	OpenLCB Standard
	Configuration Description Information
	March 30, 2023 Draft

1 Introduction (Informative)

This document defines a standard for the format of static information that describes the configuration options available on an OpenLCB node, called "Configuration Description Information (CDI)". "Configuration Description Information" in this context refers to *fixed* information available from an OpenLCB device, via OpenLCB, so that other devices can properly and correctly configure it.

5

This Standard does not address how the CDI is stored, retrieved, or used.

2 Intended Use (Informative)

CDI is intended to be used by a configurable, self-contained OpenLCB node to tell a
10 Configuration Tool (CT) how to configure the node. The configuration tool will use the CDI information to help the user configure all aspects of the node's capabilities.

The configurable values are expressed as v̶Variables, with each v̶Variable having a specific type, a size in bytes, a value for its memory s̶Space and a̶Address (to locate the v̶variable), and n̶Name and d̶Description as user-readable strings so that users understand the use of the particular
15 setting.

Variables can be grouped together, groups can be repeated (for example if a n̶Node has multiple outputs) and nested to express complex configuration setups with concise description.

3 References and Context (Informative)

For more information on format and presentation, see:

20 • OpenLCB Common Information Technical Note

For information on OpenLCB message transport and OpenLCB communications, see:

• OpenLCB Message Network Standard

For information on how to fetch the CDI information from a node, and how to read and write the configuration information, see:

25 • OpenLCB Memory Configuration 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)

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

The CDI has three parts:

- Identification: Provides specific information about the type of the node.

- ACDI: Indicates that certain configuration information in the node has a standardized simplified
35   format.

- Segments: The configuration information in the node is organized in zero or more segments, each of which contains zero or more configurable variables. A variable is the basic unit of configuration. The segment definition specifies the organization of each segment. A segment consists of zero or more bytes within a linear address space.

5 Format (Normative)

40

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
45 vocabulary defined by an XML Schema. No extensions to XML 1.0 are permitted.

This version of this Standard specifies version 1.31.1 of the schema. That version of the schema is defined at http://openlcb.org/schema/cdi/1/31/cdi.xsd and in Appendix A of this document. The CDI content shall pass validation against its referenced schema. Nodes are not required to do the validation.

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

The first line of the CDI is:

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

to define the XML version of the content.

The root element of the CDI XML is:

55
```
<cdi xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
       xsi:noNamespaceSchemaLocation="http://openlcb.org/schema/cdi/1/31/cdi.xsd">
```

to define the OpenLCB CDI version of the content.

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

The schema contents are normative.

60   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 <identification> Element

The <identification> element, if present, specifies manufacturer-provided identification information about the node. This information is not user-editable. If this element is provided and the node also
65   supports the OpenLCB Simple Node Information Protocol (SNIP), the contents of the SNIP Reply shall match the respective tags in the <identification> element. If this element is provided, and the node also provides the <acdi> element, the contents provided by the ACDI spaces shall match the respective tags in the <identification> element.

5.1.2 <acdi> Element

70   The <acdi> element, if specified without the attribute `fixed', or with the attribute `fixed="4"' or higher, specifies that the following information is available for read:

Space	Address	Size (bytes)	Type	Description
252	0	1	int	Version
252	1	41	string	Manufacturer
252	42	41	string	Model
252	83	21	string	Hardware version
252	104	21	string	Software version

The value at the vVersion variable shall be the same as the value of the attribute `fixed'.

The <acdi> element, if specified without the attribute `var', or with the attribute `var="2"' or higher, specifies that the following information is available for read and write:

Space	Address	Size (bytes)	Type	Description
251	0	1	int	Version
251	1	63	string	User-supplied name
251	64	64	string	User-supplied description

75   The value at the vVersion variable shall be the same as the value of the attribute `var'.

The <acdi> element shall be specified if and only if the Protocol Support Reply message carries the `ACDI' bit set. See the OpenLCB Message Network Standard for the Protocol Support Reply message.

If the <acdi> element is specified, and the nNode also supports the OpenLCB Simple Node Information Protocol (SNIP), then the information provided by the SNIP Reply shall match the
80   respective values provided in the ACDI space.

A node may, but is not required to, express the same configuration options as specific segments and dData eElements therein.

5.1.3 Element

85 A element defines the value of the memory sSpace in the attribute `space', which shall apply to all dData eElements within, and the value of `origin', which shall be considered as the aAddress of a dData eElement of size 0 (zero) at the beginning of the [3].

A cConfiguration tTool may, but is not required to, perform visual separation of the contents of different segments by appropriate UI elements, such as tabs, boxes or horizontal bars.

