OpenLCB Standard	
Traction Protocol	
Sep 2, 2022	Preliminary

1 Introduction (Informative)

This working note covers the Traction Protocol, the way that OpenLCB handles moving objects such as locomotives, engines, and other rolling stock.

2 Intended Use (Informative)

This protocol is for the long-term operation of really cool trains, not just “better control over DCC”. Still, DCC is an important test case, both with new control hardware (OpenLCB native throttles connected to an OpenLCB-native command station that's generating the DCC signal) and when connected to legacy throttles and/or command stations. Using legacy hardware within the OpenLCB environment is accomplished through a proxy and is implemented using this Traction Protocol and the Traction Proxy Protocol.

3 References and Context (Normative)

Node implementing the Train Protocol must implement:

- Message Transfer Protocol
- Event Transfer Protocol
- Memory Configuration Protocol (optional?) and Datagram Protocol it depends on
- CDI
- SNII and/or ACDI?

Float-16 is the half-precision numeric format defined by IEEE 754-2008. This is the format that the GNU tool chain's -mfp16-format=ieee flag and __fp16 type makes available on some CPU types.

For more information on format and presentation, see:

- OpenLCB Common Information Technical Note

3.1 Terminology

“DCC” refers to NMRA DCC; “Legacy” refers to all pre-existing protocols including DCC, TMCC, Marklin, DCS, etc.

“Trains”: For our purposes, Train is anything which can be independently controlled. In addition to a model of a prototype train from locomotive to caboose, it might be just single caboose, a set of lit & controlled passenger cars, a diesel MU lash up, or basically anything that can take an OpenLCB "decoder" or a DCC decoder with a legacy attachment.

“Train Node”: A Train is associated with a single, specific node. The Node ID is the fully-unique identifier for that Train.

“Throttles”: For the purposes of discussion, we draw a distinction between three kinds of throttles that a user might encounter:

- “Legacy Throttles” refers to throttles designed for use with extant DCC systems, e.g. a Digitrax DT402 or Lenz LH100.
- “Full-Featured Throttles” refers to full-featured native OpenLCB throttles with multi-line color screens and effectively unlimited processing power, e.g. a software throttle implemented on an iPad.
- “Simple Throttles” refers to throttles which are native OpenLCB nodes like Full-Featured Throttles, but which have more limited capabilities, e.g. no text display, a limited array of physical buttons, and constrained processing resources.

“Proxies”: In the long term, we expect that OpenLCB protocols will go all the way to the train. This has great advantages, because you're always in complete communication with the train, and don't have to worry about only being able to configure the train when it's on a service track, storing information somewhere else so that it can be retrieved while the train is moving, etc. But until radio or other technologies mature to the point that this is possible, "proxies" can be used as stand-ins for that capability. A throttle might communicate with a node that's serving as a proxy for the train, handling the communications, keeping track of status & configuration, etc. Out the back end of that proxy node is some other kind of communications, perhaps direct DCC or a connection to a legacy system that in turn makes DCC signals, or some other technology entirely. Due to the nature of those back side communications methods, the proxy may not be able to do everything that OpenLCB can, or only do some of it at certain times. The OpenLCB traction protocols need to take this reality into account. Another use case for proxies, discussed in detail below, is to provide the mechanism for consisting.

“Command Stations”: Existing DCC and other control systems use “command stations” to create a track signal for controlling the trains. Usually the command station is controlled from the user side by some other network, to which throttles and other interface devices are connected. OpenLCB, it it's native form, has no such concept. Devices, like throttles, that want to talk to a train do so directly. Only when working with legacy systems does the concept of a command station enter, and usually through the form of a proxy node that is acting for the Train.

“Consisting”: The running of multiple items together, e.g. three coupled engines, each with their own NodeID or DCC address, as a single locomotive. DCC systems provide this now in various ways and with various names.

“Listener”: A node may be interested in receiving a copy of all state changing messages from a given train node. Since throttle-to-train configuration is addressed, there is no network-level mechanism to guarantee that messages destined to the train node can be intercepted by the interested third party. The train node therefore may have a feature that allows other nodes to be registered, and the train shall forward the speed (and optionally function) setting messages to the registered nodes. A way to achieve consisting is to specify the consist members as listeners on the lead engine's train node, or to allocate a virtual node and specify all engines as listeners to that virtual node.

