OpenLCB Technical Note	
Train Search Protocol	
Jun 24, 2024	Draft

1 Introduction

This Technical Note provides background information and commentary to the OpenLCB Train Search Protocol Standard (in short 'the Standard'). This document is not intended to be read standalone, but in parallel to the Standard.

The capitalized terms Train Node, Throttle Node, and Command Station refer to the definitions according to the Section 'Interactions' in the Standard.

1.1 Background

1.1.1 Relation to Train Control Protocol

The OpenLCB Network is a peer-to-peer network where Nodes connected to the network can communicate with each other using the OpenLCB Network Protocols. These protocols are organized around the different OSI layers, and to achieve an end-to-end use-case a multitude of protocols are typically used, from different layers[1], or in a complementary manner[2].

Some of these protocols are **broadcast**, meaning that all Nodes receive the messages of the interaction; other protocols are **addressed**, meaning the interaction takes place between two specific Nodes, and other Nodes do not receive the messages of the interaction. In order for an addressed interaction to take place, the initiating Node needs to know the Node ID of the target Node. This generally requires some broadcast protocol using which the initiating Node can discover the possible target Nodes, and present these on a user interface to the user to select.

Considering the interaction of a user using an OpenLCB-connected Throttle to communicate to an OpenLCB-connected Train, or an OpenLCB-connected Command Station, there are two protocols that are part of this interaction:

- The Train Control Protocol defines the addressed interactions related to driving a train, such as setting speed and direction, or operating functions. It takes place between one Throttle and one or more Train Node(s) that are controlled.
- The Train Search Protocol is a broadcast protocol that specifies interactions needed for the Throttle to identify the Node ID of the Train Node that the user intends to drive.

1.1.2 Relation to Track Protocols

Trains may or may not be equipped with the necessary technology to directly participate in the OpenLCB Network. For now and the foreseeable future, an overwhelming majority of digitally controlled trains will be using a mobile decoder that operates using an existing standard or

[1]Example: The Memory Configuration Protocol is using the Datagram Protocol as a supporting layer.
[2]Example: for end users to configure an OpenLCB Node, the Memory Configuration Protocol is expanded with the Configuration Description Information protocol to describe what configuration settings are available at which memory offsets.

proprietary method of transmitting commands to the mobile decoder through the track, such as the NMRA® DCC® protocol suite.

In such a setting, there is a centralized component called a Command Station which is responsible for generating the track protocol signal. This device has to receive commands from the Throttles, and translate them to remote control the mobile decoders built into the trains rolling on the track. The Command Station is then responsible for participating on the OpenLCB Network and representing the remote trains. The Train Control Protocol requires each train to be a separate OpenLCB Node with its own Node ID, therefore the one Command Station device will be operating a number of Virtual Nodes, one per actively controlled train. All of these Virtual Train Nodes participate in the Train Search Protocol so that Throttles can discover, find and operate the trains connected to the track.

An important question then becomes: how does the Command Station know when the user intends to address a new train on the track that does not yet have a matching Virtual Train Node. The Train Search Protocol addresses this use-case as well.

1.1.3 Relation to other Search and Discovery Protocols

Note that the existence of Train Search Protocol does not exclude other methods and standards, including some not yet developed, for the same or similar purposes. There are also possibilities of using less capable existing Standards to achieve similar results (covering some but not all use-cases), albeit at a higher cost for the Throttle Node in terms of the necessary resources (above all RAM) and higher number of network messages to send and keep track of.

Some of these options are described in Section 3 (Alternatives Considered) below.

1.2 Served Use Cases

- The User is holding a Throttle connected to the OpenLCB Network. The Throttle has a numeric keypad and no screen. The user types in the four digit cab number they see on the locomotive and presses the Enter key. The Throttle uses the Train Search Protocol to find the matching OpenLCB Node. Then the throttle uses the Train Control Protocol to connect to that Train Node, and the user is able to drive the train.
  Note that this use-case presents the same way if the train itself has a radio-equipped decoder and is directly participating on the OpenLCB network; and also if the train has a DCC decoder and represented by a Command Station on the OpenLCB Network.
- The user owns a locomotive with road number 474 014 with a legacy track protocol decoder. The user does not remember what address is assigned to the decoder, but there is a Roster (locomotive database) in the Command Station, which has a record of the track address, and the name “Re 474 014” is assigned to this record. The user has a Throttle with a numeric keypad and a screen, and types in 474 on the keypad. The Throttle uses the Train Search Protocol to look for matching entries on the network. The Command Station searches through the Roster, instantiates the matching Virtual Nodes, provides the list of matching Node IDs to the Throttle. The Throttle queries the user-visible names of those Nodes using the Simple Node Information Protocol and presents a list to the user to select from. The user selects “Re 474 014”, and starts driving the train.
- The user purchased a new locomotive which came with a Märklin®-Motorola format mobile decoder with address 72. The user puts it onto the track, types in 72 on the throttle, selects the

Märklin-Motorola track protocol using option keys, and presses Enter. The user is in control of the locomotive.

- A friend comes over and brings a DCC-equipped locomotive with them. They warn that the locomotive has a very old decoder. The user enters the address on the Throttle, selects the DCC-14-speed-step protocol using option keys, and presses Enter. The user is in control of the locomotive.

