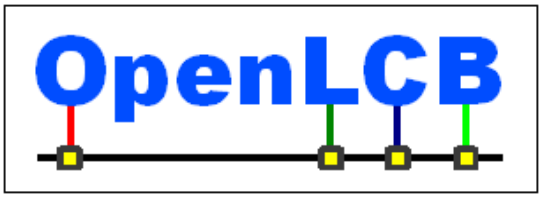


OpenLCB Working Note	
Traction Proxy Protocol	
Mar 17, 2014	Preliminary

1 Introduction

This working note covers the Traction Proxy Protocol, the way that OpenLCB interfaces with legacy control systems for objects such as locomotives, engines, and other rolling stock.

A Working Note is an intermediate step in the documentation process. It gathers together the content from various informal development documents, discussions, etc into a single place. One or more Working Notes form the basic for the next step, which is one or more Standard/TechNote pairs.

This protocol is for the bridging between legacy control system such as DCC, Marklin among others and Open LCB.

1.1 Terminology

"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, "proxy nodes" 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.

"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.

"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.

1.2 Served Use Cases

1.2.1 Legacy Train on New Layout

Jim takes his DCC-equipped train to Bill's OpenLCB- and DCC-equipped layout and puts it on the track. He picks up a throttle, hits a few keys, sees his train, selects it and starts to run it.

As an alternative, Jim takes his DCC-equipped locomotive to the layout, puts it on the track, enters the DCC address into a throttle, and starts to run it.

1.3 Unserved Use Cases

1.3.1 Multiple Independent Command Stations

Large modular layouts use multiple command stations to increase the effective bandwidth of the DCC bus. This is not an explicitly supported use case in the current work. Future work may make this possible as an extension.

1.3.2 Improved Legacy Addressing

DCC systems cannot run two locomotives with the same DCC address at the same time. This will still be true when running DCC-equipped locomotives via OpenLCB protocols to the command station.

2 Specified Sections

This is the usual section organization for a Technical Note, to accumulate the Standard and Technical Note content in its eventual order.

2.1 Introduction

Note that this section of the Standard is informative, not normative.

2.2 Intended Use

Note that this section of the Standard is informative, not normative.

2.3 Reference and Context

NMRA S9.2 and NMRA RP9.2.1 define the formats for DCC addresses

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.

2.4 Message Formats

2.4.1 Defined Event Ids

Proxy types are defined as well known ID values. This was done to allow fast and easy identification of the technology for simple bridges from the legacy technology to OLCB. Another way is to have a single “IsProxy” event then a call into this protocol to extract the legacy technologies supported. This is an acceptable way to do it if the node looking for the proxy has lots of RAM to allow it to collect all possible responses (remember in multiple command station layouts each command station may identify itself as a Proxy). This method allows a much cleaner implementation at the bridge level to find the DCC (or other legacy system) proxy.

Event Type	Well Known Event ID
IsDCCProxy	01.01.00.00.00.00.03.04
IsDCProxy	01.01.00.00.00.00.03.05
IsMärklinDigitalProxy	01.01.00.00.00.00.03.06

IsMärklinDELTAProxy	01.01.00.00.00.00.03.07
IsMärklinESUDigitalProxy	01.01.00.00.00.00.03.08
IsSelectrixProxy	01.01.00.00.00.00.03.09
IsMTHDCSProxy	01.01.00.00.00.00.03.10
IsLionelTMCCProxy	01.01.00.00.00.00.03.11
EmergencyStopAll	01.01.00.00.00.00.FF.FF

2.4.2 Traction Proxy Control Command Message

MTI: Priority 0, index 15, modifier 2, addressed => MTI 0x01FA, CAN frame [191EAsss] fd dd

The MTI modifiers are chosen to have the reply a higher priority than the request, ensuring replies to repeated instructions are always possible. The same priority and index are used for command and reply messages, changing only the modifier, to use less of the high-priority MTI space which is a scarce resource.