90 A element shall contain an optional user-readable name and optional description tags, and a sequence of zero or more dData eElements. The user-readable name and description are intended as hints for optional UI display by configuration tools.

5.1.4 Data Elements

The following elements are considered dData eElements: <group>, <int>, <string>, <eventid>, <float>.

95 The value of the address within the segment is accumulated during a depth-first traversal of the contents of the segment definition element. The aAddress is initialized with the value of the attribute `origin' on the element. Each time an offset attribute is encountered, the value of the address is incremented by the offset (which may be negative) before any other processing of the element is done. If the element defines a variable, the variable is located at the current address, and the address is then incremented by the size of that variable before advancing to the next element. This is

100 formalized as follows.

For each dData eElement the following values are defined:

- Space, which is defined by the enclosing element.

- Address, which is defined as the eEnd aAddress of the previous dData eElement plus the value of the attribute `offset' on the dData eElement.

105 - Size (in bytes)

- End aAddress, which is defined as aAddress + sSize, unless otherwise specified.

The data element's <name> and <description> elements and the <group> element's <repname> element are intended as hints for optional UI display by configuration tools.

5.1.4.1 <group> Element

110 The <group> element allows logical grouping of vVariables, providing common documentation for them, and making multiple copies of the contained vVariables. CDI implementors may, but are not required to, use this feature to express configuration of repeated hardware or software components (such as multiple input ports, output ports etc).

A <group> element shall contain an optional user-readable name, optional description and a sequence

115 of zero or more dData eElements. This sequence is considered to contain a dData eElement of size 0 (zero) before the specified dData eElements[3].

If the `replication' attribute is present with the value of N, then the group shall be considered as if the entire sequence of dData eElements were repeated N times.

---

[3]This is required to make "previous element" an unambiguous reference for the first element in the contained sequence.

120 The eEnd aAddress of a <group> element is defined as the eEnd aAddress of the last dData eElement in the contained sequence (after replication). The sSize of a <group> element is defined as the eEnd aAddress minus the aAddress of the <group> element.

Configuration Tools shall not render a <group> element with no child elements[4] on their UI.

5.1.4.2 <int> Element

The <int> element defines a vVariable of integer value.

125 The sSize of the <int> element is defined as the value of the `size' attribute in bytes. Valid values are 1, 2, 4 and 8 bytes.

The integer value shall be written to the bytes pointed to in big-endian byte order. All bytes shall be written. Values smaller than defined by the <min> or larger than defined by the <max> sub-element, if present, are invalid and shall not be written. If the <map> enumeration is present, then values not 130 present in the list of <property> entries of the enumeration are invalid and shall not be written.

5.1.4.3 <string> Element

The <string> element defines a variable holding a UTF-8 string that is user-readable.

The sSize of the <string> element is defined as the value of the `size' attribute in bytes.

The string value shall be written to the bytes pointed to, starting at the aAddress of the <string> 135 element, with at least one trailing 0 (null) byte. When writing a shorter string, any unused bytes shall be set to 0 (null).

If the <map> enumeration is present, then values not present in the list of <property> entries of the enumeration are invalid and shall not be written.

5.1.4.4 <eventid> Element

140 The <eventid> element defines a variable holding an 8-byte value representing an eEvent ID.

The sSize of the <eventid> element is defined as 8 bytes.

The eEvent ID shall be written to the bytes pointed to in big-endian byte order (most significant byte first).

If the <map> enumeration is present, then values not present in the list of <property> entries of the 145 enumeration are invalid and shall not be written.

5.1.4.5 <float> Element

The <float> element defines a variable of floating point value.

---

[4]No name, no description and no Data Elements contained.

150   The size of the <float> element is defined as the value of the `size' attribute in bytes. Valid values are 2, 4, and 8 bytes. The format of the bits within the element shall follow the IEEE format of the corresponding size.

The floating point value shall be written to the bytes pointed to in big-endian byte order. All bytes shall be written. Values smaller than defined by the <min> or larger than defined by the <max> sub-element, if present, are invalid and shall not be written. If the <map> enumeration is present, then values not present in the list of <property> entries of the enumeration are invalid and shall not be written.

155   The optional "floatFormat" attribute defines a preferred, but not mandatory, printf-style format for displaying the data to the user.

6 Future Extension (Normative)

Configuration tTools implementing a future version of this Standard must be able to process CDI content defined according to any earlier version of the Standard, including this version.

