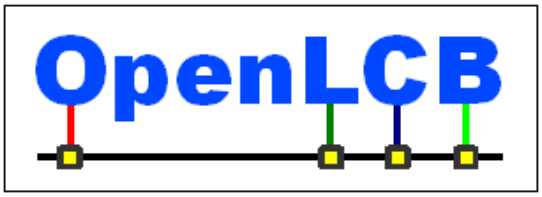

OpenLCB Standard	
Function Description Information	
~~Jul 18, 2024~~July 22, 2024	~~Preliminary~~ Adopted

1 Introduction (Informative)

~~Minimal introductory material, only the stuff that's absolutely needed to understand the Standard~~This document defines a standard for the storage of static information that describe~~d~~s the user interface options for controlling available function ~~configuration options available~~ s on an OpenLCB Train Node~~node~~, called "~~Configuration~~Function Description Information (~~C~~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. found in the configuration memory at address 0xFA.~~.

2 Intended Use (Informative)

~~Any limitations to the area of use of the Standard~~~~Intended to be used to by configurable, self-contained OpenLCB nodes~~..The Throttle Node downloads the FDI content from the Train Node using the Memory Configurat~~ii~~ton 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), optional name and description strings, 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 ~~toof~~ retrieve 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)

[4] ~~Citations to other docs, as needed.~~

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[5] Content (normative)

The FDI information shall be constant. A node may not change the FDI information after any part of it has been retrieved and before the next transition of the node away from the Initialized state.~~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~~FDI 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 1.0 Schema. No extensions to XML 1.0 are permitted.

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

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

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 required to be:

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

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 <fdi> Element

A single <fdi> element is the root of the XML information. The <fdi> element must contain exactly one element.

5.1.2 Element

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

A element may provide a “space” attribute with the fixed value of “249”. Alternative memory spaces beyond the Function Space (249, 0xF9) are not currently defined for use with Functions. A element may provide an “origin” attribute with the fixed value of “0”. The “space” and “origin” attributes are reserved, and new implementations should omit them.

5.1.3 <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~~and/or <function> elements. The user-readable name and description are intended as hints for optional UI display by throttles.

5.1.4 <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) defines 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 unsigned integer values accepted between the optional <min> and <max> sub-elements (inclusive) If not provided, the default for <min> is 0 and the default for <max> is 255.

The <function> element may contain a <name> element. The string contents of that element is meant to be used by a Throttle to convey information about this function to the user.

The <function> element shall contain a <number> element with a decimal integer as contents, which specifies the function number defined. The number must be greater than or equal to zero and less than or equal to 16777215.