This message type and MTI is specific to traction proxy control. The first byte of the content codes an "instruction", which defines the rest of the format. The instruction codes were selected with the high nibble representing protocol (0x00 for DCC legacy; others reserved)

Dedicated OpenLCB messages are defined for traction proxy control, instead of using datagrams, so that they have higher priority. Given the small size of these messages, they also use less bandwidth than datagrams.

Instruction	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Allocate DCC Proxy	0x01	DCC Address AA.AA		Speed steps (14, 28, 128)				
Allocate DC Proxy	0x02							
Allocate Märklin Digital Proxy	0x03							
Allocate Märklin DELTA Proxy	0x04							
Allocate Märklin Digital (2004 w/ESU) Proxy	0x05							
Allocate Selectrix Proxy	0x06							
Allocate MTH DCS Proxy	0x07							
Allocate Lionel TMCC Proxy	0x08							
Manage Proxy	0x80	Reserve 0x01						
		Release 0x02						

2.4.3 Traction Proxy Control Reply Message

MTI: Priority 0, index 15, modifier 0, addressed => MTI 0x01F8, CAN frame [191E8sss] fd dd

Higher priority to ensure can be sent immediately over Traction Proxy Control Command messages.
115 Coding and structure similar.

Instruction	Byte 0	Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7
Allocate Proxy Reply	0x01 .. 0x7F	Allocated Node ID						Reply Code
Manage Proxy Reply	0x80	Reserve Reply 0x01	Result: 0 == OK Non-zero == Failed					

2.4.3.1 Proxy-Specific

Allocate reply code: Zero means OK

Error codes need to be defined.

- Allocation Failed, temporary (try again) handles race conditions *(but we need to look carefully at that race condition in practice: With the current structure, does it end up adding two DCC addresses to the single proxy, instead of reserving it? Probably need to separate out reservation-for-configuration. In general, two different things like "assign" and "reserve" should be handled by two separate things)*
- Allocation Failed, permanent (out of proxies, etc)
- Allocation Type Unsupported

The configuration memory space holds the configuration for the proxy.

2.5 Interactions

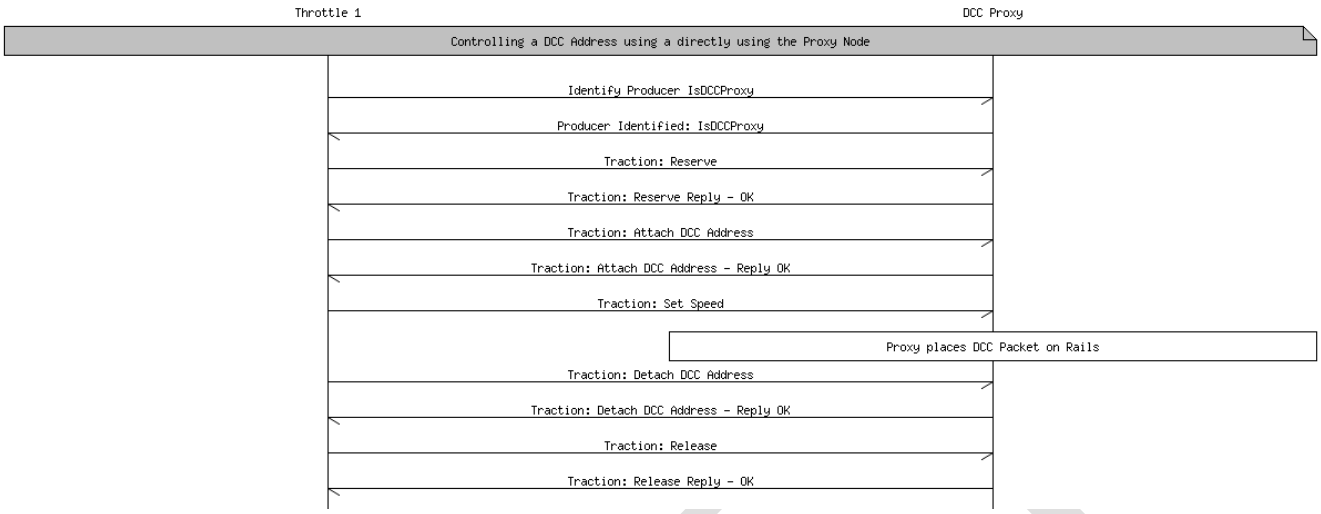
2.5.1 Emergency Stop

Receipt of the Emergency Stop instruction stops the proxy 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 for all legacy trains under its control.