160   Configuration tTools implementing major version 1 of this Standard may assume the following about future minor versions of this Standard:

- No existing tags will change the interpretation or default value of the `offset' and `size' attribute, and accordingly the aAddress and sSize value, the data type and encoding of the value in the memory space. The <group> tag will not change the interpretation of the `offset' attribute and 165 `replication' attribute.

- All unknown tags that occur within the element or <group> and have an attribute `size' shall be considered to be dData eElements with aAddress defined as the eEnd eAddress of the previous dData eElement plus the value of the `offset' attribute, and sSize defined as the value of the `size' attribute in bytes. The `size' attribute of all future dData eElements shall be 170 required.

No assumptions may be made about major version 2 and up of this Standard.

A Appendix: Schema

```
    <?xml version="1.0" encoding="utf-8"?>
    <?xml-stylesheet href="schema2xhtml.xsl" type="text/xsl"?>
175 <!-- XML Schema for OpenLCB Configuration Description Information (CDI)      -->
    <xs:schema version="CDI 1.31" xmlns:xs="http://www.w3.org/2001/XMLSchema"
    xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">

      <xs:complexType name="mapType">
180     <xs:annotation>
          <xs:documentation>
            A map relates one or more property elements (keys)
            to specific values.
          </xs:documentation>
185     </xs:annotation>
        <xs:sequence>
          <xs:element name="name" minOccurs="0" maxOccurs="1" />
          <xs:element name="description" minOccurs="0" maxOccurs="1" />
          <xs:element name="relation" minOccurs="0" maxOccurs="unbounded">
190       <xs:complexType>
            <xs:sequence>
              <xs:element name="property" minOccurs="1" maxOccurs="1" />
              <xs:element name="value" minOccurs="1" maxOccurs="1" />
            </xs:sequence>
195       </xs:complexType>
          </xs:element>
        </xs:sequence>
      </xs:complexType>


200   <xs:complexType name="groupType">
        <xs:sequence>
          <xs:element name="name" minOccurs="0" maxOccurs="1" />
          <xs:element name="description" minOccurs="0" maxOccurs="1" />
          <xs:element name="repname" minOccurs="0" maxOccurs="unbounded1" />
205       <xs:choice minOccurs="0" maxOccurs="unbounded">
            <xs:annotation>
              <xs:documentation>
                Allows any sequence of the contained element types
              </xs:documentation>
210         </xs:annotation>
            <xs:element name="group" type="groupType" minOccurs="0" maxOccurs="1" />
            <xs:element name="string" type="stringType" minOccurs="0" maxOccurs="1" />
            <xs:element name="int" type="intType" minOccurs="0" maxOccurs="1" />
            <xs:element name="eventid" type="eventidType" minOccurs="0" maxOccurs="1" />
215         <xs:element name="float" type="floatType" minOccurs="0" maxOccurs="1" />
          </xs:choice>
        </xs:sequence>
        <xs:attribute name="offset" type="xs:int" default="0">
          <xs:annotation>
220         <xs:documentation>
              Positive or negative offset between the address of
              the end of previous element and the start of
              this group's contents.
              Offset of zero means that this element starts
225           immediately after the previous one.
            </xs:documentation>
          </xs:annotation>
        </xs:attribute>
        <xs:attribute name="replication" type="xs:int" default="1" />
230   </xs:complexType>

      <xs:complexType name="eventidType">
        <xs:sequence>
          <xs:element name="name" minOccurs="0" maxOccurs="1" />
235       <xs:element name="description" minOccurs="0" maxOccurs="1" />
          <xs:element name="map" type="mapType" minOccurs="0" maxOccurs="1" />
        </xs:sequence>
        <xs:attribute name="offset" type="xs:int" default="0">
          <xs:annotation>
```

```
240              <xs:documentation>
                    Positive or negative offset between the address of
                    the end of previous element and the start of
                    this elements's contents.
                    Offset of zero means that this element starts
245                 immediately after the previous one.
                 </xs:documentation>
              </xs:annotation>
           </xs:attribute>
        </xs:complexType>
250
        <xs:complexType name="intType">
           <xs:sequence>
              <xs:element name="name" minOccurs="0" maxOccurs="1" />
              <xs:element name="description" minOccurs="0" maxOccurs="1" />
255           <xs:element name="min" minOccurs="0" maxOccurs="1" />
              <xs:element name="max" minOccurs="0" maxOccurs="1" />
              <xs:element name="default" minOccurs="0" maxOccurs="1" />
              <xs:element name="map" type="mapType" minOccurs="0" maxOccurs="1">
                 <xs:annotation>
260                 <xs:documentation>
                       The 'value' of each entry is displayed, and
                       the 'property' content (number) is sent
                       to/from the node
                    </xs:documentation>
265              </xs:annotation>
              </xs:element>
           </xs:sequence>
           <xs:attribute name="size" type="xs:int" default="1">
              <xs:annotation>
270              <xs:documentation>
                    Storage size of this variable in bytes.
                 </xs:documentation>
              </xs:annotation>