- A Gateway device connects a proprietary layout control bus to OpenLCB. Throttles connected to the proprietary bus select locomotives by a DCC address. The Gateway device intercepts these messages and sends out Train Search Protocol messages to the OpenLCB bus. A DCC Command Station connected to the OpenLCB bus creates Virtual Train Nodes for the intended locomotives. A radio-equipped OpenLCB locomotive that has an up to 6 digit number on the cab can also be selected using this method.

- A large software system designed to control model railroads has a system-specific implementation module for a variety of different command systems. The API of this module gives a 4-digit DCC address to select a locomotive. The OpenLCB implementation of this API translates this number to a Train Search Protocol message in order to trigger an OpenLCB-connected Command Station to create a Virtual Node. The Command Station uses its settings to decide whether it should be a DCC protocol train or some other protocol (e.g. Märklin-Motorola). A radio-equipped OpenLCB locomotive that has an up to 4 digit number on the cab can also be selected using this method.

1.3 Unserved Use Cases

- The Train Search protocol is not a generic search protocol. It is not possible to search for any other entity than a Node[3]. While the concept could be used for searching for other types of nodes, the specific allocation shall only be used for searching for Train Nodes. This is necessary in order to ensure that Throttles that use this allocation properly function on the network.

- The search results are only Node IDs. If additional information beyond the Node ID is needed (such as user-visible name), the Throttle has to use other protocols (such as the Simple Node Identification Protocol) to query that information.

- Only digits are supported in the search query. The maximum length of the query is 6 digits. Longer and alphanumeric search queries are not supported.

- A Command Station can only receive a command using the Train Search Protocol when the user wants a new Virtual Train Node to be created. The Train Search Protocol does not support instructing the Command Station to *delete* a Virtual Train Node.

- This protocol does *not* support having multiple Command Stations that serve trains using the same Protocol. For example, you can't have two DCC Command Stations attached, as both will try to create a Virtual Train Node when a particular DCC address is requested.

[3] For example it is not possible to search for an Event.

1.4 Overview of the Protocol

The Train Search Protocol is fundamentally a broadcast search protocol in a peer-to-peer network. The Throttle broadcasts the search query, which reaches all other Nodes; the matching Nodes reply with a broadcast message which reaches the Throttle, allowing the Throttle to compile a list of results.

The input from the Throttle side is a number with up to 6 digits. This number is encoded using BCD into 3 bytes. An additional flag byte is appended that describes options related to protocol to be used (e.g. DCC with 128-speed-step) and specifies certain behavioral parameters for what Train Nodes should report a match. A fixed 4-byte prefix is prepended, which yields an Event ID.

The Throttle sends out an Identify Producer (broadcast) message to the bus. This broadcast message will reach all segments where there is a Command Station or a native OpenLCB Train Node[4].

Train Nodes (both OpenLCB-connected and Command Station managed), upon receipt of such Identify Producer command, will evaluate the request against their identifying properties (e.g. address, cab number, Node name). If they find that the request matches their identifying properties, the Node responds with a Producer Identified message on the same Event ID. If the request does not match that node, no Producer Identified message is emitted.

If no response arrives for 200 msec, and the request specifies so, then a Command Station will create a new Virtual Train Node matching the parameters in the request. The new Node will then emit a Producer Identified message to the bus.

The Throttle is waiting for Producer Identified messages on their requested Event ID. These messages contain a Source Node ID (or equivalent representation). These are the search results. The Throttle may pick the first search result's Node ID and operate that train using the Train Control Protocol, or if multiple responses arrive, then the Throttle may interrogate those Nodes using the SNIP and PIP protocol to present a list of possible trains to select to the user, if there is a display.

2 Annotations to the Standard

The Subsections herein mirror the structure of the Train Search Protocol Standard document.

2.1 Introduction

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

The Train Search Protocol is built on top of the Event Transport Protocol, which operates using generic OpenLCB messages. This means that the Train Search Protocol can operate on any supported OpenLCB transport layer, including CAN-bus and TCP. This allows both wired and wireless Throttles to operate using the Train Search Protocol.

2.2 Intended Use

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

[4]Command Stations and Train Nodes shall declare themselves by sending a Producer Range Identified message upon startup that covers the entire range of the Train Search Protocol Event IDs. Note that this is an exception to the rule that Producer Range Identified should only be used to identify a range that is more than 50% produced as Events.

2.3 References and Context

There is no reference to the Train Control Protocol, which the Throttle would use to drive actual trains. This is on purpose, because the scope of the Train Search Protocol ends after the Throttle discovers the Node ID. The scope of the Train Control Protocol begins once a Node ID is known.

2.4 Message Formats

This Standard is built on top of the Event Transport Standard. All of the necessary messages to operate this protocol are allocated and defined in that Standard.

2.5 Allocation

2.5.1 Identifier Range allocation and License terms

The Train Search Protocol was developed for and proposed as a Standard by Train Control Systems, Inc. It is a valid use of the Unique Identifiers allocated to any specific party for that party to develop proprietary protocols for operating on the OpenLCB network. It is also valid for this party to offer the developed protocol with appropriate documentation as a proposed Standard.

The licensing terms attached to the Standard allows anyone to create compatible products without any concerns related to license fees. It is not permissible however to create an incompatible product.

2.5.2 Identifier Format

This section only defines what the individual bits and bytes of the given identifiers are allocated to. It is described in the Section ‘Interactions’ how the different parties should be using those fields and identifiers.