Emergency stop is not 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.

2.5.2 Basic Proxy Control

Eventhough it is not recommended it is possible to directly access the proxy to control a legacy system. The Throttle node must implement the Traction and Traction Proxy Protocols as must the node that is bridging OLCB with the DCC protocol.

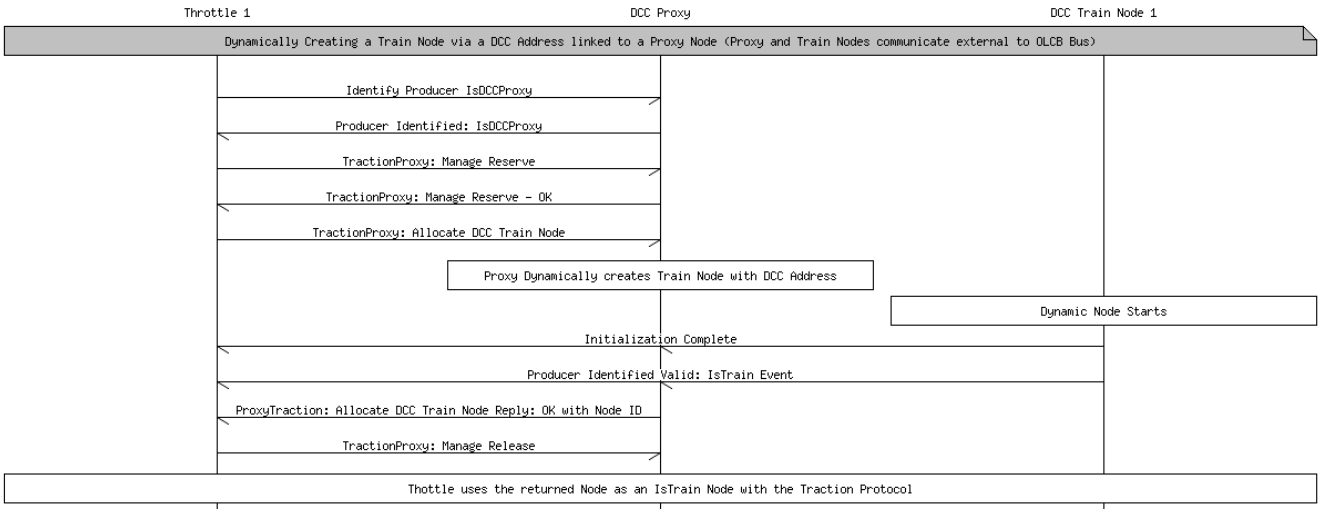


Directly accessing a legacy system through the Proxy.

While this is adequate for a minimalist system it does not allow for the additional user features a system that creates a node for each train that is being run on the layout.