Appendix A: The 1.0 Version of the XML Schema for FDI

```
<?xml version="1.0" encoding="utf-8"?>
<?xml-stylesheet href="schema2xhtml.xsl" type="text/xsl"?>
<!-- XML Schema for OpenLCB Function Description Information (FDI)     -->
<xs:schema version="FDI 1.0" xmlns:xs="http://www.w3.org/2001/XMLSchema"
xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

  <xs:simpleType name="NonNegativeInteger">
    <xs:restriction base="xs:int">
      <xs:totalDigits value="5"/>
      <xs:fractionDigits value="2"/>
      <xs:minInclusive value="0"/>
    </xs:restriction>
  </xs:simpleType>

  <xs:complexType name="groupType">
    <xs:sequence>
      <xs:element name="name" minOccurs="0" maxOccurs="1" />
      <xs:element name="description" minOccurs="0" maxOccurs="1" />
      <xs:choice minOccurs="0" maxOccurs="unbounded">
        <xs:annotation>
          <xs:documentation>
            Allows any sequence of the contained element types
          </xs:documentation>
        </xs:annotation>
        <xs:element name="group" type="groupType" minOccurs="0" maxOccurs="1" />
        <xs:element name="function" type="functionType" minOccurs="0" maxOccurs="1"
/>
      </xs:choice>
    </xs:sequence>
  </xs:complexType>

  <xs:complexType name="functionType">
    <xs:sequence>
      <xs:element name="name" minOccurs="0" maxOccurs="1" />
      <xs:element name="number" type="NonNegativeInteger" minOccurs="1"
maxOccurs="1" />
      <xs:element name="min" type="xs:int" minOccurs="0" maxOccurs="1" >
        <xs:annotation>
          <xs:documentation>
            Smallest valid value for this function.
            Only used when type is "analog".
          </xs:documentation>
        </xs:annotation>
      </xs:element>
      <xs:element name="max" type="xs:int" minOccurs="0" maxOccurs="1" >
        <xs:annotation>
          <xs:documentation>
            Largest valid value for this function.
            Only used when type is "analog".
          </xs:documentation>
        </xs:annotation>
      </xs:element>
```

```
    </xs:sequence>
    <xs:attribute name="kind" default="binary">
      <xs:annotation>
        <xs:documentation>
          Type of function being described
        </xs:documentation>
      </xs:annotation>
      <xs:simpleType>
        <xs:restriction base="xs:token">
          <xs:enumeration value="binary"/>
          <xs:enumeration value="momentary"/>
          <xs:enumeration value="analog"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
    <xs:attribute name="size" default="1">
      <xs:annotation>
        <xs:documentation>
          Storage size of this variable in bytes.
          Reserved, ignore upon receipt.
        </xs:documentation>
      </xs:annotation>
      <xs:simpleType>
        <xs:restriction base="xs:token">
          <xs:enumeration value="1"/>
        </xs:restriction>
      </xs:simpleType>
    </xs:attribute>
  </xs:complexType>

  <xs:element name="fdi">
    <xs:annotation>
      <xs:documentation>
        This is the schema for Function
        Description Information (fdi)
      </xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element name="segment" minOccurs="10" maxOccurs="1">
          <xs:annotation>
            <xs:documentation>
              Define the contents of the function memory space
            </xs:documentation>
          </xs:annotation>
          <xs:complexType>
            <xs:sequence>
              <xs:element name="name" minOccurs="0" maxOccurs="1" />
              <xs:element name="description" minOccurs="0" maxOccurs="1" />
              <xs:choice minOccurs="0" maxOccurs="unbounded">
                <xs:annotation>
                  <xs:documentation>
                    Allows any sequence of the contained element types
                  </xs:documentation>
                </xs:annotation>
                <xs:element name="group" type="groupType" minOccurs="0"
maxOccurs="1">
```

```
                    <xs:annotation>
                      <xs:documentation>
                        Allows grouping and replication of multiple locations.
                      </xs:documentation>
                    </xs:annotation>
                  </xs:element>
                  <xs:element name="function" type="functionType" minOccurs="0"
maxOccurs="1">
                    <xs:annotation>
                      <xs:documentation>
                        Describes one function.
                      </xs:documentation>
                    </xs:annotation>
                  </xs:element>

<!--
                  XML Schema 1.1 construct expressing extensibility promise
                  <xs:any minOccurs="0" maxOccurs="1" processContents="lax">
                  <xs:assert test="every $x in * satisfies
                                   (exists($x/@size) and $x/@size castable to
xs:integer)"/>
                  <xs:assert test="every $x in * satisfies
                                   (exists($x/@offset) and $x/@offset castable to
xs:integer)"/>
                    <xs:annotation>
                      <xs:documentation>
                        Extension point for future schema
                      </xs:documentation>
                    </xs:annotation>
                  </xs:any>
 -->

                </xs:choice>
              </xs:sequence>
              <xs:attribute name="space" default="249">
                <xs:annotation>
                  <xs:documentation>
                    The decimal number of the address space where the information is
found.
                    Reserved, ignore upon receipt.
                  </xs:documentation>
                </xs:annotation>
                <xs:simpleType>
                  <xs:restriction base="xs:token">
                    <xs:enumeration value="249"/>
                  </xs:restriction>
                </xs:simpleType>
              </xs:attribute>
              <xs:attribute name="origin" default="0">
                <xs:annotation>
                  <xs:documentation>
                    Starting address of the segment's contents
                    within the memory space.
                    Reserved, ignore upon receipt.
                  </xs:documentation>
                </xs:annotation>
                <xs:simpleType>
```

```
                <xs:restriction base="xs:token">
                  <xs:enumeration value="0"/>
                </xs:restriction>
              </xs:simpleType>
            </xs:attribute>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

Use of XML

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

[5.2] CFDI is constant

Other stuff

CFa

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) (Xinclude, schema default values, …) shall not be assumed to be present.

Because the using node may not have external connections, XML features requiring those

Nodes that use the 'mask' attribute in their CDI must properly implement the write-under-mask operation in their memory configuration protocol support.with legacy equipment only. New OpenLCB implementations should not use it. Instead, they should lay out their configuration variables are processed as if they each had their own addressable location. (How they are actually stored in memory is not specified.)

OpenLCB nodes. Providing optional coding, such as hexadecimal numbers, makes those nodes more complex.

~~The 'mask' attribute is for use . Although it may have made the CDI easier for people to read, the XML is primarily intended for consumption by Numbers for segments, offsets, etc must be specified as decimal numbers. Hexadecimal notation (0x1234) is not permitted~~

~~The schema at prototypes/xml/schema (better location?) is really the normative thing, because that's what we check.~~

~~But we add the <acdi> element to it, so the document isn't really complete; "Other protocols may add, but not remove, elements and attributes"? Remove constraints? How does extensibility work here?~~

Table of Contents

1 Introduction (Informative)....1
2 Intended Use (Informative)....1
3 References and Context (Normative)....2
4 Content (normative)....2
5 Format (normative)....2
5.1 XML Elements....3
5.1.1 <fdi> Element....3
5.1.2 Element....3
5.1.3 <group> Element....3
5.1.4 <function> Element....3
Appendix A: The 1.0 Version of the XML Schema for FDI....5