“Configuration”, “Functions”: Traditional DCC decoders provide “functions” for controlling accessories such as lights and sounds during operation, and provide a separate mechanism for doing long-term configuration via Configuration Variables (Cvs). OpenLCB makes the same distinction, providing access to “functions” via the traction protocol, while leaving “configuration” to the configuration protocol(s). The line between these is admittedly vague, and different node developers may implement some capability one way or the other. The general intent is that things that are changed in normal operation are considered functions, while things that are set once and forgotten are configuration.

4 Message Formats (Normative)

4.1 Defined Event IDs

IsTrain: 01.01.00.00.00.00.03.03

EmergencyStopAll: 01.01.00.00.00.00.FF.FF

4.2 Traction Control Command Message

MTI: Priority 1, index 15, modifier 3, addressed => MTI 0x05EB, CAN frame [195EBsss] fd dd

This message type and MTI is specific to traction control. The first byte of the content codes an “instruction”, which defines the rest of the format.

Instruction	**Byte 0**	**Byte 1**	**Byte 2**	**Byte 3**	**Byte 4**	**Byte 5**	**Byte 6**	**Byte 7**	**Byte 8**	**Byte 9**	**Byte 10**	**Byte 11**
Set Speed/Direction	0x00	Speed and direction as signed float16										
Set Function	0x01	Address			Value							
Emergency Stop	0x02											
Query Speeds	0x10											
Query Function	0x11	Address										
Controller Configuration	0x20	Assign Controller 0x01	Flags 0x01=Alias ID valid	Controller Node ID						Alias {Optional}		
		Release Controller 0x02	Flags 0x01=Alias ID valid	Controller Node ID						Alias {Optional}		
		Query Controller 0x03										
		Controller Changing Notify 0x04	Flags 0x01=Alias ID valid	New Requesting Controller Node ID						Alias {Optional}		
Listener Configuration	0x30	Attach Node or update flags 0x01	Flags 0x01=Alias ID valid 0x02=Rev direction 0x04=Link F0 0x08=Link Fn 0x80=Hide	Listener Node ID						Alias {Optional}		
		Detach Node 0x02	Flags 0x01=Alias ID valid	Listener Node ID						Alias {Optional}		
		Query Nodes 0x03	Listener index {optional}									

Instruction	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10	Byte 11
Traction Management	0x40	Reserve 0x01										
		Release 0x02										
		No-op 0x03										

The Set Function instruction uses a three-byte address for brevity; it's to be interpreted with a high byte of zero to make a four byte address in the function memory space (0xFA).

4.3 Traction Control Reply Message

MTI: Priority 0, index 15, modifier 1, addressed => MTI 0x01E9, CAN frame [191E9sss] fd dd

Instruction	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10
Query Speeds Reply	0x10	Set Speed		Status Bit 0: 1=E-Stop.	Commanded Speed		Actual Speed				
Query Function Reply	0x11	Address			Value						
Controller Configuration Reply	0x20	Assign Controller Reply 0x01	Result: 0 == OK Non-zero == Failed Fail Code: Bit 0=Assigned Controller Refused Connection Bit 1 = Train Refused Connection								
		Query Controller Reply 0x03	Flags 0x01=Alias ID valid	Active Controller (0.0.0.0.0.0 if no controller active)						Alias ID {Optional}	
		Controller Changed Notify Reply 0x04	Result: 0 == OK Non-zero == Reject								
Listener Configuration Reply	0x30	Attach Node Reply 0x01	Node ID						Reply Code		
		Detach Node Reply 0x02	Node ID						Reply Code		
		Query Node Reply 0x03	Node count	Node index {opt}	Flags {opt}	Node ID {optional}					

Instruction	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8	Byte 9	Byte 10
Traction Management Reply	0x40	Reserve Reply 0x01	Result: 0 == OK Non-zero == Failed								
		Heartbeat Request 0x03	Timeout in seconds								

