



OpenLCB Standard	
Configuration Description Information	
August 7, 2024	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*

- 5 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

- 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 variables, with each variable having a specific type, a size in bytes, a value for its memory 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.

- 15 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

- 20 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

- 25 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 1.⁴³ of the schema. That version of the schema is defined at <https://openlcb.org/schema/cdi/1/43/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="https://openlcb.org/schema/cdi/1/43/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.

60 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. An optional <link> sub-element can provide an optional hyperlink to additional information. An optional <map> sub-element can provide one or more key-value pairs to be optionally displayed.

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.

70 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 <segment> Element

85 A <segment> element defines the value of the memory 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 <segment>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.

90 A <segment> element shall contain an optional user-readable name and optional description tags, and a sequence of zero or more data elements. The user-readable name and description are intended as hints for optional UI display by configuration tools. [An optional <link> sub-element can provide an optional hyperlink to additional information.](#)

5.1.4 Data Elements

95 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. The address is initialized with the value of the attribute 'origin' on the <segment> 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

100 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 <segment> 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.

The data element's <name> and <description> elements and the <group> element's <repname>

110 element are intended as hints for optional UI display by configuration tools.

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

115 (such as multiple input ports, output ports etc).

A <group> element shall contain an optional user-readable name, optional description, [optional link information, optional display hints, optional replication-name information](#), and a sequence of zero or

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

more data elements. This sequence is considered to contain a data element of size 0 (zero) before the specified data elements³.

120 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 optional <link> sub-element can provide an optional hyperlink to additional information.

The optional <hints> sub-element provides hints to a configuration tool on how it might best display the integer variable. No specific behavior is required by the presence of this element. Available sub-elements for the <hints> element are <visibility> and <readOnly>.

- <visibility> has attributes "hideable" and "hidden" that recommend whether the group be displayed in a form that the user can hide, and recommend whether a hideable group is initially displayed as hidden.
- <readOnly>, if present, recommends that the contents of the group be considered to be read-only values which cannot be written.

130 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.

Configuration Tools shall not render a <group> element with no child elements⁴ on their UI.

135 **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. Valid values are 1, 2, 4 and 8 bytes.

140 The integer value shall be stored as an unsigned integer unless a <min> sub-element with a value less than zero is present, in which case the integer value shall be stored as a signed integer. 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. The default value of the <min> sub-element is zero, and the default value of the <max> sub-element is the largest possible value for the given size.

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

The optional <hints> sub-element provides hints to a configuration tool on how it might best display the integer variable. No specific behavior is required by the presence of this element. Available sub-elements for the <hints> element are <slider>, <radiobutton> and <checkbox>.

- <slider> hints that the configuration tool should present the variable value as a slider between the min and max values.

⁴No name, no description, no link and no Data Elements contained.

- The optional “tickSpacing” attribute recommends a value for the spacing between tick marks on the slider. A value of 0 (the default) recommends that no ticks be shown.
- The optional “immediate” attribute recommends that the slider value be written to the node’s configuration memory immediately when the slider value is changed, without waiting for a “Write” button to be pressed.
- The optional “showValue” attribute recommends that the Configuration Tool show the decimal slider value somewhere near the slider on the display.
- <radiobutton> hints that the configuration tool should present the variable value as a set of radio buttons corresponding to the <property> elements in the associated <map> element. The <map> element must be present.
- <checkbox> hints that the configuration tool should present the variable value as a checkbox. The unchecked value corresponds to the first <property> element in the associated <map> element. The checked value corresponds to the second <property> element in the associated <map> element. The <map> element must be present. It is an error if there are not exactly two map entries present.

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.

- 170 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.

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

- 175 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).

- 180 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.

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. The default value of the <min> sub-element is zero, and the default value of the <max> sub-element is the largest possible value for the given size.

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.

195 | **5.1.4.6 <action> Element**

The <action> element defines a write-only variable that causes the target node to perform some specific action. It may be linked to e.g. a button in the GUI.

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

200 | The optional <buttonText> sub-element, when present, is a hint to the rendering code about what should appear in the user interface e.g. as the text on the button that triggers the action's write operation. Note that no specific behavior is required.

205 | The optional <dialogText> sub-element, when non-blank, suggests to the GUI that it would be good to provide an additional warning to the user before triggering the action's write operation. This could be done by e.g. presenting an OK/Confirm dialog with the dialogText value. A blank or not-present value for the element is a hint that no additional warning is requested. Note that no specific behavior is required.

The required <value> sub-element is the specific decimal value to be written when the action is triggered.

210 | The write-only variable defined by the <action> element must not be written other than when the action is to be triggered.