Notes about the symbols used in the Track Protocol table:

- 0b10 is a binary number of size 2 bits. The MSB is 1, the LSB is 0.
- The symbol " signals a repeat of the previous row’s value in this column.
- The symbol * signals that the row applies to all possible values in this field.

2.6 Interactions

This section contains a glossary of terms used in the description of the interactions. These terms are used throughout the entire Standard and Technical Note according to these definitions.

Examples:

- A *Train Node* could be a WiFi-connected decoder built into a model locomotive, which establishes a connection to and participates in the OpenLCB Network. Alternatively, it could be a Virtual Train Node represented by a DCC Command Station for the purpose of the OpenLCB bus.
- The *Command Station* might be a DCC Command Station with OpenLCB network connection, creating the track signal. It could also be a Gateway from OpenLCB to a proprietary bus, remote controlling a proprietary Command Station connected to that proprietary bus.

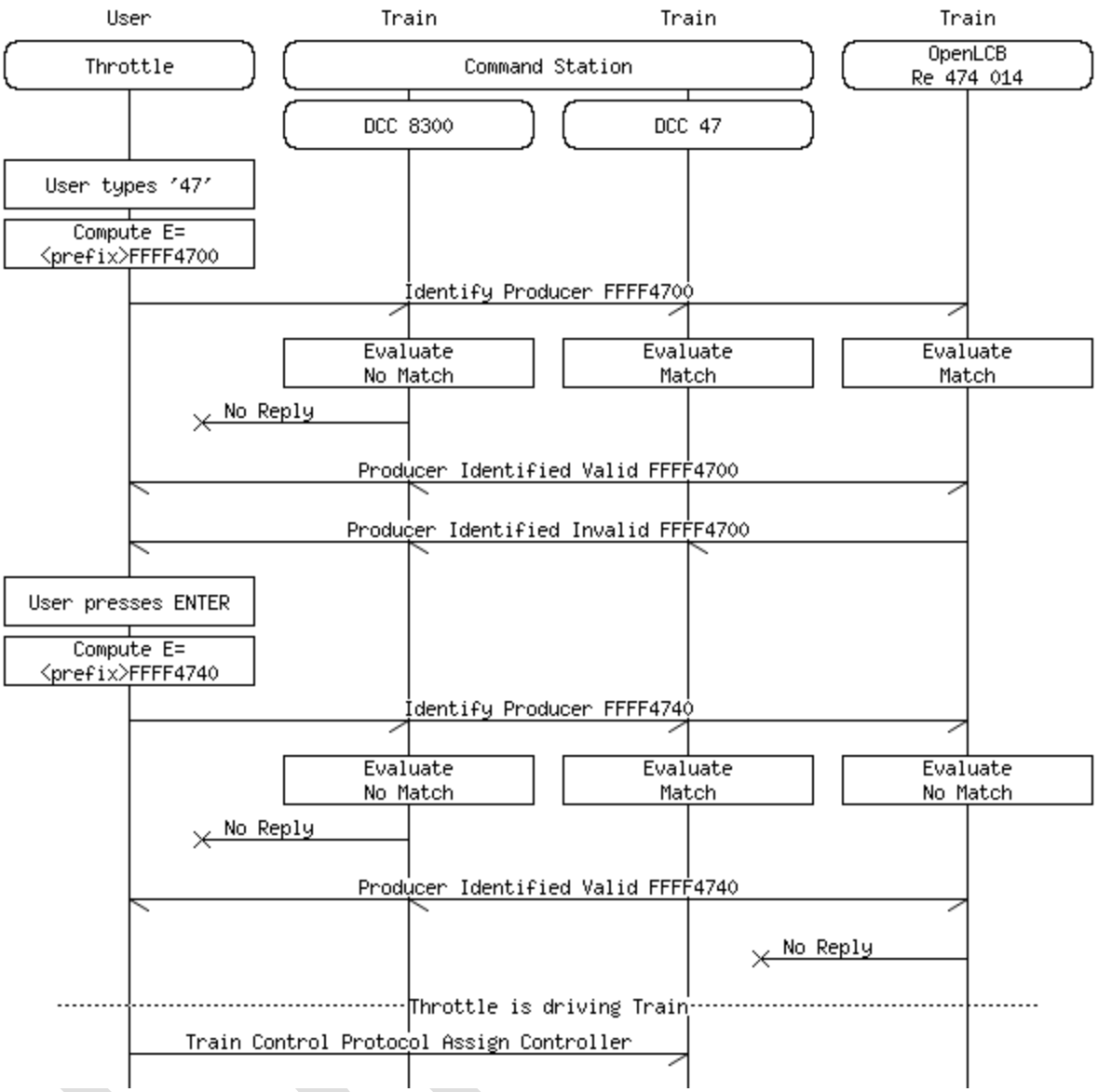
Examples for Train Node properties:

- The *Name* might be
  - “Re 474 014-5” for a Swiss electric locomotive of the series 474;
  - “BR18 313” for a German steam engine of the series 18;
  - “C&NW F7A 415” for an F7 unit of the Chicago&Northwestern Railroad.
  - “Thomas the Tank Engine”.
- *Protocol*: Protocol can be DCC with 7-bit address; DCC with 14-bit address; Märklin-Motorola; MFX/M4; OpenLCB. There is space reserved for adding further protocols in the future.
- *Protocol Version*: DCC has three different speed step modes (14, 28, 128 speed step mode). Märklin-Motorola has multiple versions.
- *Address*: 1 to 127 for DCC 7-bit, 0 to 10239 for DCC 14-bit, 1 to 80 (or 1 to 255) for Märklin-Motorola.