The Query Speed/Direction reply is almost in the Set Speed/Direction format, with the addition of the two additional speeds. If a node cannot provide any of those three speeds, it should use float16 NaN (not a number) 0xFFFF. “Set Speed” is the most recent speed received in a Set Speed/Direction instruction, or ±0 if an E-Stop command was received after the last Set Speed/Direction command. “Commanded Speed” is the speed that the traction control is currently attempting to move, taking into account momentum and any other control modifiers. “Actual Speed” is the current measured speed of the locomotive. There is no accuracy guarantee for Actual Speed.

5 States (Normative)

Full OpenLCB nodes:

- Set Speed – The speed set by a throttle, the content of the most recent “set speed” instruction
- Commanded Speed – the current speed that is intended. This may lag behind the set speed intentionally due to momentum settings, etc.
- Current Speed – a physical state, the speed at which the object is currently moving

Emergency stop is not a state.

6 Interactions (Normative)

6.1 Train Assignment

Controller configuration maintains the node that is controlling the train node. In order to operate a train the controller must be first assigned. A train node shall not respond to a traction message if it not defined as the current active Controller. Once a controller is finished operating a train it should, but is not required, to Release the active Controller. One place this may not be done is if the controller is being unplugged to be moved to a new location on the layout. The benefit of releasing the train is to allow the next controller to be connected to the train without the handshaking required to take a train from an existing active Controller. This is explained in the messaging diagram later in this document. A Controller node may be send an unsolicited message if it is currently assigned to a train node and another Controller node attempts to assign itself to the train. The new the Controller node requests to be Assigned to the train one of several actions may occur.

1) The train may block the request and return false to the Assign request
2) The train forwards a Controller Changed Notify message to the currently assigned controller node and that controller node returns false in which the train node returns false with a fail code of Controller Refused to the Assign request. The active controller may, but is not required to, indicate to the user it has been asked to release the train node.
3) The train forwards a Controller Changed Notify message to the currently assigned controller node and that controller node return true. The active controller may, but is not required to, indicate to the user it is loosing its position as the active controller of the train node. In turn the train node returns true to the Assign request and the train node sets the new controller as the active controller.

A node may Query the active node from a train node. If no controller is assigned the train nodes returns a Node ID of 0.0.0.0.0.0.

Traction Management locks/releases the train node such that interactions that require multiple messages between 2 or more nodes are guaranteed to be completed before another node can set (modify) the configuration on the train node. These messages are not needed for normal traction operations such as speed and function set/query. If the node attempting to set attributes of the train is not the active node the train node will ignore the request and as such locking/releasing is not required.

6.2 Emergency Stop

Receipt of the Emergency Stop instruction stops the locomotive as fast as possible. This sets the set speed to zero (preserving existing direction) and the commanded speed to zero (preserving existing direction) regardless of any momentum, BEMF or other operations with the train node.

Emergency stop is not a specific state. The next Set Speed/Direction instruction will act immediately to change the set speed, and start the commanded speed and actual speed moving toward that set speed.

6.3 Function Operation

Function values are stored in the 0xF9 memory space. They are written using the memory configuration protocol or using the Set Function instruction.

6.4 Train Configuration

Trains are OpenLCB nodes just like any other. As such, the Memory Configuration protocol can be used to configure them, the Configuration Description Information system can be used to make that process user friendly, etc. There's nothing traction-specific in these techniques, which are available any time that the train node is connected to the OpenLCB.

6.5 Train Identification

OpenLCB Train nodes use:

- The Event Transport protocol to locate Train nodes
- PIP for enquiry about the support

- SNII and/or Memory configuration, CDI & ACDI for identification of a specific train node.

Trains are OpenLCB nodes just like any other. As such, they can take part in protocols such as Node Verification and Simple Node Information which allows other nodes to learn about them.

Train Acquisition Protocol is necessary because the train operator doesn't want to pick up a throttle and enter “06.011.00.02.1F.2D” (a node ID), or even “110 Long” (a DCC address), but rather just pick the desired locomotive from a list of those available. (A throttle should still allow the operator to directly enter the address, when that's what the operator wants to do.)