5.1.4.7 <blob> Element

The <blob> element defines a region of memory that a GUI can transfer to an external file (“read”), transfer from an external file (“write”) or both (“readwrite”).

215 | The required size and optional offset attributes define a ten-byte section of memory containing:

- 1 byte operation flag;
- 5 bytes of a memory space followed by a four-byte address of the memory blob;
- 4 bytes of length of the memory blob.

To read the blob data:

- 220 | • Write a value of 1 to the operation flag location.
 | • Read the memory space and address of the memory blob.
 | • Read the length of the memory blob.
 | • Use the Memory Configuration Protocol to read the data.
- | To write the blob data:
- 225 | • Write a value of 1 to the operation flag location.
 | • Read the memory space and address of the memory blob.
 | • Write the length of the data to be written. The node being configured will positively acknowledge this write if the data can be accepted, and negatively acknowledge it if the data can't be accepted.
- 230 | • Use the Memory Configuration protocol to write the data.

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

235 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.
- 240 • All unknown tags that occur within the element <segment> 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.

245

A Appendix: Schema

```

<?xml version="1.0" encoding="utf-8"?>
<?xmlstylesheet href="schema2xhtml.xsl" type="text/xsl"?>
<!DOCTYPE XML_Schema_for_OpenLCB_Configuration_Description_Information (CDI)>
<xss: schema version="CDI 1.3" xmlns:xss="http://www.w3.org/2001/XMLSchema"
  xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance">
  ...
  <xss:complexType name="mapType">
    <xss:annotation>
      <xss:documentation>
        A map relates one or more property elements (keys)
        to specific values.
      </xss:documentation>
    </xss:annotation>
    <xss:sequence>
      <xss:element name="name" minOccurs="0" maxOccurs="1" />
      <xss:element name="description" minOccurs="0" maxOccurs="1" />
      <xss:element name="relation" minOccurs="0" maxOccurs="unbounded">
        <xss:complexType>
          <xss:sequence>
            <xss:element name="property" minOccurs="1" maxOccurs="1" />
            <xss:element name="value" minOccurs="1" maxOccurs="1" />
          </xss:sequence>
        </xss:complexType>
      </xss:element>
    </xss:sequence>
  </xss:complexType>
  ...
  <xss:complexType name="groupType">
    <xss:sequence>
      <xss:element name="name" minOccurs="0" maxOccurs="1" />
      <xss:element name="description" minOccurs="0" maxOccurs="1" />
      <xss:element name="repname" minOccurs="0" maxOccurs="unbounded" />
      <xss:choice minOccurs="0" maxOccurs="unbounded">
        <xss:annotation>
          Allows any sequence of the contained element types
        </xss:annotation>
        <xss:documentation>
        </xss:documentation>
        <xss:element name="group" type="groupType" minOccurs="0" maxOccurs="1" />
        <xss:element name="string" type="stringType" minOccurs="0" maxOccurs="1" />
        <xss:element name="int" type="intType" minOccurs="0" maxOccurs="1" />
        <xss:element name="eventid" type="eventidType" minOccurs="0" maxOccurs="1" />
        <xss:element name="float" type="floatType" minOccurs="0" maxOccurs="1" />
      </xss:choice>
    </xss:sequence>
    <xss:attribute name="offset" type="xs:int" default="0">
      <xss:annotation>
        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.
      </xss:annotation>
    </xss:attribute>
    <xss:attribute name="replication" type="xs:int" default="1" />
  </xss:complexType>
  ...
  <xss:complexType name="eventidType">
    <xss:sequence>
      <xss:element name="name" minOccurs="0" maxOccurs="1" />
      <xss:element name="description" minOccurs="0" maxOccurs="1" />
      <xss:element name="map" type="mapType" minOccurs="0" maxOccurs="1" />
    </xss:sequence>
    <xss:attribute name="offset" type="xs:int" default="0">
      <xss:annotation>