  There is no address for OpenLCB and for MFX/M4, because for these protocols the user is not required to configure a unique numeric value for each decoder. The manufacturer-assigned unique addresses are not relevant for the user and the user is not expected to be aware of or ever input those addresses by the number.

  It might be beneficial for an OpenLCB-connected mobile decoder to allow the user to configure a purely numeric cab number, which the Train Node may apply as if it was a DCC address. This can enable a DCC-only throttle to select that OpenLCB locomotive instead of a DCC locomotive of that address. The user has to be aware that such an option makes that specific DCC address unavailable, as the Command Station will never instantiate a Virtual Train Node for that address so long as the OpenLCB locomotive is on the network.

2.6.1 Search for existing Train Nodes



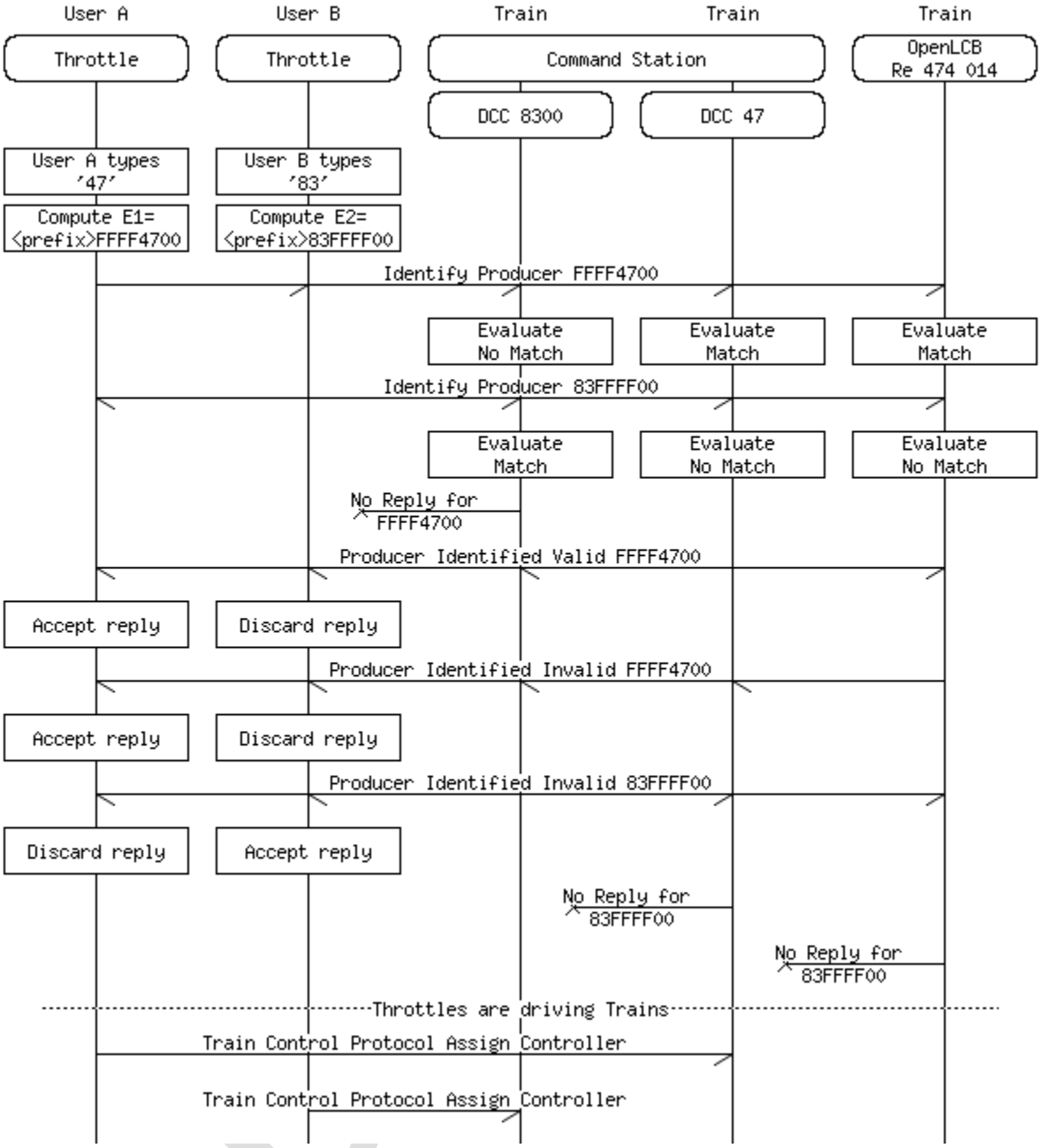
On this diagram there are two interactions with the Train Search Protocol.

In the first interaction the user enters some digits, and the Throttle sends this out as a prefix search for trains starting with “47”. There are multiple replies coming back.

In the second interaction the user presses ENTER, and the Throttle sets the “Exact” bit in the flags byte of the request. Only one reply is generated.