              <xs:simpleType>
275              <xs:restriction base="xs:token">
                    <xs:enumeration value="1"/>
                    <xs:enumeration value="2"/>
                    <xs:enumeration value="4"/>
                    <xs:enumeration value="8"/>
280              </xs:restriction>
              </xs:simpleType>
           </xs:attribute>
           <xs:attribute name="offset" type="xs:int" default="0">
              <xs:annotation>
285              <xs:documentation>
                    Positive or negative offset between the
                    address of the end of previous element and the
                    start of this elements's contents.
                    Offset of zero means that this element starts
290                 immediately after the previous one.
                 </xs:documentation>
              </xs:annotation>
           </xs:attribute>
        </xs:complexType>
295
        <xs:simpleType name="floatFormat">
           <xs:restriction base="xs:string">
              <!-- This is a somewhat limiting regex, as it does not allow all possible -->
              <!-- printf formats. It will allow the most common formats that have      -->
300           <!-- been seen and used before, however -->
              <xs:pattern value="%[0-9]*(\.([0-9]*))?f"/>
           </xs:restriction>
        </xs:simpleType>

305     <xs:complexType name="floatType">
           <xs:sequence>
              <xs:element name="name" minOccurs="0" maxOccurs="1" />
              <xs:element name="description" minOccurs="0" maxOccurs="1" />
              <xs:element name="min" minOccurs="0" maxOccurs="1" />
310           <xs:element name="max" minOccurs="0" maxOccurs="1" />
```

```
              <xs:element name="default" minOccurs="0" maxOccurs="1" />
              <xs:element name="map" type="mapType" minOccurs="0" maxOccurs="1">
                <xs:annotation>
                  <xs:documentation>
315               The 'value' of each entry is displayed, and
                  the 'property' content (number) is sent
                  to/from the node
                  </xs:documentation>
                </xs:annotation>
320           </xs:element>
          </xs:sequence>
          <xs:attribute name="size" use="required">
            <xs:annotation>
              <xs:documentation>
325           Storage size of this variable in bytes.
              </xs:documentation>
            </xs:annotation>
          <xs:simpleType>
            <xs:restriction base="xs:token">
330           <xs:enumeration value="2"/>
              <xs:enumeration value="4"/>
              <xs:enumeration value="8"/>
            </xs:restriction>
          </xs:simpleType>
335     </xs:attribute>
        <xs:attribute name="offset" type="xs:int" default="0">
          <xs:annotation>
            <xs:documentation>
              Positive or negative offset between the
340           address of the end of previous element and the
              start of this elements's contents.
              Offset of zero means that this element starts
              immediately after the previous one.
              </xs:documentation>
345       </xs:annotation>
        </xs:attribute>
            <xs:attribute name="formatting" type="floatFormat" >
                <xs:annotation>
              <xs:documentation>
350           printf-style format string for displaying data to the user, like %3.1f
              </xs:documentation>
          </xs:annotation>
              </xs:attribute>
        </xs:complexType>
355   
    __<xs:complexType name="stringType">
        <xs:sequence>
          <xs:element name="name" minOccurs="0" maxOccurs="1" />
          <xs:element name="description" minOccurs="0" maxOccurs="1" />
360       <xs:element name="map" type="mapType" minOccurs="0" maxOccurs="1" />
        </xs:sequence>
        <xs:attribute name="size" type="xs:int" use="required">
          <xs:annotation>
            <xs:documentation>
365           Storage size of this variable in bytes.
              This includes the trailing null byte that
              terminates the string content.
            </xs:documentation>
          </xs:annotation>
370     </xs:attribute>
        <xs:attribute name="offset" type="xs:int" default="0">
          <xs:annotation>
            <xs:documentation>
              Positive or negative offset between the
375           address of the end of previous element and the
              start of this elements's contents.
              Offset of zero means that this element starts
              immediately after the previous one.
            </xs:documentation>
380     </xs:annotation>
        </xs:attribute>
```

```
         </xs:complexType>

     <xs:element name="cdi">
385        <xs:annotation>
             <xs:documentation>
               This is the schema for Configuration
               Description Information (cdi)
             </xs:documentation>
390        </xs:annotation>
           <xs:complexType>
             <xs:sequence>
               <xs:element name="identification" minOccurs="0" maxOccurs="1">
                 <xs:annotation>
395                <xs:documentation>
                     Common first element to identify the decoder
                   </xs:documentation>
                 </xs:annotation>
                 <xs:complexType>
400                <xs:sequence>
                     <xs:element name="manufacturer" minOccurs="0" maxOccurs="1" />
                     <xs:element name="model" minOccurs="0" maxOccurs="1" />
                     <xs:element name="hardwareVersion" minOccurs="0" maxOccurs="1" />