```

```

315      <xs:documentation>
316      Positive or negative offset between the address of
317      the end of previous element and the start of
318      this elements's contents.
319      Offset of zero means that this element starts
320      immediately after the previous one.
321      </xs:documentation>
322      </xs:annotation>
323      </xs:attribute>
324      </xs:complexType>
325
326      <xs:complexType name="intType">
327      <xs:sequence>
328          <xs:element name="name" minOccurs="0" maxOccurs="1" />
329          <xs:element name="description" minOccurs="0" maxOccurs="1" />
330          <xs:element name="min" minOccurs="0" maxOccurs="1" />
331          <xs:element name="max" minOccurs="0" maxOccurs="1" />
332          <xs:element name="default" minOccurs="0" maxOccurs="1" />
333          <xs:element name="map" type="mapType" minOccurs="0" maxOccurs="1" />
334          <xs:annotation>
335              <xs:documentation>
336                  The 'value' of each entry is displayed, and
337                  the 'property' content (number) is sent
338                  to/from the node
339              </xs:documentation>
340          </xs:annotation>
341      </xs:element>
342      </xs:sequence>
343      <xs:attribute name="size" default="1">
344          <xs:annotation>
345              <xs:documentation>
346                  Storage size of this variable in bytes.
347              </xs:documentation>
348          </xs:annotation>
349          <xs:simpleType>
350              <xs:restriction base="xs:token">
351                  <xs:enumeration value="1"/>
352                  <xs:enumeration value="2"/>
353                  <xs:enumeration value="4"/>
354                  <xs:enumeration value="8"/>
355              </xs:restriction>
356          </xs:simpleType>
357      </xs:attribute>
358      <xs:attribute name="offset" type="xs:int" default="0">
359          <xs:annotation>
360              <xs:documentation>
361                  Positive or negative offset between the
362                  address of the end of previous element and the
363                  start of this elements's contents.
364                  Offset of zero means that this element starts
365                  immediately after the previous one.
366              </xs:documentation>
367          </xs:annotation>
368      </xs:attribute>
369      </xs:complexType>
370
371      <xs:simpleType name="floatFormat">
372          <xs:restriction base="xs:string">
373              <!-- This is a somewhat limiting regex, as it does not allow all possible -->
374              <!-- printf formats. It will allow the most common formats that have -->
375              <!-- been seen and used before, however -->
376              <xs:pattern value="%[0-9]*(\.(\\([0-9]*))?)f"/>
377          </xs:restriction>
378      </xs:simpleType>
379
380      <xs:complexType name="floatType">
381          <xs:sequence>
382              <xs:element name="name" minOccurs="0" maxOccurs="1" />
383              <xs:element name="description" minOccurs="0" maxOccurs="1" />
384              <xs:element name="min" minOccurs="0" maxOccurs="1" />
385              <xs:element name="max" minOccurs="0" maxOccurs="1" />

```

```

385   <xs:element name="default" minOccurs="0" maxOccurs="1" />
386   <xs:element name="map" type="mapType" minOccurs="0" maxOccurs="1" />
387     <xs:annotation>
388       <xs:documentation>
389         The 'value' of each entry is displayed, and
390         the 'property' content (number) is sent
391       </xs:documentation>
392     </xs:annotation>
393   </xs:element>
394 </xs:sequence>
395   <xs:attribute name="size" use="required">
396     <xs:annotation>
397       <xs:documentation>
398         Storage size of this variable in bytes.
399       </xs:documentation>
400     </xs:annotation>
401   <xs:simpleType>
402     <xs:restriction base="xs:token">
403       <xs:enumeration value="2"/>
404       <xs:enumeration value="4"/>
405       <xs:enumeration value="8"/>
406     </xs:restriction>
407   </xs:simpleType>
408 </xs:attribute>
409   <xs:attribute name="offset" type="xs:int" default="0">
410     <xs:annotation>
411       <xs:documentation>
412         Positive or negative offset between the
413         address of the end of previous element and the
414         start of this elements's contents.
415         Offset of zero means that this element starts
416         immediately after the previous one.
417       </xs:documentation>
418     </xs:annotation>
419   </xs:attribute>
420   <xs:attribute name="formatting" type="floatFormat">
421     <xs:annotation>
422       <xs:documentation>
423         printf style format string for displaying data to the user, like %3.1f
424       </xs:documentation>
425     </xs:annotation>
426   </xs:attribute>
427 </xs:complexType>
428
429   <xs:complexType name="stringType">
430     <xs:sequence>
431       <xs:element name="name" minOccurs="0" maxOccurs="1" />
432       <xs:element name="description" minOccurs="0" maxOccurs="1" />
433       <xs:element name="map" type="mapType" minOccurs="0" maxOccurs="1" />
434     </xs:sequence>
435     <xs:attribute name="size" type="xs:int" use="required">
436       <xs:annotation>
437         <xs:documentation>
438           Storage size of this variable in bytes.
439           This includes the trailing null byte that
440           terminates the string content.
441         </xs:documentation>
442       </xs:annotation>
443     </xs:attribute>
444     <xs:attribute name="offset" type="xs:int" default="0">
445       <xs:annotation>
446         <xs:documentation>
447           Positive or negative offset between the
448           address of the end of previous element and the
449           start of this elements's contents.
450           Offset of zero means that this element starts
451           immediately after the previous one.
452         </xs:documentation>
453       </xs:annotation>
454     </xs:attribute>

