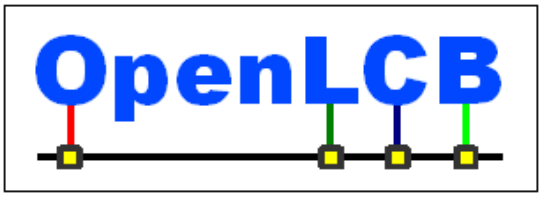

OpenLCB Standard	
Configuration Description Information	
March 30, 2023	Draft

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.

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 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 Variables, with each Variable having a specific type, a size in bytes, a value for Space and Address (to locate the variable), and Name and Description as user-readable strings so that users understand the use of the particular setting.

Variables can be grouped together, groups can be repeated (for example if a Node 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:

- 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:

- 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)

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 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)

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 1.3~~1.1~~ of the schema. That version of the schema is defined at http://openlcb.org/schema/cdi/1/3~~1~~/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 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:

```
<cdi xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance"
      xsi:noNamespaceSchemaLocation="http://openlcb.org/schema/cdi/1/3~~1~~/cdi.xsd">
```

to define the OpenLCB CDI version of the content.

The schema contents are normative.

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

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 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

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 Version 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

The value at the Version 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 Node also supports the OpenLCB Simple Node Information Protocol (SNIP), then the information provided by the SNIP Reply shall match the respective values provided in the ACDI space.

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

5.1.3 Element

A element defines the value of Space in the attribute `space', which shall apply to all Data Elements within, and the value of `origin', which shall be considered as the Address of a Data Element of size 0 (zero) at the beginning of the [3].

A Configuration Tool 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.

A element shall contain an optional user-readable name and optional description tags, and a sequence of zero or more Data Elements.

5.1.4 Data Elements

The following elements are considered Data Elements: <group>, <int>, <string>, <eventid>, <float>.

The value of the address within the segment is accumulated during a depth-first traversal of the contents of the segment definition element. Address 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 formalized as follows.

For each Data Element the following values are defined:

- Space, which is defined by the enclosing element.
- Address, which is defined as the End Address of the previous Data Element plus the value of the attribute `offset' on the Data Element.
- Size (in bytes)
- End Address, which is defined as Address + Size, unless otherwise specified.

5.1.4.1 <group> Element

The <group> element allows logical grouping of Variables, providing common documentation for them, and making multiple copies of the contained Variables. 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 of zero or more Data Elements. This sequence is considered to contain a Data Element of size 0 (zero) before the specified Data Elements[3].

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

The End Address of a <group> element is defined as the End Address of the last Data Element in the contained sequence (after replication). The Size of a <group> element is defined as the End Address minus the Address of the <group> element.

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

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

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

5.1.4.2 <int> Element

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

The Size of the <int> element is defined as the value of the `size' attribute in 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 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 Size 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 Address of the <string> 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

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

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

The Event 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 enumeration are invalid and shall not be written.

5.1.4.5 <float> Element

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

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.

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

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.

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 Tools 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.

Configuration Tools 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 Address and Size 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 `replication' attribute.
- All unknown tags that occur within the element or <group> and have an attribute `size' shall be considered to be Data Elements with Address defined as the End Address of the previous Data Element plus the value of the `offset' attribute, and Size defined as the value of the `size' attribute in bytes. The `size' attribute of all future Data Elements shall be 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"?>
<!-- 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">
    <xs:annotation>
      <xs:documentation>
        A map relates one or more property elements (keys)
        to specific values.
      </xs:documentation>
    </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">
        <xs:complexType>
          <xs:sequence>
            <xs:element name="property" minOccurs="1" maxOccurs="1" />
            <xs:element name="value" minOccurs="1" maxOccurs="1" />
          </xs:sequence>
        </xs:complexType>
      </xs:element>
    </xs:sequence>
  </xs:complexType>

  <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" />
      <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="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" />
        <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>
        <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
          immediately after the previous one.
        </xs:documentation>
      </xs:annotation>
    </xs:attribute>
    <xs:attribute name="replication" type="xs:int" default="1" />
  </xs:complexType>

  <xs:complexType name="eventidType">
    <xs:sequence>
      <xs:element name="name" minOccurs="0" maxOccurs="1" />
      <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>
```

```
        <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
          immediately after the previous one.
        </xs:documentation>
      </xs:annotation>
    </xs:attribute>
  </xs:complexType>

  <xs:complexType name="intType">
    <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" />
      <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>
            The 'value' of each entry is displayed, and
            the 'property' content (number) is sent
            to/from the node
          </xs:documentation>
        </xs:annotation>
      </xs:element>
    </xs:sequence>
    <xs:attribute name="size" type="xs:int" default="1">
      <xs:annotation>
        <xs:documentation>
          Storage size of this variable in bytes.
        </xs:documentation>
      </xs:annotation>
    </xs:attribute>
    <xs:attribute name="offset" type="xs:int" default="0">
      <xs:annotation>
        <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
          immediately after the previous one.
        </xs:documentation>
      </xs:annotation>
    </xs:attribute>
  </xs:complexType>

