OpenLCB Working Note	
Location Service	
Oct 19, 2023	Preliminary

1 Introduction

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 working note is the start of effort toward a general location service protocol which can be used by e.g. RFID and QR scanners and other detectors to publish information about the detected location of rolling stock.

1.1 Served Use Cases

1) A signal needs to determine occupancy in the block it protects. It is not concerned with identity, but it does need to know location and entry/exit.

2) A train enters a current-detected block. A detector recognizes this, successfully queries the decoder for its address, and reports that information on the OpenLCB bus. That information results in a signal sending a speed command such as Slow or Stop to that particular detector.

3) A train enters a current detected block. A detector recognizes this, and attempts unsuccessfully to query the decoder for its address. The detector emits a report without decoder address information.

4) A train exits a current-detected block. A detector recognizes this, and reports that interaction on the OpenLCB bus.

5) A train crosses an RFID reader which determines the train's DCC address and reports that address on the OpenLCB bus.

6) A train-announcement unit needs to determine the identity and location of approaching trains so it can announce their arrival.

7) A train display needs to gather all train locations for display to the users.

8) A train display needs to gather the location and identity of all trains within a specified area.

9) An application needs to gather the location and identity of all trains.

10) A node powers up and emits a query which generates messages carrying all available location information.

11) An RFID reader reads a tag and uses this protocol to distribute the raw tag contents. A separate database node receives this content, does an internal lookup to locate the rolling stock's Unique ID, and then uses this protocol to distribute a human-readable version of the original distribution that also carries the rolling stock's Unique ID.

12) A detector observes that a piece of rolling stock has been added to the layout. It reports this. A throttle or command station provides that rolling stock in its roster display.

13) A detector observes that a piece of rolling stock has been removed from the layout. A throttle or command station removes that rolling stock from its roster display.

14) The user has a throttle, which has a train on it. There is a location report coming in for that train. The user would like to display on the throttle the name of the location where that train was seen. (This has to be a user-visible name, so coming out of some configuration.) This raises two indirection issues that need to be solvable:

    A) Indirection of the OpenLCB train and the detector address. Say this is a DCC train from a command station, then there is a known DCC address, which is recognized by RailCom detectors, and reported. The harder case of OpenLCB trains needs to be handled too.

    B) Indirection from the detector address to a user-visible (configured) block name.

15) Cab signaling: display on the throttle what the next signal aspect is for the selected train. This raises indirection issues:

    A) Going from train to block as above

    B) Going from physical orientation plus reverser setting on the logical locomotive to an absolute direction inside the block

    C) Going from block + absolute direction to a next signal (which is a multi-state state machine)

    D) Aspect interpretation (state to human readable text)

1.2 Unserved Use Cases

(To be determined as the proposal is developed)

1.3 Requirements

1) The protocol should cover multiple hardware types, including RFID and QR scanners, RailCom and other DCC detectors, and follow-on hardware that can detect and identify rolling stock at a point, on a track section, or in an area.

2) To the extent possible, the protocol should identify standard forms for common information.

    A) The Unique ID of the identified rolling stock should be in a common location in the protocol's message(s) so that it can be extracted regardless of the form of the sensor that generated it: a RFID detector and a QR detector on the same layout should both provide the same information on the identity of the detected rolling stock.

    B) The Unique ID of the scanner should be in a common location in the protocol's message(s) so that it can be extracted regardless of the form of the sensor that generated it.

3) To the extent possible, the protocol should identify methods for its message(s) to carry detector-specific information. For example, a RFID detector and a QR detector on the same layout should both be able to publish their tag-specific information.

4) The identification of the scanning hardware should be guaranteed unique. This lets hardware be added to a layout without having to resolve conflicts.

5) The messages should not rely on the transmitting-node-address field of the message for decoding. I.e. the protocol doesn't say "You identify who made the scan by looking at who sent the message". This allows messages to be created by e.g. a debugging node during development.

6) It should be possible to look at the message(s) carrying a specific scan and know that they contain a scan without additional information. I.e. you don't have to configure a layout monitor with all possible scanner IDs before it can start processing or displaying scanning results.