                     <xs:element name="softwareVersion" minOccurs="0" maxOccurs="1" />
405                  <xs:element name="map" type="mapType" minOccurs="0" maxOccurs="1">
                       <xs:annotation>
                         <xs:documentation>
                           This map can be used to add arbitrary key/value
                           descriptions of the node.
410                      </xs:documentation>
                       </xs:annotation>
                     </xs:element>
                   </xs:sequence>
                 </xs:complexType>
415              </xs:element>
               <xs:element name="acdi" minOccurs="0" maxOccurs="1">
                 <xs:annotation>
                   <xs:documentation>
                     Element that identifies that memory information is available
420                  as defined by the Abbreviated Common Description Information
                     (ACDI) standard.
                   </xs:documentation>
                 </xs:annotation>
                 <xs:complexType>
425                <xs:attribute name="fixed" type="xs:int" default="4">
                     <xs:annotation>
                       <xs:documentation>
                         The decimal version number of the format for the fixed
                         information block.
430                      </xs:documentation>
                     </xs:annotation>
                   </xs:attribute>
                   <xs:attribute name="var" type="xs:int" default="2">
                     <xs:annotation>
435                    <xs:documentation>
                         The decimal version number of the format for
                         the variable information block.
                       </xs:documentation>
                     </xs:annotation>
440                  </xs:attribute>
                 </xs:complexType>
               </xs:element>
               <xs:element name="segment" minOccurs="0" maxOccurs="unbounded">
                 <xs:annotation>
445                <xs:documentation>
                     Define the contents of a memory space
                   </xs:documentation>
                 </xs:annotation>
                 <xs:complexType>
450                <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>
455                     <xs:documentation>
                          Allows any sequence of the contained element types
                        </xs:documentation>
                      </xs:annotation>
                      <xs:element name="group" type="groupType" minOccurs="0" maxOccurs="1">
460                     <xs:annotation>
                          <xs:documentation>
                            Allows grouping and replication of multiple locations.
                          </xs:documentation>
                        </xs:annotation>
465                   </xs:element>
                      <xs:element name="string" type="stringType" minOccurs="0" maxOccurs="1">
                        <xs:annotation>
                          <xs:documentation>
                            Describes a human-readable UTF-8 string in the data.
470                       </xs:documentation>
                        </xs:annotation>
                      </xs:element>
                      <xs:element name="int" type="intType" minOccurs="0" maxOccurs="1">
                        <xs:annotation>
475                       <xs:documentation>
                            Describes an integer value in the data.
                            The field can be considered either a number,
                            or a set of specific coded values via a map.
                          </xs:documentation>
480                     </xs:annotation>
                      </xs:element>
                      <xs:element name="eventid" type="eventidType" minOccurs="0" maxOccurs="1">
                        <xs:annotation>
                          <xs:documentation>
485                         Describes an 8-byte Event ID in the data.
                          </xs:documentation>
                        </xs:annotation>
                      </xs:element>

490   <!--
                      XML Schema 1.1 construct expressing extensibility promise
                      <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
495   xs:integer)"/>
                      <xs:any minOccurs="0" maxOccurs="1" processContents="lax">
                        <xs:annotation>
                          <xs:documentation>
                            Extension point for future schema
500                       </xs:documentation>
                        </xs:annotation>
                      </xs:any>
       -->

505                 </xs:choice>
                  </xs:sequence>
                  <xs:attribute name="space" type="xs:int" use="required">
                    <xs:annotation>
                      <xs:documentation>
510                     The decimal number of the address space where the information is found.
                      </xs:documentation>
                    </xs:annotation>
                  </xs:attribute>
                  <xs:attribute name="origin" type="xs:int" default="0">
515                 <xs:annotation>
                      <xs:documentation>
                        Starting address of the segment's contents
                        within the memory space.
                      </xs:documentation>
520                 </xs:annotation>
                  </xs:attribute>
                </xs:complexType>
              </xs:element>
```

```
            </xs:sequence>
525       </xs:complexType>
        </xs:element>
    </xs:schema>
```

Table of Contents

530