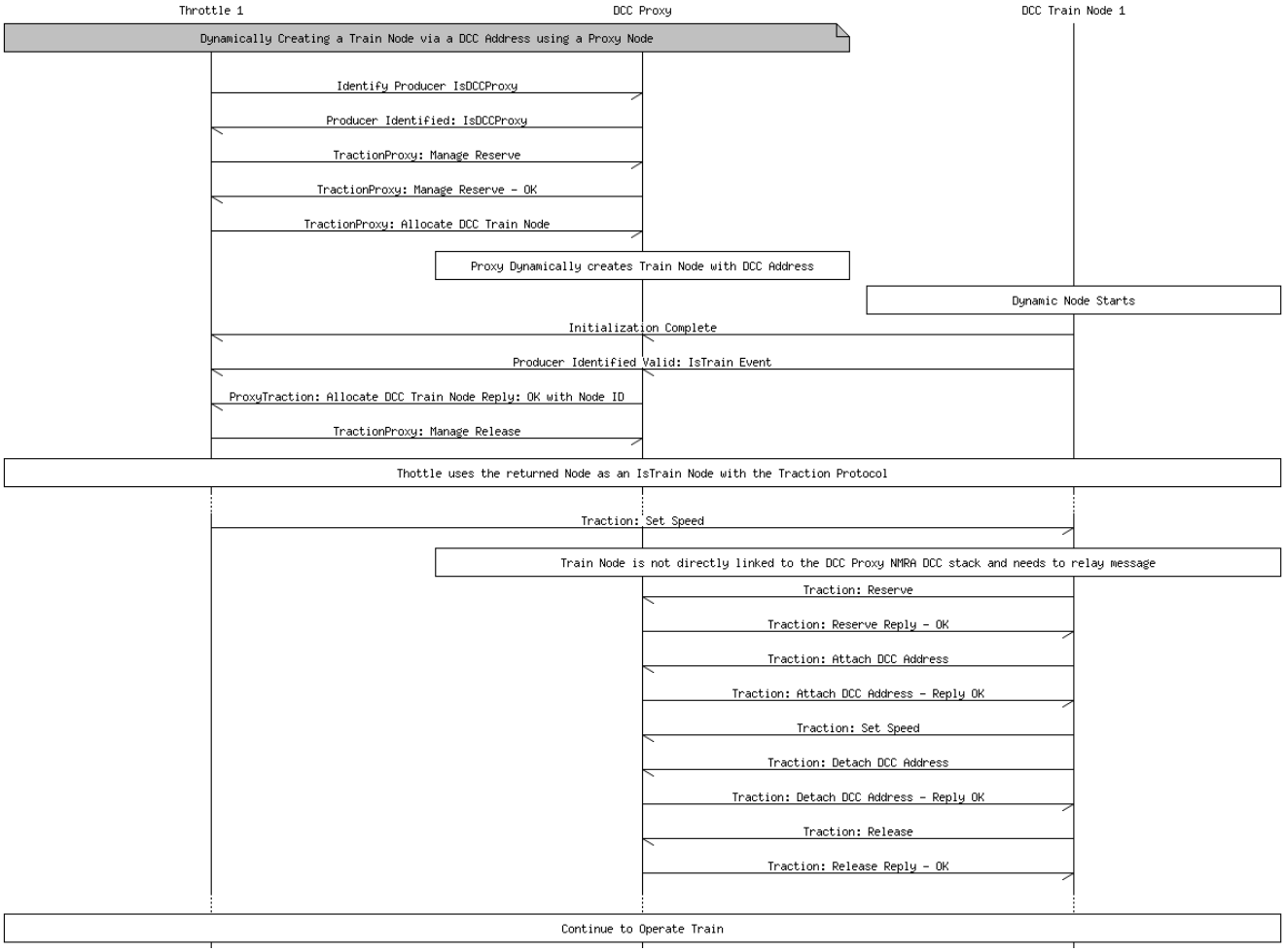
2.5.3 Allocate Train Node – Direct Legacy System Access

The minimal recommended traction proxy control system should create a node that is linked to the legacy system through a means other OpenLCB. This could be a command station that maintains a database of virtual nodes that have direct access to the DCC protocol stack, through a back end proprietary network or other means. The Throttle node is this case will use the Traction Proxy Protocol to allocate a train node that is linked to the underlying legacy technology through some manufacturer specific way. The result of this allocation will be a node that, from the throttles point of view, is a pure OpenLCB train (IsTrain) and can be controlled through the Traction Protocol.