```

```

455  </xs:complexType>
456  ...
457  <xs:element name="cdi">
458  <xs:annotation>
459  <xs:documentation>
460  This is the schema for Configuration
461  Description Information (cdi)
462  </xs:documentation>
463  </xs:annotation>
464  <xs:complexType>
465  <xs:sequence>
466  <xs:element name="identification" minOccurs="0" maxOccurs="1">
467  <xs:annotation>
468  <xs:documentation>
469  Common first element to identify the decoder
470  </xs:documentation>
471  </xs:annotation>
472  <xs:complexType>
473  <xs:sequence>
474  <xs:element name="manufacturer" minOccurs="0" maxOccurs="1" />
475  <xs:element name="model" minOccurs="0" maxOccurs="1" />
476  <xs:element name="hardwareVersion" minOccurs="0" maxOccurs="1" />
477  <xs:element name="softwareVersion" minOccurs="0" maxOccurs="1" />
478  <xs:element name="map" type="mapType" minOccurs="0" maxOccurs="1">
479  <xs:annotation>
480  <xs:documentation>
481  This map can be used to add arbitrary key/value-
482  descriptions of the node.
483  </xs:documentation>
484  </xs:annotation>
485  </xs:element>
486  </xs:sequence>
487  </xs:complexType>
488  </xs:element>
489  <xs:element name="acdi" minOccurs="0" maxOccurs="1">
490  <xs:annotation>
491  <xs:documentation>
492  Element that identifies that memory information is available
493  as defined by the Abbreviated Common Description Information
494  (ACDI) standard.
495  </xs:documentation>
496  </xs:annotation>
497  <xs:complexType>
498  <xs:attribute name="fixed" type="xs:int" default="4">
499  <xs:annotation>
500  <xs:documentation>
501  The decimal version number of the format for the fixed-
502  information block.
503  </xs:documentation>
504  </xs:annotation>
505  </xs:attribute>
506  <xs:attribute name="var" type="xs:int" default="2">
507  <xs:annotation>
508  <xs:documentation>
509  The decimal version number of the format for
510  the variable information block.
511  </xs:documentation>
512  </xs:annotation>
513  </xs:attribute>
514  </xs:complexType>
515  </xs:element>
516  <xs:element name="segment" minOccurs="0" maxOccurs="unbounded">
517  <xs:annotation>
518  <xs:documentation>
519  Define the contents of a memory space
520  </xs:documentation>
521  </xs:annotation>
522  <xs:complexType>
523  <xs:sequence>
524  <xs:element name="name" minOccurs="0" maxOccurs="1" />
525  <xs:element name="description" minOccurs="0" maxOccurs="1" />

```

```

530      <xs:choice minOccurs="0" maxOccurs="unbounded">
531          <xs:annotation>
532              <xs:documentation>
533                  Allows any sequence of the contained element types
534              </xs:documentation>
535          </xs:annotation>
536          <xs:element name="group" type="groupType" minOccurs="0" maxOccurs="1">
537              <xs:annotation>
538                  <xs:documentation>
539                      Allows grouping and replication of multiple locations.
540                  </xs:documentation>
541              </xs:annotation>
542          </xs:element>
543          <xs:element name="string" type="stringType" minOccurs="0" maxOccurs="1">
544              <xs:annotation>
545                  <xs:documentation>
546                      Describes a human readable UTF 8 string in the data.
547                  </xs:documentation>
548              </xs:annotation>
549          </xs:element>
550          <xs:element name="int" type="intType" minOccurs="0" maxOccurs="1">
551              <xs:annotation>
552                  <xs:documentation>
553                      Describes an integer value in the data.
554                      The field can be considered either a number,
555                      or a set of specific coded values via a map.
556                  </xs:documentation>
557              </xs:annotation>
558          </xs:element>
559          <xs:element name="eventid" type="eventidType" minOccurs="0" maxOccurs="1">
560              <xs:annotation>
561                  <xs:documentation>
562                      Describes an 8 byte Event ID in the data.
563                  </xs:documentation>
564              </xs:annotation>
565          </xs:element>

566      <!--
567          XML Schema 1.1 construct expressing extensibility promise
568          <xs:assert test="every $x in * satisfies (exists($x/@size) and $x/@size castable to
569          xs:integer)" />
570          <xs:assert test="every $x in * satisfies (exists($x/@offset) and $x/@offset castable to
571          xs:integer)" />
572          <xs:any minOccurs="0" maxOccurs="1" processContents="lax">
573              <xs:annotation>
574                  <xs:documentation>
575                      Extension point for future schema
576                  </xs:documentation>
577              </xs:annotation>
578          </xs:any>
579      -->

580      </xs:choice>
581      </xs:sequence>
582      <xs:attribute name="space" type="xs:int" use="required">
583          <xs:annotation>
584              <xs:documentation>
585                  The decimal number of the address space where the information is found.
586              </xs:documentation>
587          </xs:annotation>
588      </xs:attribute>
589      <xs:attribute name="origin" type="xs:int" default="0">
590          <xs:annotation>
591              <xs:documentation>
592                  Starting address of the segment's contents
593                  within the memory space.
594              </xs:documentation>
595          </xs:annotation>
596      </xs:attribute>
597      </xs:complexType>
598  </xs:element>

