OpenLCB Technical Note	
Event Identifiers	
Feb 6, 2016	Adopted

1 Introduction

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

2 Annotations to the Standard

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

Although event identifiers generally take the format of a Unique Node Identifier plus a 16-bit extension, there are some exceptions to this noted in the Event Identifiers Standard which will be detailed in this document.

2.2 Intended Use

An Event Identifier is intended to be globally unique and for a specific purpose defined either by user configuration or enforced by the Event Identifiers Standard. Any node may be configured to produce or consume any event, regardless of its assigned Node ID. However, logically, a unique Event Identifier is defined to have a unique purpose, whether assigned by user configuration or explicitly defined in the Event Identifier Standard.

The "globally unique" requirement only refers to the universe of connected nodes; nodes that never need to communicate with each other don't need to have separate Event Identifiers. 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. Therefore, we require global uniqueness for all Event Identifiers.

To ensure uniqueness, the top six bytes of an Event Identifier that the manufacturer or user defines are required to be within a Node ID space that the manufacturer or user controls. The low two bytes can be any number, so long as each value is used for only one event.

This requirement applies equally to events defined by a hardware node, like a push button, a software node in a computer, or Event Identifiers that are defined by a human writing them on a piece of paper. In each case, the thing doing the definition must ensure it has control over the Node ID corresponding to the top six bytes, so it can ensure that the Event Identifier not be reused.

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
Node ID assigned to you						Unique 16 bit extension	

The Node ID part can be from real nodes which a user owns.

A software configuration tool might define events and assign Event Identifiers to them. A board manufacturer may prefer a pushbutton configuration process. A modular club may decide that certain events form the "boundaries" of modules, and need to be assigned Well-Known Event Identifiers to make it easier to create large modular layouts. Whatever the method, the Event Identifiers need to be globally unique, which is ensured by requiring Event Identifiers to be created using Node ID numbers assigned to them (and therefore not assigned to anybody else), plus an additional 16 bits that they are responsible for using only once. This is worth repeating, each Event Identifier can only be used once for a specific state or meaning. It doesn't mean that multiple nodes cannot use it, but rather it should not be used for a different purpose or meaning. This is because the meaning and its use is shared across all of the nodes using it, and unless it can be guaranteed that they all have been changed there will be conflict between them. It is much safer to use a new Event Identifier, and nodes will usually have a mechanism to supply new, "virgin" Event Identifiers.

Note that the Node ID part of an Event Identifier does not have to correspond to any physical node. So long as it is assigned out of a Node ID address space that the assigning body owns, and, by extension, has control of over, it can be used. For example, a fast-clock manufacturer might want to define a range of Event Identifiers so that a different event is emitted every fast minute. For the sake of argument, let's say 24 bits worth of specific events are needed (it's actually smaller, but let's use this as an example). The manufacturer has a large range of Node IDs assigned already, so can use 8-bits worth of that space, plus the extra 16 bits in the Event Identifiers below the Node ID portion, to do this. If a manufacturer had a range of Node IDs that included at least 12.34.56.78.9A.00 through 12.34.56.78.9A.FF (in other words, all possible values of the low byte), they could create a set of Event Identifiers with the low 24 bits used to carry a time value:

Table 2: Example of defining a Set of Events

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
Assigned Node ID						Unique 16 bit extension	
0x12	0x34	0x56	0x78	0x9A	Time Value		

This is guaranteed to be unique, no matter where one of these devices is used, because the node ID range is guaranteed to belong to the manufacturer, and they will use it only once for this purpose.

2.3 References and Context

This Standard is in the context of the following OpenLCB Standards:

- The Message Network Standard, which defines the basic messages and how they interact. Higher-level protocols are based on this message network, but are defined elsewhere.
- The Event Transport Standard, which defines the protocol for transporting events.

- The Unique Identifiers Standard which defines the format and allocation of unique 48-bit identifiers.

2.4 Format

Event Identifiers are intended to be globally unique 64-bit values.

2.5 Allocation

2.5.1 Node ID Based

The majority of Event Identifiers fall into this category. The six most significant bytes are derived from a globally unique six byte Node ID followed by the two least significant bytes which are typically chosen through user configuration.

2.5.1.1

2.5.2 Well-Known Automatically-Routed

Unlike Node ID Based Event Identifiers, the Well-Known Automatically Routed Event Identifiers are explicitly defined by the Event Identifiers Standard for a specific purpose. Additionally, gateways are required to route these events to all segments.

Note a change that it is not specified whether or not the Well-Known Automatically Routed events participate in producer/consumer identify. The primary purpose of the producer/consumer identify is to pair consumers with produces, especially for the purpose of routing between OpenLCB segments. Since these Well-Known Automatically-Routed events must always be routed, it could be argued that the identify events for these Well-Known Event Identifiers are redundant, but are retained for symmetry.