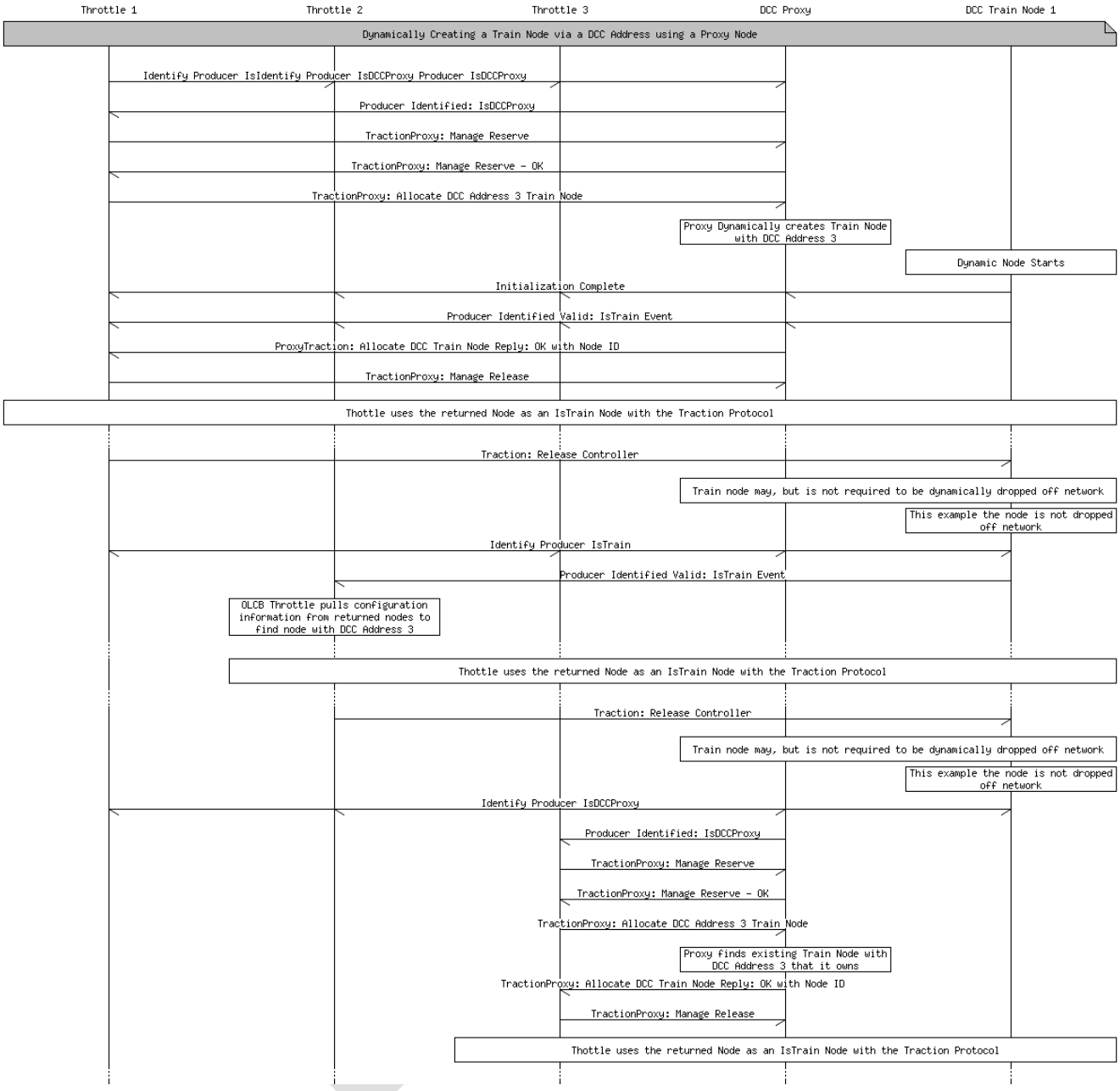


2.5.4 Allocate Train Node – OpenLCB Legacy System Access

It is not required that the allocated node have a backend connection with the underlying legacy technology. It is possible for the allocated node to appear as a pure OpenLCB IsTrain (Traction Protocol) to the throttle and the train node relays the commands to the legacy proxy without the need for the Throttle to know anything about the proxy. In the case of using the Traction Proxy protocol this may seem redundant but this is only one way of finding a train to run with the throttle. Another way would be to send and Identify Producers with IsTrain and have a database with DCC trains that have metadata associated with them. The DCC address would be already defined in this metadata. The Throttle could select the train by means other than the DCC address. In this scenario the throttle will not need to know anything about the proxy and will just start controlling the train via the pure OpenLCB Traction Protocol but the train node will relay the command to the proxy as it has been preloaded with the information to make this connection.