```

```
600     </xs:sequence>
      </xs:complexType>
      </xs:element>
    </xs:schema>
<?xml version="1.0" encoding="utf-8"?>
<?xmlstylesheet href="schema2xhtml.xsl" type="text/xsl"?>
<!-- XML Schema for OpenLCB Configuration Description Information (CDI) -->
<xs:schema version="CDI 1.4" xmlns:xs="http://www.w3.org/2001/XMLSchema"
605           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>
620
625
630
635
640
<xs:simpleType name="booleanType">
  <xs:annotation>
    <xs:documentation>
      General definition for a string that represents a boolean
      via yes/true/1 and no/false/0 values
    </xs:documentation>
  </xs:annotation>
  <xs:restriction base="xs:token">
    <xs:enumeration value="yes"/>
    <xs:enumeration value="no"/>
    <xs:enumeration value="true"/>
    <xs:enumeration value="false"/>
    <xs:enumeration value="1"/>
    <xs:enumeration value="0"/>
  </xs:restriction>
</xs:simpleType>
```

```

645 <xs:complexType name="groupHintsType">
646   <xs:annotation>
647     <xs:documentation>
648       Presents optional hints to a GUI program
649       for how to render the contents of this group.
650     </xs:documentation>
651   </xs:annotation>
652   <xs:sequence>
653     <xs:element name="visibility" minOccurs="0" maxOccurs="1" >
654       <xs:complexType>
655         <xs:attribute name="hideable" type="booleanType" default="no">
656           <xs:annotation>
657             <xs:documentation>
658               Recommends that the group be
659               presented in a form that can be made visible
660               or hidden at the user's request.
661             </xs:documentation>
662           </xs:annotation>
663         </xs:attribute>
664         <xs:attribute name="hidden" type="booleanType" default="no">
665           <xs:annotation>
666             <xs:documentation>
667               If yes, recommends a hideable=true group
668               be hidden when first displayed
669             </xs:documentation>
670           </xs:annotation>
671         </xs:attribute>
672         <xs:complexType>
673           <xs:element>
674             <xs:element name="readOnly" minOccurs="0" maxOccurs="1" />
675           </xs:sequence>
676         </xs:complexType>

677         <xs:complexType name="groupType">
678           <xs:sequence>
679             <xs:element name="name" minOccurs="0" maxOccurs="1" />
680             <xs:element name="description" minOccurs="0" maxOccurs="1" />
681             <xs:element name="link" type="linkType" minOccurs="0" maxOccurs="1" />
682             <xs:element name="repname" minOccurs="0" maxOccurs="unbounded" />
683             <xs:element name="hints" type="groupHintsType" minOccurs="0" maxOccurs="1" />
684           <xs:choice minOccurs="0" maxOccurs="unbounded">
685             <xs:annotation>
686               <xs:documentation>
687                 Allows any sequence of the contained element types
688               </xs:documentation>
689             </xs:annotation>
690           </xs:choice>

```

695

```

<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:element name="action" type="actionButtonType" minOccurs="0" maxOccurs="1" />
<xs:element name="blob" type="blobType" 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="integerHintsType">
<xs:annotation>
<xs:documentation>
```