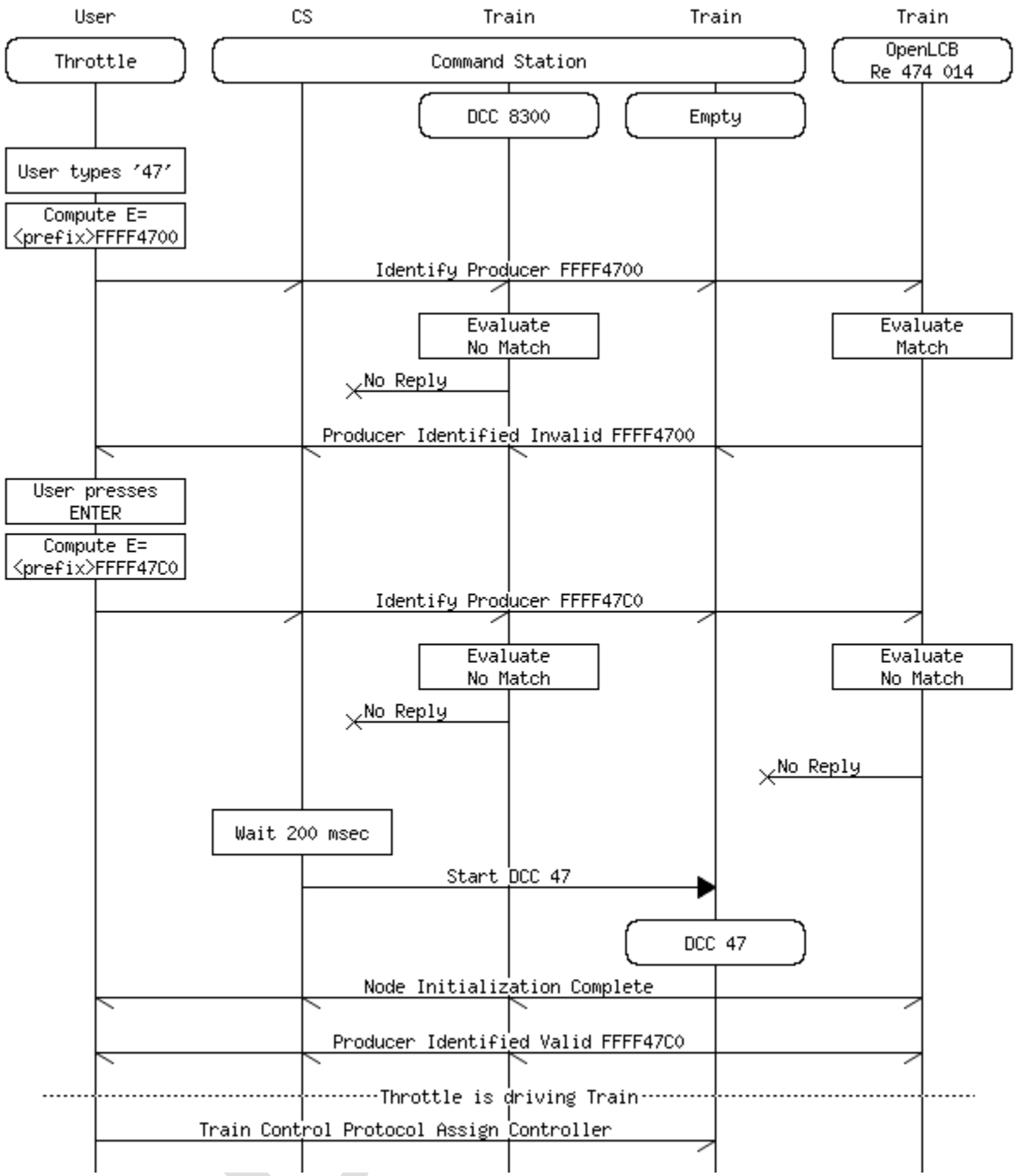
It is of critical importance that the Train Nodes use the exact same Event ID in the replies as what came in the request from the Throttle. This is necessary for the Throttle to identify which of the broadcast responses are requested from the particular throttle. If multiple Throttles are operating the Train Search Protocol simultaneously, the Event ID ensures that there is no race condition between the interactions.

This is depicted in the following diagram:



Depending on the packet queuing behavior of the individual Nodes in this interaction the exact sequence might be slightly different. On CAN-bus, the Producer Identified replies have a higher priority than any potential additional request that might be coming in; as such, the replies for the first request will typically precede the second incoming request on the bus. This ensures that Train Nodes and Command Stations do not need an unbounded amount of buffer to handle Train Search Protocol.

2.6.2 Allocate a new Train Node



215 On this diagram the same two interactions are presented as in Section 2.6.1 , where the DCC 47 node does not exist at the beginning of the interaction. The Throttle is configured to send the request with the "Allocate" bit set when the user pressed ENTER. The Command Station waits for 200 msec for any responses to come in to the exact match request, after which it assigns an empty Train Node to proxy for DCC Address 47. The newly started node then responds to the Throttle.

220 If the low five bits of the flag byte contain a protocol selection information, then the Command Station would create the new Virtual Train Node according to that protocol. Here the CS used the default settings (DCC protocol).

If the Command Station does not support the protocol requested, or the address requested for that protocol[5], then no new virtual Train Node is created and no response is sent to the throttle. The throttle will typically display a "Train not found or Command Station not responding" error in this case.

2.6.3 Event Identifier matching algorithm

In this section a complex logic expression is defined. The parentheses are important in such an expression and they are represented by typographical levels of hierarchies:

- A bullet point list is interpreted to have an open parentheses before the first bullet and a close parentheses after the last bullet.
- Each bulleted line has an open parentheses at the beginning of the line and a close parentheses before the operator at the end of the line.

