OpenLCB Standard	
Configuration Description Information	
Feb 13, 2015	In Review

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 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 Standard: Message Network

For information on XML encoding and XML Schema, see:

- World Wide Web Consortium (W3C) "Extensible Markup Language (XML)"[i]
- World Wide Web Consortium (W3C) "XML Schema"[ii]

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 standard 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 with 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 0x12 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 1.1 of the schema. That version of the schema is defined at http://openlcb.org/schema/1/1/cdi.xsl and in Appendix A of this document. The CDI content shall pass validation against the particular schema. Nodes are not required to check this.

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/1/1/cdi.xsd">
```

The schema contents are normative.

Numerical values in attributes and element text shall be specified as decimal numbers.

5.1 XML Elements

5.1.1 <identification> Element

The <identification> element, if included, 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', with the attribute `fixed="1"', `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', with the attribute `var="1"', `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 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 an Element of size 0 (zero) at the beginning of the [1].

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 description tags, and a sequence of zero or more Data Elements.

5.1.4 Data Elements

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

For each Data Element the following values are defined:

- Space
- Address
- Size
- End Address, which shall be Address + Size.

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 occupying a contiguous memory area. Nodes 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, 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[1].

The Address of a <group> is defined as the Address of the previous Element plus the value of the attribute `offset'.

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, with a Data Element of size 0 (zero) separating the individual instances.

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.

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

5.1.4.2 <bit> Element

The <bit> Element defines a Variable of boolean value.

The Address of the <bit> Element is defined as the End Address of the previous Data Element plus the value of the `offset' attribute.

The Size of the <bit> Element is defined as the value of the `size' attribute divided by 8 rounded up to the next integer.

A value of true / on / yes shall be represented by writing a non-zero integer value to the data pointed to. A value of false / off / no shall be represented by writing zeros to the data pointed to.

5.1.4.3 <int> Element

The <int> Element defines a Variable of boolean value.

The Address of the <int> Element is defined as the End Address of the previous Data Element plus the value of the `offset' attribute.

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 falling outside of the range defined by the <min> and <max> sub-elements, 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.

5.1.4.4 <string> Element

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

The Address of the <string> Element is defined as the End Address of the previous Data Element plus the value of the `offset' attribute.

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. When writing a shorter string, any unused bytes shall be set to 0 (zero).

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.

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 1.1" 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="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="bit" type="bitType" 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: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 previous element and the start of this group's contents.
        </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.
        </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:complexType name="bitType">
    <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: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 bits.
        </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
          previous element and the start of this element'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="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
          previous element and the start of this element'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
            definition information. See ACDI Specification.
          </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 definition information.  See ACDI Specification.
          </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="bit" type="bitType" minOccurs="0" maxOccurs="1">
            <xs:annotation>
              <xs:documentation>
                Describes a bit field 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="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: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" 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>
```

Table of Contents

5 Format (normative)....2
5.1 XML Elements....2
5.1.1 <identification> Element....2
5.1.2 <acdi> Element....2
5.1.3 Element....2
6 Future Extension....2
7 Appendix....2

i http://www.w3.org/XML/
ii http://www.w3.org/XML/Schema