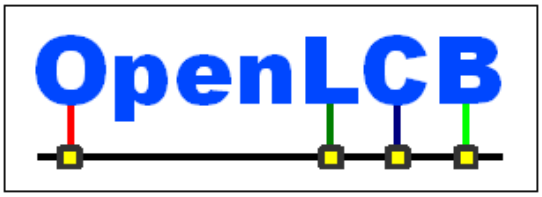

OpenLCB Standard	
Train Search Protocol	
~~July 22, 2024Jul 14, 2024~~	~~Preliminary~~ Adopted

1 Introduction (Informative)

This standard defines a method for Throttle Nodes to find Train Nodes on the network, and for ~~legacy~~track protocol use-cases, instruct an OpenLCB Command Station to create a virtual Train Node given a ~~legacy address and~~ track protocol and an address. This standard is not specific to any ~~wire~~link layer ~~protocol~~choice.

2 Intended Use (Informative)

The OpenLCB ~~Traction~~Train Control p~~P~~rotocol describes how a Node acting as ~~t~~throttle can and should control a Node acting as a train. An important component of an ecosystem is how the throttle can find the remote ~~t~~Train ~~n~~Node that corresponds to the user's desired locomotive to control. This standard defines one possible method on how to find ~~t~~Train ~~n~~Nodes from a throttle. As a basic transport this method uses the Event Transport protocol.

When an OpenLCB Command Station is controlling many locomotives through a track protocol (e.g. DCC), it acts as a gateway between the OpenLCB Train Control protocol and the track protocol. For the purpose of the Train Control protocol, all of the controlled trains have to have unique Train Nodes on the OpenLCB bus, therefore the Command Station is representing many Nodes on the OpenLCB bus. These are called *virtual Nodes*, or *virtual Train Nodes*.

In addition to finding already existing Train Nodes, this standard describes an interaction that can be followed by a Command Station to instantiate the virtual Train Nodes ~~in order to act as a gateway between the OpenLCB traction protocol and a legacy track protocol such as DCC.~~ . As part of this interaction, certain options are specifically enumerated in this standard that are specific to current commonly used ~~legacy~~ track protocols (such as DCC), while leaving expansion space for including additional such protocols in future revisions.

Note that this standard does not exclude other methods and standards, including some not yet developed, for the same or similar purposes. See the Train Search Protocol Technical Note for some alternatives that were considered or can be employed based on more general enumeration protocols.

~~TODO: move this discussion to the TN as "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).~~

- ~~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.~~
- ~~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.~~

~~A legacy track protocol Command Station could expose a command interface either via dedicated OpenLCB messages or via a Datagram-based protocol to maintain it's 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.~~

3 References and Context (Normative)

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

- The Event Transport Standard, which defines the protocol for transporting events, including the messages and interactions for inquiry and discovery of event Producers.
- The Event Identifiers Standard which describes the allocation scheme of Event Identifiers.

For more information on format and presentation, see:

- OpenLCB Common Information Technical Note

4 Message Formats (Normative)

This standard does not define any OpenLCB messages.

5 Allocation (Normative)

5.1 Identifier Range allocation and License terms

For the purpose of the Train Search Protocol the following block of consecutive Event Identifiers is allocated:

09.00.99.FF.00.00.00.00 – 09.00.99.FF.FF.FF.FF.FF

All Event Identifiers in this range are reserved for exclusive use according to the interactions defined by this standard and shall not be used for any other purpose.

The legal entity to whom this Event Identifier range is allocated by the ~~Event~~ Unique Identifiers Standard, Train Control Systems, Inc, hereby grants an irrevocable, non-transferable license to anyone for using the quoted Event Identifiers on the condition, and only so long as, that their use is compliant to this Standard or any later version of it, published by Train Control Systems, Inc., or the OpenLCB Group.

5.2 Identifier Format

Note that the semantic meaning of these identifiers are defined in Section 6 (Interactions).