This logic expression defines the exact interpretation of the various flag bits and query nibbles which were allocated in the Section 'Allocations'. For example, this logic expression normatively defines what the DCC "Default Address Space" means (it means short address for 1..127 and long address for 128..10239).

It is important that this algorithm is part of the Standard, therefore Command Stations and Train Nodes must follow it very precisely. The interoperability of throttle features and use-cases of mixing Command Station and OpenLCB Train Nodes on the same network depends on the correct implementation of this algorithm.

The following non-normative statements help in interpreting the algorithm. The terminology is that the request is *accepted* to generate a response, or *dropped* to not generate a response.

- If the request specifies an unknown (reserved) option, protocol or nibble in the query, the request is dropped.
- The request may or may not specify a Protocol. If there is no Protocol specified (five LSB bits are all zero), then a Train Node with any protocol is eligible to accept. As an example, if the request specifies OpenLCB as protocol, then the Train Nodes maintained by Command Stations for DCC trains will not reply.
- If the Address-Only bit is set in the flags, the match of the numeric part of the request shall only be checked against the Address of the Train Node; otherwise both the Address and the Name properties shall be checked.
- If there is more than one separate sequence of digits in the *qq qq qq* nibbles, then Address will not match. In this case it's hard to interpret what numerical value this query should represent. Typical throttles with numeric keypad will only ever send a single digit sequence.
- An Exact match against the address is if the numerical value matches the Address's numerical value. This is the only acceptable match is the Exact-Only bit is set in the flags. If this bit is clear, then prefix match is also checked, for example '83' will prefix-match '8300'.
- DCC has two overlapping address ranges. Address values over 128 are always interpreted as DCC long addresses. Address values 0-127 are interpreted as either short or long addresses

[5] An example would be addresses 82 to 255 for Märklin-Motorola protocol or short address 100 to 127 when using certain DCC Command Stations.

when doing a default search (i.e., match both short and long addresses). The Throttle may specify the protocol to be DCC and set the bit DCC Force Long Address to indicate that a long address train is requested. A common practice for throttles is to select DCC Long Address when the user prefixes the number with an extra zero, e.g. the user typing “053” would be sent as DCC-force-long-address 53, with the Event ID <prefix>53FFFFEC (Note[6]). When the Command Station receives a request with Force Long Address, only the trains with actual long addresses will match. When the Protocol is Default/Any or Protocol is DCC but the Force Long Address bit is clear, the Allocate bit is consulted to decide what to do. If the Allocate bit is clear, both short and long address Trains are matching. When the Allocate bit is set, only short address Trains are matching. This special case is necessary to be able to allocate a DCC train short-address-53 when there is already a DCC train with long-address-53 on the network.

Some specific examples for this rule:

- request “53”, not force long, not allocate => Train 053L accepts, Train 53S accepts;
- request “53”, force long => Train 053L accepts, Train 53S drops;
- request “53”, not force long, allocate => Train 053L drops, Train 53S accepts.

- To match a number against the Name of the Train, digits in the Name are matched. For each substring of digits in the Name, the throttle’s digits are matched at the beginning of this substring to get a prefix match. Digits from the Name are used up from here on, jumping over any other characters until the nibbles in the number received from the throttle are exhausted. If Exact-Only match is requested, then at this point the next character in the Name has to be not a digit. For example the Name of “C&NW F7A 415” is exact-matched by the query 415, prefix-matched by the query 41 or 4. The query 15 or 5 does not match this name. The query 7 is also exact-matched, because “7” inside “F7A” fulfills the requirement that there are no further digits before and no further digits after. This Name also matches the query nibbles 7F415F, because both terms in the query (“7” and “415”) match the Name. Also the query 7415 exact-matches (‘A’ and the space character was jumped over to match this), but 741 is only a prefix-match.

2.6.4 Search Result Differentiation

Some possible methods for this differentiation are as follows:

- An exact match might be rated at a higher quality than a non-exact match. This differentiation is shown in the message diagrams in this Section.
- A Command Station may collect statistics about usage of individual addresses. A train that is more recently used could be marked with a higher quality than one that was used a long time ago.
- From the same statistics, a train that is more frequently used might be marked with higher quality than a train that is less frequently used.

Throttles are not required to make use of the differentiated answers. A possible use-case might be as follows:

[6]This Event ID also sets Allocate, Exact and Address-only bits to ensure that it connects to a DCC train via a Command Station.

- A throttle with a UI might show a list of Nodes that replied, and this list might be sorted to show better matches ahead of others.

3 Alternatives considered

- A more generic protocol could be developed for searching for Nodes on the network; this may be peer-to-peer or based on a centralized database built and maintained by a single Node that stores details about all Nodes available on the network (including all Train Nodes).

  At the time of this writing, no specific proposals exist for this protocol. The requirements space and ambiguity in this domain is very large, and initial discussions in the OpenLCB Group have not sufficed to identify a straightforward solution to this problem.

- All Nodes can be enumerated by the unaddressed Verify Node ID Global message. There is no throttling mechanism provisioned to maintain the list of replies with a limited amount of memory. The Node list can then be further processed using Protocol Support Inquiry (PIP) messages to restrict to Train Nodes and Simple Node Information Request (SNIP) messages for querying identifying properties such as train name.

  Without a throttling mechanism, using this method to find and identify Train Nodes would require an unbounded amount of memory for the Throttle Nodes. This makes it difficult to build simple and cost efficient Throttles.