7) The identification of which scanner is reporting should be early in the message(s). This allows nodes to drop the receipt of the message(s) early, without having to buffer them completely. (Note that the relative priority of this requirement and the one below has not yet been determined)

8) The identification of the detected rolling stock should be early in the message(s). This allows nodes to drop the receipt of the message(s) early, without having to buffer them completely. (Note that the relative priority of this requirement and the one above has not yet been determined)

9) There should be a standard way to find the UniqueID of the scanned piece of rolling stock, when the scanner can provide one. For some scanners, e.g some types of RFID scanner, the scanner might not be able to map the scanned value to a specific piece of rolling stock, in which case this information field would be blank.

10) Note that the item directly above is referring to a Unique ID (node ID), not a DCC address. Someday, there will be LCC-native trains and this protocol should be ready for them. The alternative is inevitably a messy migration at some point in the future.

11) Ideally, the messages for this protocol could be a super-set of those from the DCC Detection Protocol. (This does not seem possible now, but at an early stage of requirement generation one can dream...)

12) Ideally, information about short circuit and current overload status could be conveyed by this protocol. These are often generated by the same device as the transponder messages e.g. LocoNet devices. These are providing location-specific information. It would be good to have some commonality between all location based messages. Another example would be current or voltage measurements at a point.

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.

A “scanner’ is a device that can gather information from the railroad, typically when something happens. RFID readers, QR code readers, point optical detectors and track current detectors are all examples of scanners.

A scanner forwards one instance of the gathered information as a “report”.

2.2 Intended Use

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

This protocol is intended to provide a standardized way for scanners to distribute report information across the OpenLCB network.

2.3 Reference and Context

See also the DCC Detection working note for a simpler form specific to DCC.

This working note assumes adoption of the Event With Payload working note as a Standard.

2.4 Message Formats

The proposal is to carry these within a Producer Consumer Event Report as an event with payload defined by (note bytes 0-7 are the Event ID of the message, 8-264 are the payload of the message)

- Bytes 0-1: A defined prefix from the well-defined space - to be assigned. This will enable nodes not interested in this protocol to discard the event with payload transmission immediately.
- Bytes 2-7: A Unique ID that specifies the device that read the information e.g. a scanner. This will be in the 1st message (1st CAN frame), allowing early rejection of uninteresting scanner sources.
- Bytes 8-9: Flag bits, set below
- Bytes 10-15: A Unique ID that specifies the item being scanned, e.g. a train node ID, or zeros if that is not available. This will fit in the 2nd message, allowing early rejection of uninteresting items when available.
- Bytes 16-NN: Any read (scanned) content that's available.

    Note that the proposed 256 byte payload size allows 248 bytes here.

Flag bits

0-1 – direction of relative motion information – defined with respect to the front of the locomotive or piece of rolling stock

0b00 stopped

0b01 forward

0b10 reverse

0b11 unknown

2-3 – direction of absolute motion information – the meanings of these are defined during set up of the detector

0b00 stopped

0b01 east/north

0b10 west/south

0b11 unknown

4-7 reserved: Send as zero and ignore on receipt

8-10 reserved: Send as zero and ignore on receipt

11-15 – information format – see below

The information format field is used to indicate the format of bytes 16-NN

0x00 reserved, do not send; it is an error to receive this

0x01 occupancy/location information only, no additional payload

0x02 human-readable format – not intended for machine parsing

0x03 JSON formatted data (where the specific JSON content is defined how? "Self-defining" is not great in a standard)

0x04 - 0x07 raw content of a read RFID tag, QR code, etc. (Do we need to have subtypes here for the type of tag, i.e. ISO14443 vs ISO15693.

(Much more work needed here)

2.5 States

This protocol has no states

2.6 Interactions

When a scanner gathers new information, it may emit a PCER message with the above format. PCER messages are global, so all interested nodes will receive this message. The interested nodes can then process them as desired.