Table 1. The Event Identifiers in the given range are defined as follows:

Byte 1	Byte 2	Byte 3	Byte 4	Byte 5	Byte 6	Byte 7	Byte 8
09	00	99	FF	qq	qq	qq	rr
Fixed prefix				Search query (6 nibbles)			Flags

70 The search query ‘qq qq qq’ shall be a sequence of 6 nibbles in MSB-first order stored in Bytes 5, 6 and 7 of the Event Identifier.

Table 2. Each individual nibble is one position of the search string:

Search Query Nibble value	Description
0 - 9	The given position of the search term is the given number
0xA - 0xE	Reserved. Do not send, check upon receipt.
0xF	Empty / unused character position. Short queries shall be padded with this nibble value to 6 characters long. Multiple search terms may be concatenated to a search query with this nibble as a separator between them, which means the individual terms shall be evaluated with an AND relation between them.

Table 3. The flag byte ‘rr’ is defined as follows:

Bit 7	Bit 6	Bit 5	Bits 4-0	Description
Allocate				0x80: Force allocate ~~legacy~~new gateway node 0: Search only existing nodes
	Exact			0x40: Exact match only 0: All matches (including partial match)

Bit 7	Bit 6	Bit 5	Bits 4-0	Description
		Address only		0x20: Match only in address 0: Match everywhere (address and name)
			Track Protocol	See ~~separate t~~Table 4 for assignment

75

Table 4. The Track Protocol values are defined as follows:

Bit 4-3	Bit 2	Bits 1-0	Description
0b00 ~~0b00~~ ~~0b00~~ ~~0b00~~ ~~0b00~~ " """"""""" " "	0 ~~0~~ ~~0~~ ~~0~~	0b00	Any / Default track protocol
		0b01	Native OpenLCB Train Node
		0b10	MFX® / M4® track protocol
		0b11	Reserved (do not send, check on receipt).
	1 " " " "	*	M~~ä~~ärklin-Motorola track protocol
		0b00	MM – Any / Default version
		0b01	MM – Protocol version I (14 speed steps + F0)
		0b10	MM – Protocol version II (Directional + F0-F4)
		0b11	MM – Protocol version II with following address for F5-F8 support.
0b01 " " " " " "	*	*	DCC track protocol
	0	* *	DCC – Default address space
	1		DCC – Force 14-bit (long) address
	* * * *	0b00	DCC – Any / Default speed steps
		0b01	DCC – 14 speed steps
		0b10	DCC – 28 speed steps
		0b11	DCC – 128 speed steps
0b10	*	*	Reserved (do not send, check upon receipt)
0b11	*	*	Reserved (do not send, check upon receipt)

6 Interactions (Normative)

The following hardware and software nodes are taking part in the interactions presented here:

- The **Train Node** is an OpenLCB Node that implements the ~~Traction~~Train Control Protocol, controlling a single (physical) train with one or more coupled engines. The hardware implementing the Train Node may be physically built into a model, or it may be built into a centralized gateway hardware that converts to a non-OpenLCB protocol in order to remotely control an engine, for example via Bluetooth or a ~~legacy~~pre-existing track protocol such as DCC or Märklin-Motorola. In this case a single piece of hardware may be responsible for representing multiple OpenLCB Nodes on the network.
- The **Throttle Node** is an OpenLCB Node that intends to send ~~Traction~~Train Control Protocol commands to a desired Train Node. The Throttle Node may be a physical hardware device with a user interface to be used by an operator, a computer software with a user interface for operators to control trains, or a fully automated software.
- The **Command Station** is a gateway for proxying to some non-OpenLCB protocol, implementing Train Node(s), and having an OpenLCB network connection. There may or may not be a separate OpenLCB Node that represents the Command Station itself on the OpenLCB network.