Presents optional hints to a GUI program

for how to render this integer variable.

```

</xs:documentation>
</xs:annotation>
740 <xs:sequence>
<xs:element name="slider" minOccurs="0" maxOccurs="1" >
<xs:complexType>
<xs:attribute name="tickSpacing" type="xs:integer" default="0">
<xs:annotation>
745 <xs:documentation>
    Recommends a value for the spacing between
    tick marks on the slider. For example, "10" with a
    minimum value of 0 would recommend ticks at 0, 10, 20 etc
    along the slider axis. Values of zero or less
    recommend that no tick marks be displayed.
50 </xs:documentation>
</xs:annotation>
</xs:attribute>
<xs:attribute name="immediate" type="booleanType" default="no">
55 <xs:annotation>
<xs:documentation>
    If yes, recommends that the slider should write to the
    configuration memory immediately when changed
</xs:documentation>
60 </xs:annotation>
</xs:attribute>
<xs:attribute name="showValue" type="booleanType" default="no">
65 <xs:annotation>
<xs:documentation>
    If yes, recommends that a slider should also
    show the exact value, perhaps in a near-by numeric field
</xs:documentation>
70 </xs:annotation>
</xs:attribute>
</xs:complexType>
770 </xs:element>
<xs:element name="radiobutton" minOccurs="0" maxOccurs="1" />
<xs:element name="checkbox" minOccurs="0" maxOccurs="1" />
</xs:sequence>
775 </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" />
880 <xs:element name="max" minOccurs="0" maxOccurs="1" />
```

```

785   <xs:element name="default" minOccurs="0" maxOccurs="1" />
786   <xs:element name="map" type="mapType" minOccurs="0" maxOccurs="1">
787     <xs:annotation>
788       <xs:documentation>
789         The 'value' of each entry is displayed, and
790         the 'property' content (number) is sent
791         to/from the node
792       </xs:documentation>
793     </xs:annotation>
794   </xs:element>
795   <xs:element name="hints" type="integerHintsType" minOccurs="0" maxOccurs="1" />
796   </xs:sequence>
797   <xs:attribute name="size" default="1">
798     <xs:annotation>
799       <xs:documentation>
800         Storage size of this variable in bytes.
801       </xs:documentation>
802     </xs:annotation>
803     <xs:simpleType>
804       <xs:restriction base="xs:token">
805         <xs:enumeration value="1"/>
806         <xs:enumeration value="2"/>
807         <xs:enumeration value="4"/>
808         <xs:enumeration value="8"/>
809       </xs:restriction>
810     </xs:simpleType>
811   </xs:attribute>
812   <xs:attribute name="offset" type="xs:int" default="0">
813     <xs:annotation>
814       <xs:documentation>
815         Positive or negative offset between the
816         address of the end of previous element and the
817         start of this elements's contents.
818         Offset of zero means that this element starts
819         immediately after the previous one.
820       </xs:documentation>
821     </xs:annotation>
822   </xs:attribute>
823   </xs:complexType>

824   <xs:simpleType name="floatFormat">
825     <xs:restriction base="xs:string">
826       <!-- This is a somewhat limiting regex, as it does not allow all possible -->
827       <!-- printf formats. It will allow the most common formats that I have -->
828       <!-- seen and used before, however -->
829       <xs:pattern value="%[0-9]*(\.([0-9]*))?f"/>

```

```

830      </xs:restriction>
830      </xs:simpleType>

835      <xs:complexType name="floatType">
835          <xs:sequence>
835              <xs:element name="name" minOccurs="0" maxOccurs="1" />
835              <xs:element name="description" minOccurs="0" maxOccurs="1" />
835              <xs:element name="min" minOccurs="0" maxOccurs="1" />
835              <xs:element name="max" minOccurs="0" maxOccurs="1" />
835              <xs:element name="default" minOccurs="0" maxOccurs="1" />
835              <xs:element name="map" type="mapType" minOccurs="0" maxOccurs="1">
840                  <xs:annotation>
840                      <xs:documentation>
840                          The 'value' of each entry is displayed, and
840                          the 'property' content (number) is sent
840                          to/from the node
845                  </xs:documentation>
845                  </xs:annotation>
845                  </xs:element>
845                  <xs:sequence>
845                      <xs:attribute name="size" use="required">
850                          <xs:annotation>
850                              <xs:documentation>
850                                  Storage size of this variable in bytes. 2, 4 and 8 are valid.
850                              </xs:documentation>
850                          </xs:annotation>
855                      <xs:simpleType>
855                          <xs:restriction base="xs:token">
855                              <xs:enumeration value="2"/>
855                              <xs:enumeration value="4"/>
855                              <xs:enumeration value="8"/>
860                          </xs:restriction>
860                      </xs:simpleType>
860                      <xs:attribute>
860                          <xs:attribute name="offset" type="xs:int" default="0">
865                          <xs:annotation>
865                              <xs:documentation>
865                                  Positive or negative offset between the
865                                  address of the end of previous element and the
865                                  start of this elements's contents.
865                                  Offset of zero means that this element starts
870                                  immediately after the previous one.
870                              </xs:documentation>
870                          </xs:annotation>
870                          <xs:attribute>
870                              <xs:attribute name="formatting" type="floatFormat" >