  <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    -->
        <!-- been seen and used before, however -->
        <xs:pattern value="%[0-9]?(\.[0-9])?f"/>
      </xs:restriction>
  </xs:simpleType>

  <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" />
      <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>
            The 'value' of each entry is displayed, and
            the 'property' content (number) is sent
            to/from the node
          </xs:documentation>
```

```
        </xs:annotation>
      </xs:element>
    </xs:sequence>
    <xs:attribute name="size" type="xs:int" default="4">
      <xs:annotation>
        <xs:documentation>
          Storage size of this variable in bytes.
        </xs:documentation>
      </xs:annotation>
    </xs:attribute>
    <xs:attribute name="offset" type="xs:int" default="0">
      <xs:annotation>
        <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
          immediately after the previous one.
        </xs:documentation>
      </xs:annotation>
    </xs:attribute>
      <xs:attribute name="formatting" type="floatFormat" >
          <xs:annotation>
        <xs:documentation>
          printf-style format string for displaying data to the user.
        </xs:documentation>
      </xs:annotation>
      </xs:attribute>
  </xs:complexType>

  <xs:complexType name="stringType">
    <xs:sequence>
      <xs:element name="name" minOccurs="0" maxOccurs="1" />
      <xs:element name="description" minOccurs="0" maxOccurs="1" />
      <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>
          Storage size of this variable in bytes.
          This includes the trailing null byte that
          terminates the string content.
        </xs:documentation>
      </xs:annotation>
    </xs:attribute>
    <xs:attribute name="offset" type="xs:int" default="0">
      <xs:annotation>
        <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
          immediately after the previous one.
        </xs:documentation>
      </xs:annotation>
    </xs:attribute>
  </xs:complexType>

  <xs:element name="cdi">
    <xs:annotation>
      <xs:documentation>
        This is the schema for Configuration
        Description Information (cdi)
      </xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <xs:sequence>
        <xs:element name="identification" minOccurs="0" maxOccurs="1">
          <xs:annotation>
            <xs:documentation>
              Common first element to identify the decoder
```

```
      </xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <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" />
        <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.
            </xs:documentation>
          </xs:annotation>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
  <xs:element name="acdi" minOccurs="0" maxOccurs="1">
    <xs:annotation>
      <xs:documentation>
        Element that identifies that memory information is available
        as defined by the Abbreviated Common Description Information
        (ACDI) standard.
      </xs:documentation>
    </xs:annotation>
    <xs:complexType>
      <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.
          </xs:documentation>
        </xs:annotation>
      </xs:attribute>
      <xs:attribute name="var" type="xs:int" default="2">
        <xs:annotation>
          <xs:documentation>
            The decimal version number of the format for
            the variable information block.
          </xs:documentation>
        </xs:annotation>
      </xs:attribute>
    </xs:complexType>
  </xs:element>
  <xs:element name="segment" minOccurs="0" maxOccurs="unbounded">
    <xs:annotation>
      <xs:documentation>
        Define the contents of a 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="string" type="stringType" minOccurs="0" maxOccurs="1">
            <xs:annotation>
```

```
                    <xs:documentation>
                      Describes a human-readable UTF-8 string in the data.
                    </xs:documentation>
                  </xs:annotation>
                </xs:element>
                <xs:element name="int" type="intType" minOccurs="0" maxOccurs="1">
                  <xs:annotation>
                    <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>
                  </xs:annotation>
                </xs:element>
                <xs:element name="eventid" type="eventidType" minOccurs="0" maxOccurs="1">
                  <xs:annotation>
                    <xs:documentation>
                      Describes an 8-byte Event ID in the data.
                    </xs:documentation>
                  </xs:annotation>
                </xs:element>
<!--
                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
xs:integer)"/>
                <xs:any minOccurs="0" maxOccurs="1" processContents="lax">
                  <xs:annotation>
                    <xs:documentation>
                      Extension point for future schema
                    </xs:documentation>
                  </xs:annotation>
                </xs:any>
 -->

              </xs:choice>
            </xs:sequence>
            <xs:attribute name="space" type="xs:int" use="required">
              <xs:annotation>
                <xs:documentation>
                  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">
              <xs:annotation>
                <xs:documentation>
                  Starting address of the segment's contents
                  within the memory space.
                </xs:documentation>
              </xs:annotation>
            </xs:attribute>
          </xs:complexType>
        </xs:element>
      </xs:sequence>
    </xs:complexType>
  </xs:element>
</xs:schema>
```

510

Table of Contents

1 Introduction (Informative)....1
2 Intended Use (Informative)....1
3 References and Context (Informative)....1
4 Content (Normative)....2
5 Format (Normative)....2
5.1 XML Elements....3
5.1.1 <identification> Element....3
5.1.2 <acdi> Element....3
5.1.3 Element....4
5.1.4 Data Elements....4
5.1.4.1 <group> Element....4
5.1.4.2 <int> Element....5
5.1.4.3 <string> Element....5
5.1.4.4 <eventid> Element....5
6 Future Extension (Normative)....5
A Appendix: Schema....7