Though not explicitly required, it would be prudent for a manufacture to identify in a product's documentation as to which Well-Known Automatically Routed events it produces and consumes, and what are the resulting actions it takes when these events occur.

2.5.2.1 Emergency Off

The Emergency Off Event Identifier (01.00.00.00.00.00.FF.FF) is a request for a node to de-energize all of its outputs. A node receiving this event may continue to remain a powered participant of the OpenLCB bus, but may de-energize any outputs unrelated to maintaining OpenLCB communications. The meaning of de-energize is not prescribed for any given node, it is up to the node manufacturer and/or user to prescribe what, if anything, should happen in the node if it receives this event. For example:

- If the node is a DCC Power Station, it might disable its amplified DCC output.
- If the node is an accessory controlling turnout motors, it might remove power from the motors

A node may revert to its previous energized state following the reception of a Clear Emergency Off event (01.00.00.00.00.00.FF.FE).

A node that has recently joined the OpenLCB network is not expected to know about or react in any specifically prescribed way to the current Emergency Off status defined prior to the node joining the network until the next Emergency Off or Clear Emergency Off event is produced.

2.5.2.2 Emergency Stop

The Emergency Stop Event Identifier (01.00.00.00.00.00.FF.FD) is a request for a node to command all of its outputs to a safe state. A node receiving this event is not required to de-energize any of its outputs. The meaning of "safe state" is not prescribed for any given node, it is up to the node manufacturer and/or user to prescribe what, if anything, should happen in the node if it receives this event. For example:

- If the node is acting on behalf of one or more DCC trains, it might send the global emergency stop command onto the DCC signal bus.
- If the node is an accessory controlling turnout motors, it might do nothing, route its outputs to a manufacture default state, route its outputs to a user defined state, or something else altogether.

A node may revert to its previous non-Emergency Stop state following the reception of a Clear Emergency Stop event (01.00.00.00.00.00.FF.FC). A node that has recently joined the OpenLCB network is not expected to know about or react in any specifically prescribed way to the current Emergency Stop status defined prior to the node joining the network until the next Emergency Stop or Clear Emergency Stop event is produced.

2.5.2.3 Other Well-Known Automatically Routed

Other Well-Known Automatically Routed Event Identifiers not discussed here have their uses prescribed and/or discussed elsewhere. No assumptions should be made about the use of these Well-Known Automatically Routed Event Identifiers as prescribed by this document.

2.5.3 Well-Known

Unlike Node ID Based Event Identifiers, the Well-Known Event Identifiers are explicitly defined by the Event Identifiers Standard for a specific purpose. Gateways are not required to actively route these events to all segments, and may maintain a static or learned routing table for these events to prevent unnecessary propagation.

Though not explicitly required, it would be prudent for a manufacture to identify in a product's documentation as to which Well-Known events it produces and consumes and what are the resulting actions it takes when these events occur.

2.5.3.1 Duplicate Node ID

The Duplicate Node ID Detected event would typically be sent by a node that receives a packet from another node containing its own unique Node ID. The production of this event by a node is not explicitly required.

2.5.3.2 MERG CBUS

OpenLCB allocates a Node ID address space specifically for mapping MERG CBUS events into OpenLCB. CBUS events come in two types: long and short. A long event is uniquely identified by the CBUS Node ID and Event ID. A CBUS short event is translated into an OpenLCB event by using 0x0000 as the Node ID in the "CBUS Node ID" field of the OpenLCB event identifier. This is allowed because in the CBUS standard, node-id 0x0000 is not permitted.

CBUS events can also come in the form of a request. These requests should be translated into an Identify Producer on OpenLCB in order to solicit the response that the CBUS request is asking for. Two Identify Producer messages will have to be sent, one for the ON state range and another for an OFF state range, the response of which will need to be translated back for CBUS.

2.5.3.3 Other Well-Known

Other Well-Known Event Identifiers not discussed here have their uses prescribed and/or discussed elsewhere. No assumptions should be made about the use of these Well-Known Event Identifiers as prescribed by this document.

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....2
2.4 Format....3
2.5 Allocation....3
2.5.1 Node ID Based....3
2.5.2 Well-Known Automatically-Routed....3
2.5.2.1 Emergency Off....3
2.5.2.2 Emergency Stop....4
2.5.2.3 Other Well-Known Automatically Routed....4
2.5.3 Well-Known....4
2.5.3.1 Duplicate Node ID....4
2.5.3.2 MERG CBUS....5
2.5.3.3 Other Well-Known....5