```

875 `<xs:annotation>`
 `<xs:documentation>`
 printf-style format string for displaying data to the user, like %3.1f
 `</xs:documentation>`
 `</xs:annotation>`
 `</xs:attribute>`
 `</xs:complexType>`

880

885 `<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>`

900

905

910 `<xs:complexType name="actionButtonType">`
 `<xs:sequence>`
 `<xs:element name="name" minOccurs="0" maxOccurs="1" />`
 `<xs:element name="description" minOccurs="0" maxOccurs="1" />`
 `<xs:element name="buttonText" minOccurs="0" maxOccurs="1">`
 `<xs:annotation>`
 `<xs:documentation>`
 Text to be displayed on the button.
 This is required.
 `</xs:documentation>`

915

920

925

```
</xs:annotation>
</xs:element>
<xs:element name="dialogText" minOccurs="0" maxOccurs="1">
  <xs:annotation>
    <xs:documentation>
      Text to be displayed on the confirmation dialog.
      If empty or not present, no confirmation dialog is displayed.
    </xs:documentation>
```

930

```
</xs:annotation>
</xs:element>
<xs:element name="value" minOccurs="1" maxOccurs="1">
  <xs:annotation>
    <xs:documentation>
      Value to be written when the button is triggered.
    This is required.
```

935

```
</xs:documentation>
</xs:annotation>
</xs:element>
```

940

```
</xs:sequence>
<xs:attribute name="size" use="required">
  <xs:annotation>
    <xs:documentation>
      Storage size of this variable in bytes.
      This is required for backwards compatibility.
```

945

```
</xs:documentation>
</xs:annotation>
<xs:simpleType>
  <xs:restriction base="xs:token">
    <xs:enumeration value="1"/>
    <xs:enumeration value="2"/>
    <xs:enumeration value="4"/>
    <xs:enumeration value="8"/>
```

950

```
</xs:restriction>
</xs:simpleType>
</xs:attribute>
```

955

```
<xs:attribute name="offset" type="xs:int" default="0">
  <xs:annotation>
    <xs:documentation>
```

