OpenLCB Technical Note	
Function Description Information	
July 22, 2024	Adopted

1 Introduction

"Function Description Information" (FDI) in this context refers to *fixed* information available from an OpenLCB Train Node, via OpenLCB, so that other devices can properly and correctly operate it.

Other information may be available via e.g. manuals or the Internet, but the format of that information is not under specification here.

A key use for FDI is to enable an OpenLCB throttle to determiine the features provided by a Train Node's functions. The throttle will use the FDI information to render some form of suitable Graphical User Interface to allow the user to easily and intuitively operate the functions provided by the Train Node

An important design choice was to embed the FDI into each node so that the system has all it needs to operate the Train Node without having to source the information externally to the OpenLCB network from the manufacturer or some other on-line repository via the Internet or a CD/DVD etc. An OpenLCB throttle is not required to have an Internet connection.

1.1 Requirements

- Train Nodes must carry enough context that a stand-alone throttle can provide a useful human interface without getting any data from an external source, e.g. needing an Internet download or the user entering or uploading a database.

1.2 Preferences

- Train Nodes shouldn't need a lot of processing power, e.g. to compress or decompress data in real time. Memory usage should also be limited, but is a second priority.
- The necessary OpenLCB operations should be state-less and idempotent to simplify software at both ends.

2 Annotations to the Standard

2.1 Introduction

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

The concept and interaction are using the same pattern and similar format as the Configuration Description Information, purposefully, with the expectation that OpenLCB devices will be able to share the software implementation between these two use-cases.

2.2 Intended Use

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

The most important use-case of operating a model train is controlling speed and functions. The protocol for these are defined in the OpenLCB Train Control protocol Standard. The user typically uses a Throttle for performing these operations, either as a special-purpose hardware device, or as a software function on a generic computing device.

While the semantic meaning of speed control is very well defined, the meaning of functions is left very vague by the OpenLCB Train Control standard, just like other standards of the industry; a function is some form of remote controlled feature of the train (or locomotive). This vague definition is intentional, as manufacturers have come up with new and innovative ideas on what kind of additional features they want to add to their models, and the typical way to operate these features is using the function abstraction.

Typical functions are lighting functions (like headlights, tail lights, class marker lights, etc.), motor control functions (momentum, braking, etc.), sound functions (horn, bell, whistle, engine sounds, compressor, etc.) and a variety of other functions, such as a smoke unit, raising and lowering a pantograph or operating a locomotive-mounted remote uncoupler. There is no definitive list of all possible functions. The TN-218 document by RailCommunity lists over 120 different types.

Not only is the list of possibilities is very large, one specific model alone can carry dozens of different functions. Users of proprietary throttles have to activate these typically by their function number, for example using a numeric keypad. This is easy to operate, provided that the user has memorized which function number does what on that specific locomotive. With dozens of functions per locomotive and up to hundreds of locomotives on a big layout, this is no easy task. As such, it is an important use-case for the OpenLCB Throttle to display a user interface that provides information to the user about what function numbers are available, and which numbers perform what operations on the train.

The Function Description Information (FDI) provides exactly this metadata for the throttle: the list of functions, for each function the number at which it is accessible, and a textual description of what the given function does.

In addition to the description for the user, the throttle user interface needs some additional metadata to determine how the particular function button should behave. There are two common behaviors for on/off functions: *momentary* functions are turned on while the user is holding the button, and turned off upon release. *Toggle* (or binary) functions are turned on with a press of the button, then they stay on after release, until the user presses the button again. A typical momentary function is horn, a typical toggle function is headlight. This is purely a behavior of the user interface, as the underlying protocol for turning the function on and off is the same for both cases. *Analog* functions, such as a horn volume or pitch, take a numerical value to control their operation.

2.3 Reference and Context

The Memory Configuration Protocol is one way to retrieve the FDI, but FDI is independent of that. Download of this data via a direct BLE transfer offering a similar capability would make this specification equally applicable.