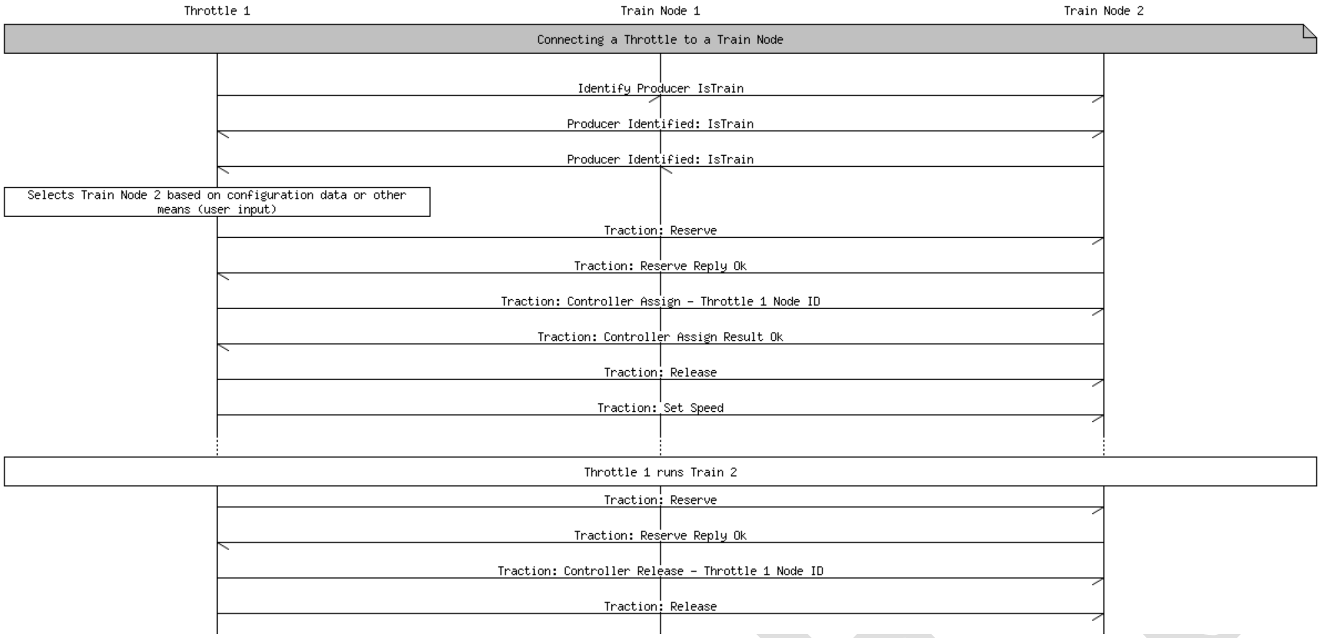
The train acquisition process simplifies locating desired train nodes so that small hand-held throttle nodes can efficiently take part. It does this using several approaches, which can be used as needed by throttles:

- Events are used to announce the existence and status of Train nodes
- Train nodes will generally implement the Simple Node Information protocol so that throttles can get basic, user-readable identification from them
- A search protocol is (being) defined to make it possible to locate individual Train nodes without having to read information from all of them