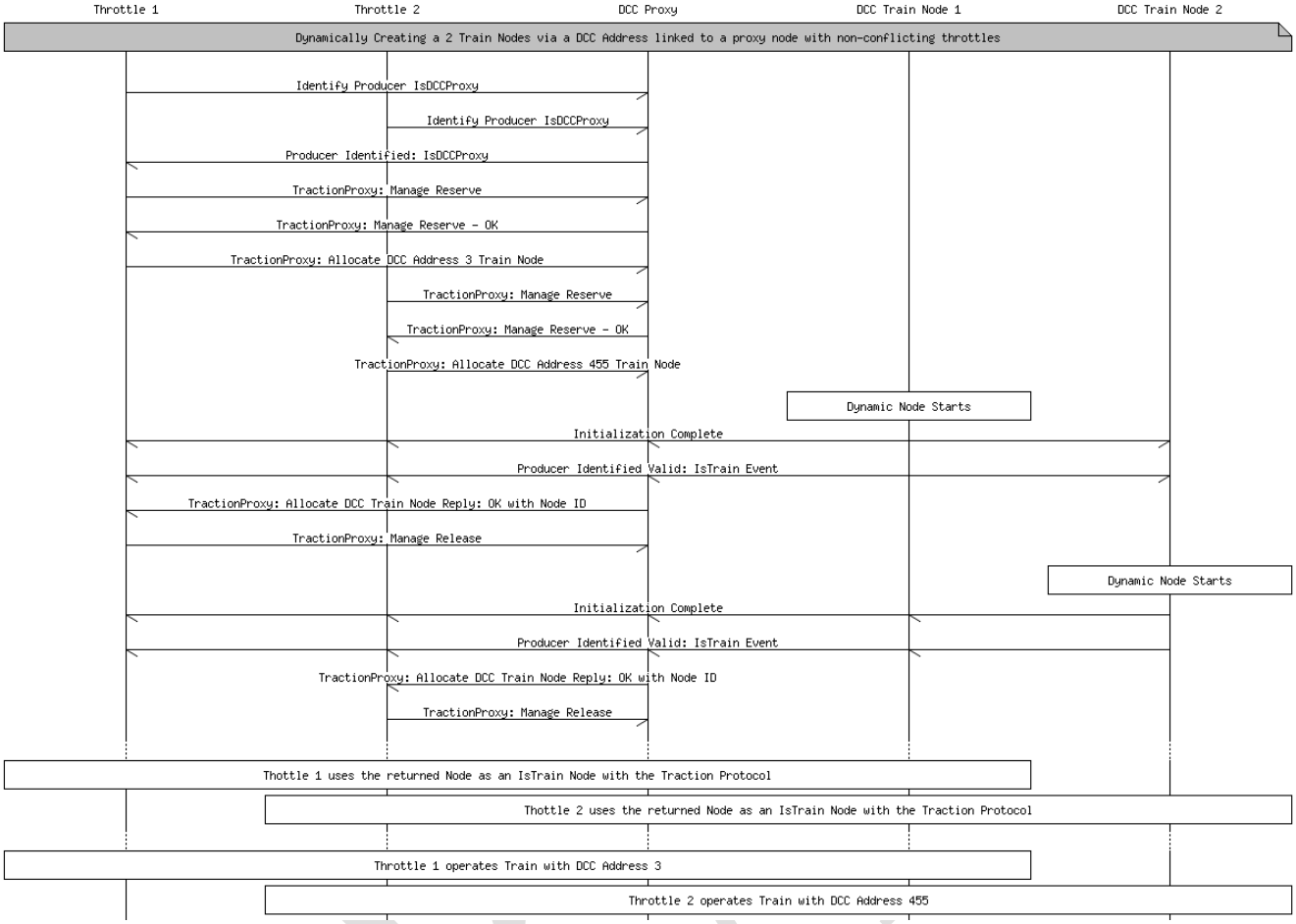


When the throttle no longer wants to operate the train it may deallocate it. The system may, but is not required to remove the node from the network. If the node is not removed from the network it may be recycled and used when another throttle allocates that DCC address.