2.6.1 Identification of scanners

Every node that produces information via this protocol also consumes a specific well-known Event ID consisting of the well-known prefix assigned above followed by six 0x00 bytes. Sending this Event ID in an Identify Producer message will result in Producer Identified messages that indicate which nodes can take part in this protocol.

This removes the need to include it in the Protocol Identification Protocol bits, although that can also be considered for ease of access.

Once the nodes that use this protocol have been identified, an Identify Events directed to each of them will return Producer identified messages indicating the Unique IDs of all the scanners.

The six bytes of 0x00 was chosen because that is not a valid unique ID for a reader.

2.6.2 Identification of scanned objects

There is currently no way to efficiently request the unique IDs of all scannable objects. You can retrieve all presently-visible objects by retrieving the most recent scan reports, see section below.

2.6.3 Retrieving most recent scan reports

When a new node joins the network, it may want to catch up on previous scan reports. In that case, it can produce (in a PCER message) the well-known ID of the well-know prefix followed by size 0x00 bytes. This is consumed by all nodes that implement this protocol. When consumed, the node will re-emit the most recent report from each of its scanner implementati

How to handle multiple overlapping reports from the same scanner? Just send them all? In what order? Option for the implementor?

2.6.4 Traffic limiting

Producing nodes may not produce the same event with payload content twice in a row

This is a compromise. It reduces the amount of state that a producing node needs to retain down to one message regardless of the number of attached readers, while at the same time trying to reduce the bandwidth consumed by repeating detection messages.

Another approach would be to say that a node may not produce the same content twice in a row from each scanner that it supports. This will prevent "scanner A, scanner B, scanner A, scanner B" bursts of messages. The cost is some additional buffering.

Another approach would be to say that a node may not produce the same content more than N times a second. This may require buffering of O(10kB/N) bytes, although there may be hardware or software structures that require less than this.

Note that the Unique ID that specified the device doing the reading (bytes 2-7) does not need to be the Node ID of the scanning node. It just needs to be uniquely assigned by the node creator from some assigned space to indicate the device doing the read.

Note that a single node implementing multiple readers will send event with payload messages from those separate readers with different Unique reader IDs, but must not interleave the messages making up the payload of those separate PCER messages. The event with payload protocol requires that the sending node only sends one sequence at a time. This may require internal buffering for the messages from each reader until they can be sent.

2.6.5 Motivate the use of high-bytes in the Event ID to identify this protocol

Use of high bytes in Event ID to identify this protocol…

3 Background Information

RFID readers and tags come in multiple forms. With 13.56MHz RFID, be it ISO14443 (as used in the RC522 readers), or ISO15693 as used in StaRFIshrail, the following points apply:

- Up to 8 bytes can be read "on the fly", i.e. as the tag on a train passes over a reader at speed. Depending on how the reader is set up, these 8 bytes can either be the UID of the tag, programmed in at time of manufacture, or 8 bytes of EPROM, which can be programmed into the tag by the user. The usage of these 8 bytes would be determined by the user, but could, say, be 2 bytes for Stock ID, 2 bytes for DCC address, and 4 bytes for other purposes.
- In addition, there at least another 96 bytes of user programmable EPROM, but these need to be read when the tag is essentially static over a reader, due to the time taken to read all the bytes.

The tags used by the MERG CAN RFID readers provide five bytes of programmable payload. Multiple conventions have been proposed for the contents of those bytes.

QR codes come in multiple forms. Depending on version, size and error correction level, they can contain thousands of characters.

Some detectors can report the direction of the detected train. For some, this is whether the train is moving in forward or reverse. For others, it's whether the train is moving in one direction on the track vs the other (North vs South, East vs West). ProTrak Grapevine systems can report moving North/East, not moving, and moving South/West.

RPS detectors reported absolute X, Y and Z coordinates of the detected rolling stock. The report was emitted at specified intervals. Although these are no longer being produced, something similar may emerge in the future with developments in e.g. video monitoring of a layout.

Traditional block detectors and point detectors are an interesting special case. They're simple, low-cost and common on layouts. They can't sense any address information from the detected rolling stock. The protocol should be able to convey their information too.

Table of Contents