2.4 Content

The function description information for a Train Node is invariant while the node has any OpenLCB connections in the Initialized state. The Train Node's transition back to Initialized state via sending an Initialization Complete global message tells other nodes to flush their caches and pick up any changed content.

There are no default values that e.g. associate "Bell" with a particular location or function. These are thought to be too brittle, and there are just too many possibilities to be useful.

2.5 Format

The structure is somewhat similar to the Configuration Definition Information structure, with named sections that organize and describe individual items.

The use of XML 1.0 format is intended to make the content as easy as possible to parse.

The use of XML Schema 1.0 is intended to allow the use of common XML parsers. Note that the schema definition file includes an XML Schema 1.1 check for extensibility that has been commented out so that the operational schema remains valid Schema 1.0.

2.5.1 XML Elements

Note that nothing in the Standard prevents providing an xml-stylesheet instruction, and it's common to provide one of the form:

```
<?xml-stylesheet type='text/xsl' href='xslt/fdi.xsl'?>
```

2.5.1.1 <fdi> Element

The <fdi> element forms the root of the document. It provides the "xmlns:xsi" and "xsi:noNamespaceSchemaLocation" attributes that define the schema for automated processing.

2.5.1.2 Element

The element is provided for future extensibility. The "space" and "origin" attributes are carried over from the Configuration Description Information format, but at the present state of the standard, there is no way to express a different memory space than 0xF9. Similarly, the layout of offsets in the memory space is not defined today. If a future version of this standard provides such capabilities, then the "space" and "origin" attributes may be used in a backwards-compatible way, with 0xF9 and 0 being their respective defaults. They should be omitted in current implementations.

2.5.1.3 <group> Element

Groups may contain one or more inner groups, with the same representation. This may continue to any desired level.

Note that unlike in the CDI standard, groups can not be specified with a repetition count in the FDI. The reason for this is that individual functions are addressed by the <number> sub-element, and repeating a group is not meaningful, because that sub-element would remain the same.

2.5.1.4 <function> Element

The “kind” attribute is an enumerated type of (currently) "binary", “momentary”, and "analog".

The “number” element indicates which function is being described by this element. This value is to be used in the Train Control Protocol Set Function operation in the "address" field. That field is 24 bits, therefore the maximum value in this element is 16777215.

Note that there is no requirement on the ordering of the <function> elements with respect to their <number>. It is not required for functions to be listed in ascending order by number, and it is not required for every function number in a range to be present (i.e., numbers can be skipped). A function number not listed in the FDI is assumed by throttles to be doing nothing (i.e., not presented on a UI having a list of functions).

The “name” element provides information which can be used if the throttle has e.g. a graphical interface to the user.

The “size” attribute is optionally allowed for compatibility with some early, pre-Standard implementations. It should be omitted in current and future implementations.

3 Additional Information

3.1 Design points

Basic configuration is done with the configuration protocol defined elsewhere.

Function values are stored in the 0xF9 memory space. They can be addressed through the Train Control protocol’s Set Function and Query Function commands. The Memory Configuration Protocol can also be used to read and write the memory space 0xF9, if that is implemented. This allows reading a large number of functions with a single request. However, while the Train Control Protocol Set Function has 16 bits available to represent a function's value, in the memory space 0xF9 there is only 1 byte per function address; the default value of deprecated attribute 'size' being 1 represents this. There is no resolution for this discrepancy in the current Standard.