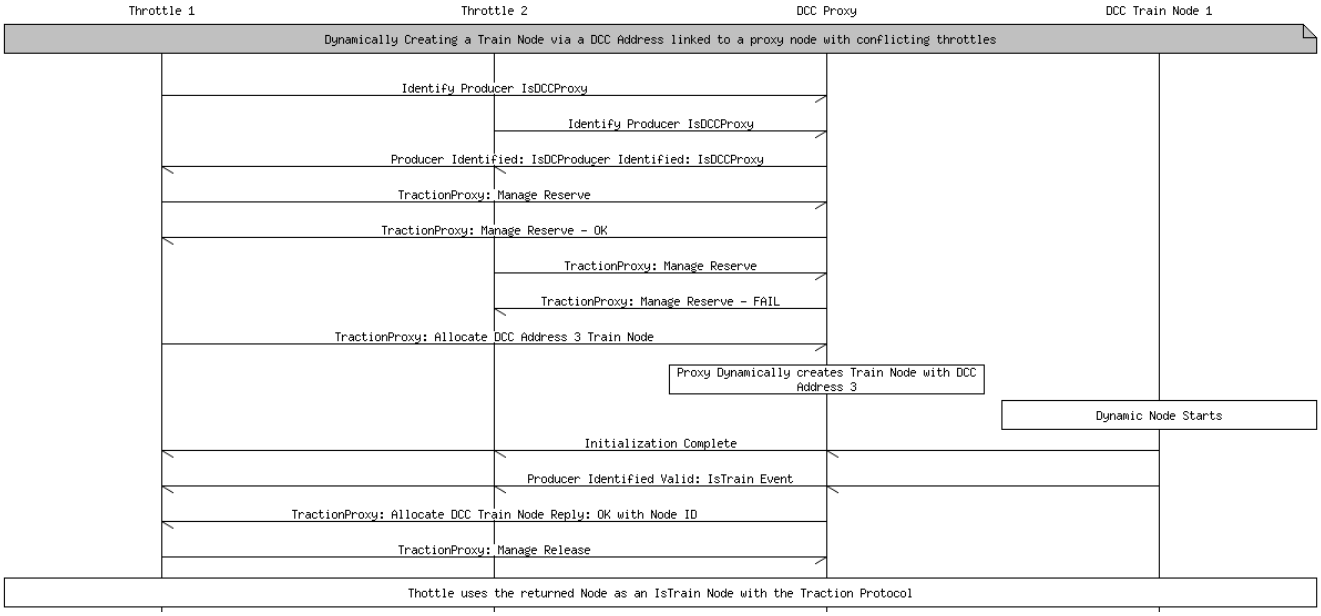


175

2.5.5 Multiple Throttle: Non Conflicting



2.5.6 Multiple Throttles: Conflicting



3 Legacy DCC Configuration

For DCC trains, there are two parts:

- CV access (with the added complexity of indexed and double-indexed Cvs)
- Programmer (main, programming track) control.

