Checking the OpenLCB Configuration Description Information Standard

February 26, 2024

1 Introduction

This note documents the procedure for checking an OpenLCB implementation against the Configuration Description Information (CDI) Standard.

The checks are traceable to specific sections of the Standard.

The checking assumes that the Device Being Checked (DBC) is being exercised by other nodes on the message network, e.g. is responding to enquiries from other parts of the message network. .

2 Configuration Description Information Procedure

This plan assumes that the Datagram Transport Protocol and the Memory Configuration Protocol have been separately checked. It uses those, but does not do any detailed checking of them.

2.1 CDI Memory Present checking

This section checks that the CDI memory defined in section 4.2 of the Memory Configuration Protocol Standard is present.

A node which does not self-identify in PIP that it supports Configuration Description Information passes this check.

The Get Address Space Information Command from the Memory Configuration Protocol is used to validate that space 0xFF is present and read-only.

2.2 Validation checking

This section checks the content of the CDI against the XML Schema defined in Section 5 of the Standard to make sure that it is valid XML.

A node which does not self-identify in PIP that it supports Configuration Description Information passes this check.

A node which has the CDI bit set in PIP, but does has the Memory Configuration bit reset in PIP, fails this check.

This check reads the information from the 0xFF memory space. If the space is not available, or there are no bytes of content in it, the check fails.

The check then validates the content against the 1.3 XML Schema which is stored in a local "schema.xsd" file.

2.3 ACDI checking

This checks against the statements in section 5.1.2 of the Standard.

Depending on the bits set in the PIP information, there are several possible configurations to check. Each of these is checked in turn, if relevant.

1. ACDI bit set and Memory Configuration bit not set

   This configuration fails the check. PIP ACDI set implies that certain information is available in the configuration memory.

2. ACDI bit set and Memory Configuration bit set

   The 251 (0xFB) and 252 (0xFC) memory spaces are checked for being present in the Memory Configuration Protocol.

   The two version numbers in the 251 (0xFB) and 252 (0xFC) memory spaces are read and checked against their required values.

   The six ACDI-defined strings are read from their respective memory spaces to make sure that can be done without error.

3. SNIP bit, ACDI bit and Memory Configuration bit all set

   The SNIP data is acquired.

   The six SNIP strings are checked for equality, up to the first zero byte, to the strings acquired from memory in the step above.

4. Memory Configuration bit and CDI bit set

The check reads the first 820 bytes of the CDI information in space 255 (0xFF) to determine if an `<acdi>` element is present [1].

If the `<acdi>` element is present and the ACDI bit is not set in PIP, or if the `<acdi>` element is not present and the ACDI bit is set in PIP, the check fails.

2.4 Identification Element checking

This checks against the statements in section 5.1.1 of the Standard.

A node which does not self-identify in PIP that it supports Configuration Description Information passes this check.

A node which has the CDI bit not set in PIP passes this check.

The check reads the first 320 bytes of the CDI information in space 255 (0xFF) to determine if an `<identification>` element is present [2].

If the `<identification>` element is not present, this check pases.

If the SNIP bit is set in PIP, the contents of the `<identification>` sub-elements are checked against the SNIP contents up to the first zero byte.

If the `<acdi>` element is present, the contents of the `<identification>` sub-elements are checked against the ACDI-defined contents of the 251 (0xFB) memory space up to the first zero byte.

[1] The proper placement of the `<acdi>` element was checked in the Validation checking step above

[2] The proper placement and contents of the `<identification>` element was checked in the Validation checking step above