3.2 Sample Function Definition XML:

```
<?xml version='1.0' encoding='UTF-8'?>
<?xml-stylesheet type='text/xsl' href='xslt/fdi.xsl'?>
<fdi xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
xsi:noNamespaceSchemaLocation='https://openlcb.org/schema/fdi/1/0/fdi.xsd'>
```

```
<segment>
  <group>
    <name>Lighting Functions</name>
    <description>Headlights and Marker lights and Running lights, oh my.</description>
    <function kind="binary">
      <name>Headlights</name>
      <number>0</number>
    </function>
    <function>
      <name>Ditch Lights</name>
      <number>4</number>
    </function>
  </group>
  <group>
    <name>Sound Functions</name>
    <description>Toot toot!.</description>
    <function kind="analog">
      <name>Horn</name>
      <number>8</number>
      <min>0</min>
      <max>2</max>
    </function>
  </group>
</segment>
</fdi>
```

3.3 Sample Older Implementation

A sample older implementation of the DCC F0-F28 functions, with some optional attributes still in place:

```
<?xml version='1.0' encoding='UTF-8'?>
<?xml-stylesheet type='text/xsl' href='xslt/fdi.xsl'?>
<fdi xmlns:xsi='http://www.w3.org/2001/XMLSchema-instance'
xsi:noNamespaceSchemaLocation='https://openlcb.org/schema/fdi/1/0/fdi.xsd'>
<segment origin='0' space='249'>
  <group>
    <name>F0-F28</name>
    <description>Standard DCC functions</description>
    <function size="1" kind="binary">
      <name>Headlight</name>
      <number>0</number>
    </function>
    <function size="1">
      <number>1</number>
    </function>
    ...
    <function size="1">
      <number>28</number>
    </function>
  </group>
</segment>
</fdi>
```

4 Possible Future Work

This section discusses some areas where future work may be desired. It is meant as a record of the discussions to date, not as defining any specific solution(s).

4.1 Data Compression

Some work has gone into XML content compression for the Configuration Description Information (CDI) data. See the CDI Technical Note for various considerations that have been discussed. A solution there, if developed, standardized and deployed, could also be used here.

The function definition information is expected to be significantly smaller than a Train Node's CDI, so there may be less need to compress the FDI. On the other hand, CDI gets retrieved relatively rarely (only when a user intends to modify configuration), whereas FDI gets retrieved regularly as part of normal operations of a model railroad.

4.2 Internationalization of the Content

Not everybody runs their trains in English.

The xml:lang attribute could be used to define versions of the <name> and <description> elements that use an alternate language.[1] This approach can greatly increase the size of the XML content if the internationalization effort is very successful. On the other hand, it's transparent to XML-using nodes that don't implement internationalization.

4.3 Machine-readable function classification

Currently the function's type is expressed in a free-text form in the <name> element. This has a wide expressibility, but presents challenges for user interface development:

- Text generally needs to be rendered with a font size that is readable for the given user.
- Long text takes a significant amount of screen real-estate.
- Text display assumes the user can read, which excludes children under a certain age.

Instead of this, a variety of competing model railroad systems define function types which are then represented by appropriate icons. The icons can be standardized in size, and convey significantly more information on a given real estate than text. This limits the user's choice (there is a fixed number of different icons provided by the implementation usually), but a larger number of functions will fit on the user interface than in a text version.

In order to add machine-readable function type, we need two things:

- An enumeration that lists possible function types and assigns numbers to them. There is such a list in TN-218 by RailCommunity.
- An attribute or element for transmitting the function enumeration in the FDI.

Table of Contents

1 Introduction....1
1.1 Requirements....1
1.2 Preferences....1
2 Annotations to the Standard....1
2.1 Introduction....1
2.2 Intended Use....2
2.3 Reference and Context....3
2.4 Content....3
2.5 Format....3

[1] JMRI DecoderPro does this in its decoder definitions. That experience guides some of the rest of this paragraph.

2.5.1 XML Elements....3
2.5.1.1 <fdi> Element....3
2.5.1.2 Element....3
2.5.1.3 <group> Element....4
2.5.1.4 <function> Element....4
3 Additional Information....4
3.1 Design points....4
3.2 Sample Function Definition XML:....4
3.3 Sample Older Implementation....6
4 Possible Future Work....6
4.1 Data Compression....6
4.2 Internationalization of the Content....6
4.3 Machine-readable function classification....7