Context.....	1
2.4 Format.....	1
2.5 Allocation.....	2
2.5.1 Overview.....	2
2.5.2 Reserved Leading Zero.....	2
2.5.3 Well-Known Global Identifiers.....	2
2.5.4 Manufacturer Specific.....	3
2.5.5 Self-Assigning Groups.....	3
2.5.6 Assigned by Software at Runtime.....	4
2.5.7 Specifically Assigned by Request.....	5
2.5.8 Locomotive Control Systems.....	5
2.5.9 RFID and NFC.....	6
2.5.10 Temporary Assigned by Software at Runtime.....	6
2.5.11 Long (12-bit) NMRA DCC Manufacture Specific.....	6
2.5.12 Locally-Allocated Identifiers.....	7
2.5.13 Devices originating RailCom© and Bi-Directional Communications.....	7
2.5.14 Reserved Unique Identifiers.....	8
3 Implementation Information.....	8