In the special case of legacy equipment, for example DCC locomotives which are configured via CV values, the situation is a little more complex. The train may need to be on a special section of "programmer" track, or only certain values can be changed in certain ways, etc. These legacy devices may have to use a more restricted form of the configuration protocols. For more information, see the page on DCC CV Programming. Non-DCC roster information could be separately stored in the proxy node serving a particular locomotive.

OpenLCB has a protocol for handling configuration of nodes. It should work fine for native OpenLCB rolling stock. Can/should it be adapted to handle the way existing DCC locomotives are configured through Cvs?

For CVs, one idea that we'd have a memory space that maps straight to the decoder CV space.

That would allow decoder-specific CDI (if that was available) to customize what's presented. Like DecoderPro, the user wouldn't have to deal with CV 111, but rather with "Motor multisnarb angle offset" or whatever the user-fiendish manufacturer decides to provide as options.

There are two remaining issues. First, is handling "indexed" CVs. (Ones where you write 12 to CV 51, then 8 to CV52, and then CV54 is the value you want to read/write). QSI is the only extensive user of these, but they're starting to appear in other places too.

There are a couple ways to handle it. First, we could just map it as is: The configuring node would have to explicitly do those reads and writes. But that's a mess, well outside the OpenLCB model, and I've spent way to much time debugging weird failures with that.

A better approach, I think, is to use the large address space. E.g. CV 59 is found at 0x00 00 00 3B, while the one I mentioned above is found at 0x01 00 0C 08. (The 0x01 tells how to decode the address space, e.g. which CV the 2nd byte is written to, which the 3rd byte is written to. We'd have to extend CDI to carry that info, but it's within reason.

The other issue is packed CVs, e.g. mapping parts of one or more CVs to a single "variable". This could be just a single bit in one CV, or something more complicated split across two (like long addresses). I'm not sure how to map those as a general case. For bits, I'd suggest another mapping trick, where some other part of the space is actually bit mapped:

0xFF 00 53 3C

is the middle bits (the 3C mask) of CV 53, and the 0xFF is just an arbitrary key for this. That doesn't work with noncontiguous splits across generic CVs, though, as that takes a lot to configure.

(And don't get me started on CV1/CV29 sequencing; I think we just ignore that entirely at the CDI level & build it into the gateway to DCC)

In the end, these legacy pains can't really be avoided. We have to croft the OpenLCB protocols to do a reasonable job with them, but I don't think we have to go nearly as far into the weird special cases as e.g. DecoderPro does.

4 Legacy Throttles and Throttle Busses

The most straight-forward way to allow use of existing legacy DCC throttles and systems is to connect the DCC output of their command station to an OpenLCB node. That node then extracts the speed, function, configuration, etc information from the DCC stream and transmits it either (1) directly to an OpenLCB DCC command station or (2) as Traction Protocol Messages after conversion.

Method (1) works for attaching the input (throttle) and output (DCC train driving) parts of a legacy control system to a new OpenLCB core, which in turn allows OpenLCB control of the layout.

Method (2) works more generally, and will be more valuable once full OpenLCB trains become available.

5 To Do

More clarity on proxy access to DCC CVs. Discuss selection of programming modes. Addressing.

Need to devise a solution for two additional use cases: “Two throttles attempt to access a single Train; this is an error and isn't allowed.” and “Two throttles share access to a single Train, with speed & function commands accepted from both and reflected back to each other”

Multiple DCC proxies can be attached to the same DCC address, for example to represent two different locomotives with address 1234, holding the options and configuration for each one separately. (Only one can run at a time, of course) What's needed to do this right? The user interaction might be “I want to run 1234” “This one, with name Foo, is defined, is that the right one?” “No, define a new one” “OK, a new proxy exists as node 12.34.56.78.9A”, “OK, run that one”. What else is needed? Error detection?

For certain legacy system throttle busses, e.g. perhaps LocoNet or XPressNet, it may also be possible to attach the throttle bus to an OpenLCB node for translation. That node then acts as an intermediary between Legacy Throttles and the OpenLCB, and can operate in either of the modes described above.

Table of Contents