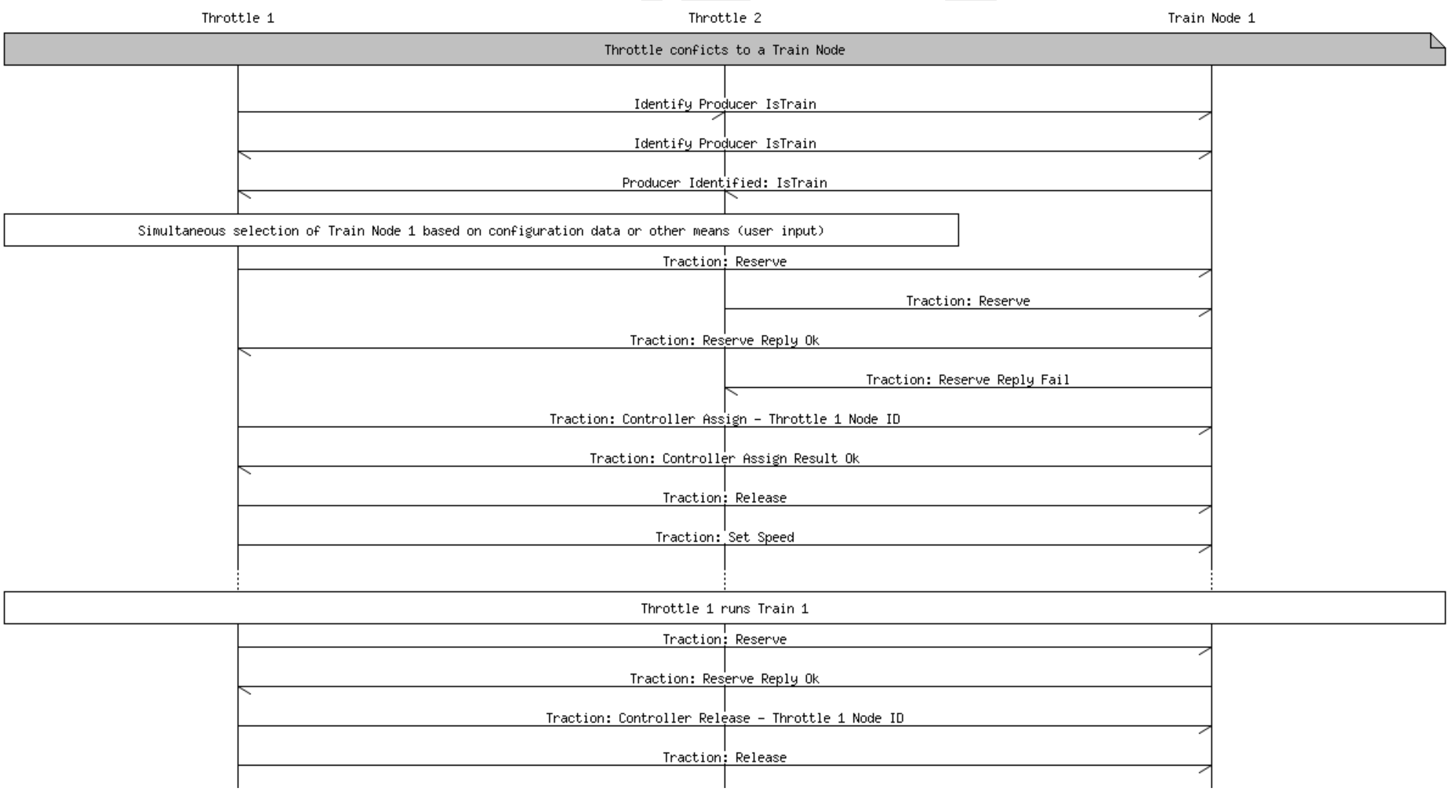
So that other nodes can find them, Train nodes must produce the well-known reserved event 01.01.00.00.00.00.03.03. This means they must produce that event in a Producer/Consumer Event Report message when they power up, and reply to requests for producers of that event. An IdentifyProducer request will therefore find all the train nodes on the OpenLCB, and further protocols can be used to get additional information on the individual Train nodes it locates.

SNIP will be used to carry both manufacturer-provided and user-provided information about the particular train node. In particular, the user (Node) Name and (Node) Description fields are to be used to hold train identification information that can be retrieved and presented by throttles for selection.