960

```
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.
```

965

```
</xs:documentation>
</xs:annotation>
</xs:attribute>
```

970 </xs:complexType>

975 <xs:complexType name="blobType">
 <xs:sequence>
 <xs:element name="name" minOccurs="0" maxOccurs="1" />
 <xs:element name="description" minOccurs="0" maxOccurs="1" />
 </xs:sequence>
 <xs:attribute name="size" use="required">
 <xs:annotation>
 <xs:documentation>
 Storage size of this variable in bytes.
 This is fixed at 10 bytes.
 An explicit value is required for backwards compatibility.
980 </xs:documentation>
 </xs:annotation>
 <xs:simpleType>
 <xs:restriction base="xs:token">
 <xs:enumeration value="10"/>
985 </xs:restriction>
 </xs:simpleType>
 <xs:attribute>
 <xs:attribute name="offset" type="xs:int" default="0">
990 <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.
995 </xs:documentation>
 </xs:annotation>
 <xs:attribute>
 <xs:attribute name="mode" use="required">
1000 <xs:annotation>
 <xs:documentation>
 Whether the underlying blob can be read ("read"), can be written ("write"),
 or both ("readwrite").
 </xs:documentation>
1005 </xs:annotation>
 <xs:simpleType>
 <xs:restriction base="xs:token">
 <xs:enumeration value="read"/>
 <xs:enumeration value="write"/>
1010 <xs:enumeration value="readwrite"/>
 </xs:restriction>
 </xs:simpleType>

```
1015       </xs:attribute>
1016       </xs:complexType>
1017
1018       <xs:complexType name="linkType">
1019        <xs:simpleContent>
1020          <xs:extension base="xs:string">
1021           <xs:attribute name="ref" type="xs:string" use="required">
1022            <xs:annotation>
1023             <xs:documentation>
1024              URL for this link. When the link is triggered
1025              in a configuration tool, it's recommended that this
1026              link be retrieved and displayed.
1027            </xs:documentation>
1028           </xs:annotation>
1029          </xs:extension>
1030        </xs:simpleContent>
1031       </xs:complexType>
1032
1033       <!-- Definition of the primary "cdi" element -->
1034       <xs:element name="cdi">
1035        <xs:annotation>
1036          <xs:documentation>
1037           This is the schema for Configuration
1038           Description Information (cdi)
1039          </xs:documentation>
1040        </xs:annotation>
1041        <xs:complexType>
1042          <xs:sequence>
1043           <xs:element name="identification" minOccurs="0" maxOccurs="1">
1044            <xs:annotation>
1045             <xs:documentation>
1046              Common first element to identify the decoder
1047             </xs:documentation>
1048            </xs:annotation>
1049          <xs:complexType>
1050            <xs:sequence>
1051             <xs:element name="manufacturer" minOccurs="0" maxOccurs="1" />
1052             <xs:element name="model" minOccurs="0" maxOccurs="1" />
1053             <xs:element name="hardwareVersion" minOccurs="0" maxOccurs="1" />
1054             <xs:element name="softwareVersion" minOccurs="0" maxOccurs="1" />
1055             <xs:element name="link" type="linkType" minOccurs="0" maxOccurs="1" />
1056             <xs:element name="map" type="mapType" minOccurs="0" maxOccurs="1">
1057               <xs:annotation>
1058                <xs:documentation>
1059                 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:element name="link" type="linkType" minOccurs="0" maxOccurs="1" />

<xs:choice minOccurs="0" maxOccurs="unbounded">

1105 <xs:annotation>
1106 <xs:documentation>
1107 Allows any sequence of the contained element types
1108 </xs:documentation>
1109 </xs:annotation>
1110 <xs:element name="group" type="groupType" minOccurs="0" maxOccurs="1">
1111 <xs:annotation>
1112 <xs:documentation>
1113 Allows grouping and replication of multiple locations.
1114 </xs:documentation>
1115 </xs:annotation>
1116 </xs:element>
1117 <xs:element name="string" type="stringType" minOccurs="0" maxOccurs="1">
1118 <xs:annotation>
1119 <xs:documentation>
1120 Describes a human-readable UTF-8 string in the data.
1121 </xs:documentation>
1122 </xs:annotation>
1123 </xs:element>
1124 <xs:element name="int" type="intType" minOccurs="0" maxOccurs="1">
1125 <xs:annotation>
1126 <xs:documentation>
1127 Describes an integer value in the data.
1128 The field can be considered either a number,
1129 or a set of specific coded values via a map.
1130 </xs:documentation>
1131 </xs:annotation>
1132 </xs:element>
1133 <xs:element name="eventid" type="eventidType" minOccurs="0" maxOccurs="1">
1134 <xs:annotation>
1135 <xs:documentation>
1136 Describes an 8-byte Event ID in the data.
1137 </xs:documentation>
1138 </xs:annotation>
1139 </xs:element>
1140 <xs:element name="float" type="floatType" minOccurs="0" maxOccurs="1">
1141 <xs:annotation>
1142 <xs:documentation>
1143 Describes a float type in the data
1144 </xs:documentation>
1145 </xs:annotation>
1146 </xs:element>
1147 <xs:element name="action" type="actionButtonType" minOccurs="0" maxOccurs="1">
1148 <xs:annotation>
1149 <xs:documentation>
1150 Describes how to create an action button to write into the data.

```

1155      </xs:documentation>
1155      </xs:annotation>
1155  </xs:element>
1155  <xs:element name="blob" type="blobType" minOccurs="0" maxOccurs="1">
1155      <xs:annotation>
1155          <xs:documentation>
1155              Describes a blob of data that can be read/written/both.
1155          </xs:documentation>
1155      </xs:annotation>
1155  </xs:element>

1160  <!--
1160      XML Schema 1.1 construct expressing extensibility promise
1160      <xs:any minOccurs="0" maxOccurs="1" processContents="lax">
1165          <xs:assert test="every $x in * satisfies
1165              (exists($x/@size) and $x/@size castable to xs:integer)" />
1165          <xs:assert test="every $x in * satisfies
1165              (exists($x/@offset) and $x/@offset castable to xs:integer)" />
1165      <xs:annotation>
1165          <xs:documentation>
1165              Extension point for future schema
1165          </xs:documentation>
1165      </xs:annotation>
1165  </xs:any>
1170  -->
1170      </xs:choice>
1170      </xs:sequence>
1170  <xs:attribute name="space" type="xs:int" use="required">
1175      <xs:annotation>
1175          <xs:documentation>
1175              The decimal number of the address space where the information is found.
1175          </xs:documentation>
1175      </xs:annotation>
1175  </xs:attribute>
1180      <xs:attribute name="origin" type="xs:int" default="0">
1185          <xs:annotation>
1185              <xs:documentation>
1185                  Starting address of the segment's contents
1185                  within the memory space.
1185              </xs:documentation>
1185          </xs:annotation>
1185  </xs:attribute>
1190  </xs:complexType>
1195  </xs:element>
1195  </xs:sequence>

```

[`</xs:complexType>`](#)
[`</xs:element>`](#)
[`</xs:schema>`](#)



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	<segment> 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