A Train Node has a set of properties, the following of which are relevant for the interactions defined in this Standard:

- **Name** is a user-assigned textual (alphanumeric) description for the train.
- **Protocol** defines the native command set that is used to address this train. The options are listed in Table 4.
- **Address** is a user-assigned numerical value that uniquely identifies a train within a Protocol. There cannot be two trains with the same Protocol and same Address on the track at the same time. Not all Protocols have a user-assigned Address. The Address for the DCC Protocol also carries a disambiguator on whether it is a short (7-bit) or long (14-bit) DCC address.
- **Protocol Version** selects the exact command set to use for a Track Protocol. Users sometimes have to select a specific Protocol Version to correctly operate a train or to access all possible features of it.

6.1 Search for existing train nodes

The goal of this use-case is for a Throttle Node to enumerate Train Nodes that exist on the network and match certain criteria. The Throttle Node shall represent the given criteria as an Event Identifier *E* according to Section 5.2, and shall set the flag byte ‘rr’ Bit 7 (Allocate) to zero (0).

- The Throttle Node shall send an “Identify Producer” message with setting the Event Identifier to *E* to the network.
- A Train Node, upon receipt of an Identify Producer message with an Event Identifier *E* falling into the Event Identifier Range of Section 5.1, shall

- compare the Train Node's identifying properties to the search criteria represented by the Event ~~Identier~~Identifier *E* according to Section 6.3;
- in case of a match, the Train Node shall emit a "Producer Identified" message with the Event Identifier *E* to the network, setting the Producer validity bits according to Section 6.4 ;
- in absence of a match, the Train Node shall not emit a "Producer Identified" message with the Event Identifier *E*.

6.2 Allocate a new Train Node

The goal of this use-case is for a Throttle Node to instruct a Command Station to create a new Train Node in the case that no existing Train Node(s) match the search criteria requested by the Throttle Node. This interaction can also be used to change the Protocol Version of an existing Train Node.

The Throttle Node shall represent the requested address as an Event Identifier *E* according to Section 5.2, with ~~setting~~ the flag byte 'rr' Bit 7 (Allocate) set to one (0x80). The Throttle Node may, but is not required to, specify the desired track protocol in the flag byte. It is recommended that the Throttle Node also sets ~~the~~ Bit 6 (Exact) to one (0x40) in the flag byte.

- The Throttle Node shall send an "Identify Producer" message with setting the Event Identifier to *E* to the network.
- A Command Station upon receipt of an Identify Producer message with an Event Identifier *E* falling into the Event Identifier Range of Section 5.1 with the flag byte 'rr' Bit 7 (Allocate) set to one, shall
  - validate that it has the ability to create a Train Node matching its identifying properties to the search criteria represented by the Event Identifier *E* according to Section 6.3; for properties marked as 'Any / Default' by the Event Identifier *E* the Command Station may pick any implementation-specific default or user-configured value;
  - wait at least 200 msec for existing Train Nodes to reply with a "Producer Identified" message;
  - in the absence of such reply, the Command Station shall
    - allocate a new Train Node according to the properties defined by the Event Identifier *E*, or
    - adjust the Protocol Version of an existing Train Node conflicting with the requested address to match the value defined by the Event Identifier *E,*

    then instruct the Train Node to emit a "Producer Identified Valid" message with the Event Identifier *E*.

6.3 Event Identifier matching algorithm

This section defines when an Event Identifier *E* in the range defined by Section 5.1 matches a Train Node with the identifying properties of <*Name, Address, Protocol, Protocol Version*>.

The Train Node matches the Event Identifier *E* if and only if

- the search query represented by the *qq qq qq* nibbles of *E* matches, ~~and~~AND
- the requirement by the *rr* flag byte matches the *Protocol* ~~and *Protocol Version*~~, ~~and~~AND
- the Flag byte or the query nibbles specify no reserved values marked as “check upon receipt”.

The flag byte *rr* matches *Protocol* if and only if the Track Protocol field of *rr* is set to “Default/Any” ~~or~~OR *Protocol* matches the value of the Track Protocol field.

The *Protocol Version* is not used by the matching algorithm.