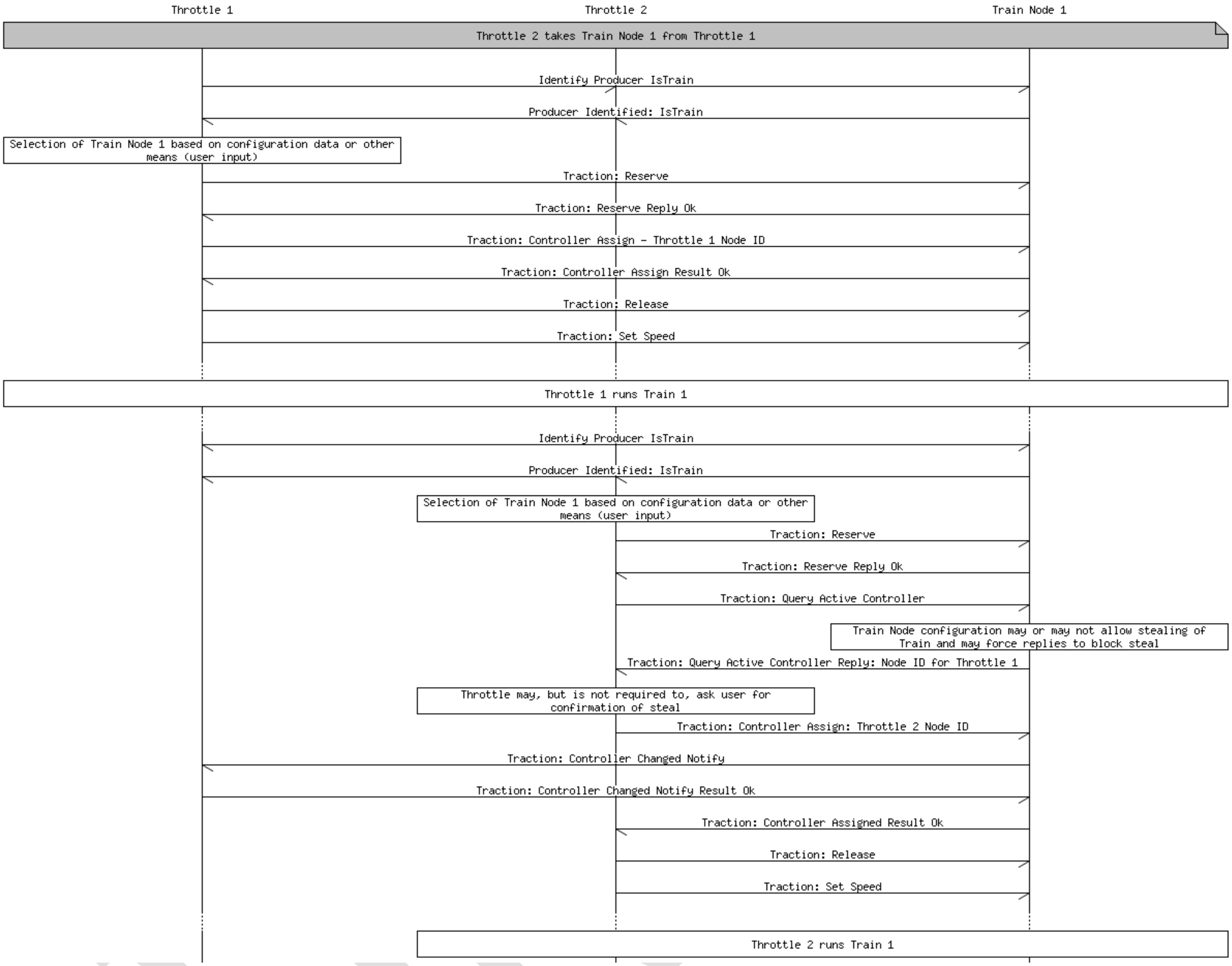
6.6 Basic Throttle and Train Connection



6.7 Conflicting Throttles



6.8 Throttle to Throttle Successful Hand-over (steal)



6.9 Listeners

When listeners are configured, the incoming state changing traction messages are forwarded by the train node to all the listener nodes. The forwarded message shall be a Traction Control Request message.

If the source node of the message is on the listener list, the message is not forwarded to avoid message loops.

The listener flags define what messages shall be forwarded:

- Set speed message is always forwarded. If the flag “Rev direction” is set for the listener, the direction of the forwarded speed is flipped.
- Set function message with function number = 0 is forwarded iff the “Link F0” flag is set on the listener configuration.