- All Train Nodes can be identified using the Well-Known Event ID IsTrain. There is no throttling mechanism provisioned to maintain the list of replies with a limited amount of memory.

  Without a throttling mechanism, using this method to find and identify Train Nodes would require an unbounded amount of memory for the Throttle Nodes. This makes it difficult to build simple and cost efficient Throttles.

- A legacy track protocol Command Station could expose a command interface either via dedicated OpenLCB messages or via a Datagram-based protocol to maintain its list of virtual Train Nodes. This protocol could also be out-of-band, for example through a user interface (physical or HTTP-based), or could be coupled to a persistent database using the Memory Configuration Protocol and the Configuration Descriptor (CDI) XML to expose the user interface via a Configuration Tool.

  There was an actual protocol proposal developed for this alternative under the name of Traction Proxy Protocol. The difficulty with this approach is that a Throttle can be programmed to operate using the Traction Proxy Protocol, which would make the Throttle work well with a Command Station, but naturally exclude the ability of this Throttle to connect to native OpenLCB Train Nodes. This makes the solution not desirable as a long-term offering, as it does not provide a seamless upgrade path to radio-equipped locomotives.

- Instead of encoding the information to transmit in Event IDs, a protocol could be developed using custom OpenLCB messages that are marked as global/broadcast in the MTI. This would allow fewer compromises around the encoding.

Creating a new broadcast protocol would pose difficult questions around buffering and traffic routing. The Event Transport Standard has specific preparation about interconnecting different network segments with filtering and routing for example. All of these features would need to be specifically duplicated to reach similar performance characteristics as using Events. The duplicate features would then have to be specifically implemented by all gateways and routers.

Even with a protocol defining custom messages, there would be a fairly strict limit on the amount of payload bytes that can be transported. This is because there is no generic protocol layer for transporting broadcast messages on CAN-bus with more than 8 bytes of payload.

4 Test Vectors

	Query	|---Train Node Properties---|		
Result	Event ID	Addr	Name	Protocol
no match[7]	090099FFFFFF1300	45	"FooBar"	DCC
no match[8]	090099FFFFFF1300	4513	"FooBar"	DCC
match[9]	090099FFFFFF1300	45	"FooBar13"	DCC
match[10]	090099FFFFFF1300	45	"Foo135"	DCC
no match[11]	090099FFFFFF1340	45	"Foo135"	DCC
match	090099FFFFFF1340	45	"Foo13 5"	DCC
match[12]	090099FFFFF13540	45	"Foo13 5"	DCC
no match[13]	090099FFFFF13540	45	"Foo13 51"	DCC
match	090099FFFFF11340	45	"1 11 3"	DCC
no match	090099FFFFFF1300	45	"1 11 3"	DCC
match[14]	090099FFFFFF1300	13	"FooBar777"	DCC
match[15]	090099FFFFFF1300	135	"FooBar777"	DCC
no match[16]	090099FFFFFF1340	135	"FooBar777"	DCC
match	090099FFFFF13540	135	"FooBar777"	DCC
no match[17]	090099FFFFFF3500	135	"FooBar777"	DCC
match[18]	090099FFFFFF1360	13	"FooBar777"	DCC
no match[19]	090099FFFFFF3520	135	"FooBar35"	DCC
match	090099FFFFFF5300	53	"Foo777"	DCC 28-speed long address
match	090099FFFFFF5300	53	"Foo777"	DCC 28-speed
no match	090099FFFFF0530C	53	"53S"	DCC 28-speed
match	090099FFFFF0530C	53	"Foo777"	DCC 28-speed long address
no match	090099FFFFFF53E0	53	"053L"	DCC 28-speed long address
match	090099FFFFF053EC	53	"Foo777"	DCC 28-speed long address
match	090099FFFFF0130C	13	"FooBar777"	DCC long address

[7]Neither Name or Address matches here.
[8]Address contains 13 but not as a prefix, therefore it does not match.
[9]Name contains a sequence of digits matching 13.
[10]Name contains a sequence of digits which has a prefix of 13.
[11]Exact match fails in the name.
[12]Name match jumps over spaces between the digits here.
[13]Not an exact match, because there is an extra digit after the match.
[14]Address matches here.
[15]Prefix match in the address.
[16]Exact match requested, but only prefix match found.
[17]Not a prefix match.
[18]Address-only request and match in the address.
[19]Match is only in the Name, but was requested in the address.

```
no match[20] 090099FFFFF0130C    13     "FooBar777"  DCC 28-speed
no match     090099FFFFF013EC    13     "FooBar777"  DCC
no match     090099FFFFF0130C    13     "FooBar777"  DCC 28-speed
no match     090099FFFFF013EC    13     "FooBar777"  DCC

no match[21] 090099FFFFF13FE6    13     "FooBar777"  DCC
   match     090099FFFFF13FE6    13     "FooBar777"  Märklin-Motorola II
   match     090099FFFFF0130C    13     "FooBar777"  DCC 14-speed long address

no match[22] 090099FFFFFF13E0    13     "013"        DCC 28-speed long address
no match[23] 090099FFFFF013EC    13     "13"         DCC 28-speed

no match[24] 090099FFFFF13FE6    13     "013L"       DCC 28-speed long address
no match     090099FFFFF13FE6    13     "13S"        DCC 28-speed
   match[25] 090099FFFFF13FE6    13     "13m"        Märklin-Motorola I
   match     090099FFFFF13FE6    13     "13M"        Märklin-Motorola II
   match     090099FFFFFF13E0    13     "13m"        Märklin-Motorola I
   match     090099FFFFFF13E0    13     "13M"        Märklin-Motorola II
no match[26] 090099FFFFF013EC    13     "13m"        Märklin-Motorola I
no match     090099FFFFF013EC    13     "13M"        Märklin-Motorola II
no match     090099FFFFF13FEC    13     "13M"        Märklin-Motorola II
no match     090099FFFFF13FEC    13     "13m"        Märklin-Motorola I
no match[27] 090099FFFFF13FE8    13     "13m"        Märklin-Motorola I
no match     090099FFFFF13FE8    13     "13M"        Märklin-Motorola II
```