~~The flag byte *rr* matches *Protocol Version* iff the Track Protocol field of *rr* is set to the specific value or range matching *Protocol* and the version bits of *rr* are set to “Default/Any” or to a value that matches *Protocol Version*.~~

The Flag byte *rr requests a DCC long address* if and only if *rr* specifies the DCC protocol AND Bit 2 (DCC - Force long address bit) is set.

The search query represented by the *qq qq qq* nibbles of *E* matches if and only if

- *qq qq qq* matches the *Address* of the Train Node, ~~or~~OR
- Bit 5 (~~the~~ Address only) ~~bit~~ is ~~not set~~clear in the Flag byte *rr* ~~and~~AND *qq qq qq* matches the *Name*.

The *qq qq qq* nibbles of *E* matches the *Address* if and only if *qq qq qq* contains exactly one contiguous sequence of digit nibbles ‘0’-’9’, ~~and~~AND

- *Protocol* is no~~t~~n- DCC, ~~or~~OR
- the decimal value represented by these nibbles is >= 128, OR
- the Flag byte *rr* requests a DCC long address AND *Address* is a DCC 14-bit (long) address~~specifies “Default/Any” protocol~~, ~~or~~OR
- the Flag byte *rr* does not request a DCC long address AND *Address* is a DCC 7-bit (short) address, OR
- the Flag byte *rr* does not request a DCC long address AND Bit 7 (Allocate) is clear in the Flag byte *rr*~~Bit 2 (DCC - Force long address bit) is clear in the Flag byte *rr*, or~~

~~*Address* is >= 128, or~~

- ~~*Address* is a DCC 14-bit (long) address~~

~~and~~AND

- the Exact bit of the Flag byte *rr* is set ~~and~~AND the decimal value represented by these nibbles is the value of the *Address*, ~~or~~OR
- the Exact bit if the Flag byte rr is clear ~~and~~AND the decimal nibbles form a prefix of the decimal representation of *Address*.

The *qq qq qq* nibbles of *E* matches the *Name* if and only if *qq qq qq* contains one or more contiguous sequence of digit nibbles 0x0 0x0-0x9 separated with one or more nibble 0xF and each of those match the *Name.*

~~A maximal[1] consecutive*vv...v* has a prefix of *nn...n* sequence[2] of digit nibbles *nn...n* match the *Name* iff there is a maximal consecutive sequence of decimal digits *vv...v* in *Name* where~~

~~if the, or~~

~~*vv...v* equals *nn...n.*~~

A maximal[3] contiguous sequence of digit nibbles *nn...n* in *E* match the *Name* if and only if there is a pair of positions [*p,q*] in *Name* such that there is no digit character immediately preceding position *p* in *Name*, AND *nn...n* equals the concatenation of all digit characters found in *Name* from position *p* to *q*[4], ~~substring of decimal digits *vv...v* in Name not immediately preceded by any further digits where *nn...n* = *vv...v* and~~AND

- Flag byte *rr* has Bit 6 (Exact) not set, OR~~or~~
- there is no digit character immediately following position *q*~~vv...v~~ in *Name* ~~is not immediately followed by any further digits~~.

6.4 Search Result Differentiation

A Train Node replying with Producer Identified message may, but is not required to to differentiate on how accurately the search query matches the Train Node's properties by picking the response message from the set of Producer Identified Valid, Producer Identified Invalid and Producer Identified Unknown messages. ~~to differentiate on how well the search query matches the Train Node's properties, using the~~The responses represent the quality of match with~~ratings of~~ 'Valid' > 'Invalid' > 'Unknown' ~~in quality~~. A Node may use any implementation-specific algorithm, which may also take into account properties not represented in this standard for making the determination.

If a Train Node does not make a differentiation, it shall use Producer Identified Unknown as response message for Search responses.

[1]No '0'-'9' characters immediately before or after this substring.

[2]Contiguous subsequence is commonly referred to as substring.

[3]No 0x0-0x9 ~~characters~~nibbles immediately before or after this substring within *qq qq qq~~this substring~~*.

[4]Any non-digit characters shall be skipped for this definition.

210

Table of Contents