OpenLCB Standard	
Function Description Information	
Jul 1, 2024	Draft

1 Introduction (Informative)

This document defines a standard for the storage of static information that describes the user interface options for controlling available functions on an OpenLCB Train Node, called “Function Description Information (FDI)”.

The format is XML-based in order to allow reasonable extensibility, while enabling programmatic generation and parsing of the structured information. This XML document is specific to the particular Train, and is exposed at the Memory Space 0xFA. Throttle Nodes retrieve this data from the Train Node using the Memory Configuration Protocol.

This concept and interaction is 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 Intended Use (Informative)

The Throttle Node downloads the FDI content from the Train Node using the Memory Configuraiton Protocol after the user selected the given train. The XML is exposed at a fixed Memory Space number (0xFA). The throttle parses the XML and uses the information therein to configure the function buttons’ behavior and/or the display items shown to the user.

The information presented is the list of available function numbers, the type of the function button (toggle, momentary or analog function) and for analog functions the available range of values that are valid.

If the throttle has fixed marked buttons (e.g. a button labeled Horn/Whistle), it will have to assign a varying function number to these buttons, by finding which function number belongs to the given behavior.

OpenLCB connected native Train Nodes built into specific models may ship a fixed constant FDI XML, as it will describe the behavior of the functions as implemented by the hardware and software. Flexible / configurable OpenLCB decoders may have to provide a mechanism to update the FDI XML if their function numbers can be reconfigured. DCC Command Stations that provide Virtual Train Nodes for OpenLCB throttles may either provide a generic fixed FDI (e.g. listing Headlight, Bell, Horn, F3..F28), or may have a Roster configuration storage, where the user can store a per-DCC-address mapping of functions to behaviors. In this case the Command Station generates the FDI XML on-the-fly based on the DCC address that is being controlled from the throttle. For an MFX/M4 decoder, or a DCC decoder supporting DCC-A, RailComPlus, or the appropriate standardized RailCom data pages, the function mapping can be read out from the locomotive directly using the track protocol, and this information can be used to generate the FDI XML on-the-fly.

The FDI provides a read-only interaction of the function metadata. There is no provision in this Standard on how to modify the function metadata. For modification, the regular configuration standards and protocols are recommended.

TODO: move all of the text below to the TN. This is background information really.

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 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 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 Throttle to display a user interface that provides information to the user about what function numbers are available, and which numbers perform what on the train.

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

In addition to the description for the user, the throttle user interface needs some additional metadata to determine how theparticular 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.

3 References and Context (Normative)

For more information on format and presentation, see:

- OpenLCB Common Information Technical Note

For information on how to fetch the FDI information from a node, see:

- OpenLCB Memory Configuration Protocol Standard

For information on how to control the functions, see

- OpenLCB Train Control Protocol Standard

For information on XML encoding and XML Schema, see:

- World Wide Web Consortium (W3C) “Extensible Markup Language (XML)”[1]
- World Wide Web Consortium (W3C) “XML Schema”[2]

4 Content (normative)

The function description information for a node is invariant while the node has any OpenLCB connections in the Initialized state.

5 Format (normative)

The CDI is provided as a zero-terminated string of bytes. The bytes encode UTF-8 characters. There is no byte-order mark (BOM) at the start of the string. Lines in the string are delimited with 0x0A Newline (NL) characters.

The content defines the configuration description information in XML 1.0 format using a specific XML vocabulary defined by an XML Schema. No extensions to XML 1.0 are permitted.

This version of this Standard specifies version 1.0 of the schema. That version of the schema is defined at https://openlcb.org/schema/fdi/1/0/fdi.xsd and in Appendix A of this document. The FDI content shall pass validation against its referenced schema. Nodes are not required to do the validation.

The version number of an OpenLCB FDI schema contains two numbers: The major version first, and the minor version second.

The first line of the FDI is:

```
<?xml version="1.0"?>
```

to define the XML version of the content.

The root element of the FDI XML is:

```
<fdi xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
     xsi:noNamespaceSchemaLocation="https://openlcb.org/schema/fdi/1/0/fdi.xsd">
```

[1] http://www.w3.org/XML/
[2] http://www.w3.org/XML/Schema

to define the OpenLCB FDI version of the content.

The schema contents are normative.

Numerical values in attributes and element text shall be specified as decimal numbers. OpenLCB nodes are not required to parse any other numeric format.

5.1 XML Elements

5.1.1 <segment> Element

A <segment> element defines the value of a memory space in the attribute `space'. Alternative memory spaces beyond the Function Space (0xF9) are not currently defined for use with Functions.

A <segment> element shall contain an optional user-readable name and optional description tags, and a sequence of zero or more group or function elements. The user-readable name and description are intended as hints for optional UI display by throttles.

5.1.2 <group> Element

The <group> element allows logical grouping of functions, providing common documentation for them.

A <group> element shall contain an optional user-readable name and optional description tags, and a sequence of zero or more group or function elements. The user-readable name and description are intended as hints for optional UI display by throttles.

5.1.3 <function> Element

The <function> element describes the metadata of one function.

The ‘size’ attribute is currently reserved, and if specified, shall carry a value of 1.

The ‘kind’ attribute specifies the behavior of the function button, with the following allowed values:

- ‘binary’ (default) an on-off function with a toggle button behavior
- ‘momentary’ defines an on-off function with a momentary button behavior
- ‘analog’ defines an analog function with integer values accepted between the <min> and <max> sub-element (inclusive).

The <function> element shall contain a <number> element with a decimal integer as contents, which specifies the function number defined.

========= below here is old stuff that needs to be reviewed

5.2 FDI is constant

The FDI information shall be constant. A node may not change it after any part of it has been retrieved and before the next transition of the node away from the Initialized state. (The transition back to Initialized state tells other nodes to flush their caches and pick up any changed content)

6 Stuff to be merged into the above (or the TN)

The first byte is used to distinguish the coding.

“<” (which is part of the XML “<?xml version='1.0'?>” definition): Uncoded characters

UTF BOM: Various UniCode forms

0x01 – tag for compressed. One format defined for total start string of 0x0101. (If you want to use another format, decompress on board)

(can't use 0x00 as lead, since that's the end-of-string indication)

Table of Contents