- Set function messages for all other function numbers shall be forwarded iff the “Link Fn” flag is set on the listener configuration.

To add or remove a listener, the Attach Node or Detach Node message shall be sent to the target train node. The response shall contain the same node ID and a 2-byte OpenLCB Error Code. The following error codes may be helpful:

- 0x1030: Permanent error, Not found. Useful if the caller tried to remove a node that is not attached as a listener.
- 0x1032: Permanent error, Already Exists. Useful if the caller tried to add a node as a listener to itself.

If a caller wants to update the flags of a specific listener, the Attach Node message shall be sent with the same node ID and the new flags.

To query the listeners from a train node, a Listener configuration Query Nodes message shall be sent to the train node. The response message contains the number of listeners currently configured. Listeners are indexed from 0 to count-1. Sending the Query Nodes message with the index specified, the train node will return the information for that specific listener, including Node ID and the flags specifying the forwarding options. If the requested index is out of bounds, or no index is requested, then the response will be short (no node information).

User interfaces that show the listeners shall not show listener nodes that are marked with the flag “Hide”. This flag is intended to be set for controlling throttles and train automation systems.

To discover and retrieve all listeners from a train node, multiple query messages are required. This presents a race condition when a third node changes the listener configuration during the process. It is not specified how the listener indexes change by attaching or detaching a listener. The querying node can recognize that their information is out of date by observing a change in the count of listeners compared to the expected number of listeners and restart the enumeration process.

6.10 Heartbeat

The Heartbeat Request may be sent by a train node to the currently active Controller node at the discretion of the train node. The argument is a deadline in seconds for the Controller node to reply. The Controller node may reply with any control or query command, or a No-op command. If there is no command received within the deadline, the train node may elect to stop the train.

Trains shall not initiate a heartbeat request if the last Set Speed is zero (including E-stop).

Trains shall accept any command or query sent from the controller node to the train node to clear the heartbeat request. If the controller node does not have anything to command or query, the Traction Management No-op request may be used to clear the heartbeat.

If a train does not receive any command or query from the Controller node within the deadline presented in the heartbeat request, the train shall interpret that as a Set Speed 0 command. This command shall be forwarded to any registered listeners at the same time.

In case there is no assigned Controller node, there is no connection to check with heartbeats, and the train shall continue operating as last commanded. Controllers can choose to release themselves from the train if they wish to not receive heartbeats; this is recommended during any orderly shutdown of a controller or else the train will stop after the heartbeat period.

7 Memory Spaces

7.1 Configuration Information

The configuration memory space holds the configuration of the train, such as how functions will work, how speed in scale meters/second will control motor operation, etc.

7.2 CDI

7.3 Function Information 0xF9

Functions, such as lights and sounds, can be operated by the Traction Control Set Function instruction, and their current value can be retrieved via the Traction Control Query Function instruction.

The NMRA 9.2.1 Recommended Practice describes DCC as having three separate sets of "functions". The OpenLCB 0xF9 space is allocated to cover all these by using the third byte of the address as a selector.

Type	Low Address	High Address	Values
F0-F28	0x0 00 00	0x0 00 1C	A non-zero value indicates "ON", a zero value is "OFF".
Binary State Controls (full space)	0x1 00 00	0x1 7F FF	"
Binary State Controls (short space, if separate)	0x2 00 00	0x2 00 7F	"
Analog Outputs	0x3 00 00	0x3 00 FF	

The default Function Definition Information, when nothing is known about a particular decoder's capabilities, will just describe these as above. If the node serving the DCC decoder information knows more about the particular decoder's capabilities and/or preferred labeling, it can provide more information via the Function Definition Information.

7.4 Function Definition Information FDI 0xFA

“Function Definition Information”, similar in intent to Configuration Definition Information (CDI) is stored in XML format in address space 0xFA to provide user-oriented context. That includes:

- Memory layout of the function values, allowing for multiple data types from binary (one and off for lights) through integer values (for e.g. sound intensities) and strings (sign displays?).
- Function naming, so that a throttle can display useful names to the user such as "Bell", "Coupler Clank" and "Master Volume".

Table of Contents