5 Recommended Operation

While the protocol contains a lot of different features, a fairly specific set of rules are provided in the standard on the interpretation of these features. These rules were put together to allow a certain set of user experience options to be easy to implement and produce the behavior that is believed to meet the expectations of modelers in how typical digital command systems operate.

In the following section these different classes of user interfaces are enumerated in how they are expected to use the protocol features.

- Utility throttle with numeric keyboard 0-9 and a select key, with no screen.

  On this throttle the only available user interaction for locomotive selection is that the user will punch in a number and press the select key. There will be only one round of interaction of the Train Search Protocol. The Throttle is recommended to perform the request by setting the *qq qq qq* nibbles to the user-entered digits, and the *rr* options to 0xC0 (allocate + exact). If the user

[20]This match fails because the request specifies DCC long address, but the train has DCC short address.
[21]The request specifies a Märklin-Motorola address, but the train is DCC.
[22]The Address does not match here, because all of the bullet points are false in Section 6.3, “qq qq qq nibbles of E matches the Address if and only if”. This is the most complex case. Specifically, calling up the default with allocate explicitly requests a short address, but the locomotive has a long address. If this item were a match, it would become impossible to run both long address 13 and short address 13 from a displayless throttle.
[23]Requests long address, train has short address.
[24]Request Märklin-Motorola II, train has DCC.
[25]Protocol versions (MM I vs MM II, just like DCC speed step modes) do not partake in the match algorithm. The Command Station might take this allocate request as a hint that the user wants to change the protocol version of the given Train Node, but that Train Node has to match the request. A new Train Node can not be created, because it would have the same OpenLCB Node ID as the existing one.
[26]Requests DCC long address, but train has MM.
[27]Requests DCC address, but train has MM.

typed in a prefix zero, then use 0xEC (force DCC long address), if two prefix zeros, then use 0xE6 (force MM protocol).

There are some compromises on this throttle with respect to differentiating between Train Nodes that have the same address but a different protocol. It is strongly recommended against setting the “address only” bit or else it becomes impossible to select a native OpenLCB Train Node. With the above settings the specific concession is that if a native OpenLCB Train Node has a cab number in the 1..127 range, then it becomes impossible to drive a DCC short address locomotive with the same address so long as the native Train Node is turned on.

- Smart throttle with numeric keyboard 0-9, a screen supporting scrolling and select keys.

  For every number entered by the user, the Throttle is recommended to execute a *search* operation, and list the resulting locomotives on the screen. The search operation uses the digits specified so far, with the *rr* options as 0x00. This instructs a prefix search with no protocol restriction. The search operation will return all native OpenLCB locomotives matching, and all virtual Train Nodes that were created so far with this prefix.

  The user has two choices for continuing their journey. Either they see their intended locomotive on the screen, so they scroll and select one given locomotive. This does not require any further searches, the locomotive can be operated using the Train Control Protocol directly.

  Alternatively, the user may press the select/enter key without picking a locomotive from the result list. Then the intention is interpreted as driving a given address via the command station, creating a new virtual Train Node as needed. The throttle sends another Train Search request with the digits and *rr* options 0xE0 (allocate+exact+address only). If the user typed in a prefix zero, then use 0xEC (force DCC long address), if two prefix zeros, then use 0xE6 (force MM protocol). The alternate protocols could be a user-configurable setting.

  The user interface might have additional elements for protocol selection, which would be expressed in the options field, but the recommended default choice is to have the above behavior.

- A gateway connecting a proprietary throttle bus to OpenLCB, where the throttle bus transmits only a four-digit number for locomotive selection.

  The gateway is recommended to use the interaction from the utility throttle with no screen.

- A gateway connecting a proprietary throttle bus to OpenLCB, where the throttle bus transmits a four-digit number and an *is_long* boolean for locomotive selection.

  When is_long == false, use the given number with *rr* options of 0xC0. When is_long == true, use options 0xEC.

These recommendations are laid out with the following principles:

1. The user should be able to select both DCC locomotives and native OpenLCB locomotives. This is why fixing the protocol selection bits to DCC are avoided during the default interaction type.

2. The user should be able to select between locomotives from different address spaces (e.g. DCC short address, DCC long address, address of another protocol), and be able to run two locomotives of the same address at the same time if they belong to different address spaces. Although the user interface might limit the available interactions, any available possibilities are used to select between short and long DCC addresses; usually one prefix zero selects long address. Numeric addresses 128 and above do not need a prefix zero to be unambiguous.
3. If possible, there should be a distinction between searching for existing trains (which might return OpenLCB Train Nodes) vs allocating a new virtual Train Node using the DCC address. If the user interface is too limited, principle 1 takes precedence over this.

Table of Contents