OpenLCB Standard	
Abbreviated Configuration Description Information	
Jun 15, 2012	Preliminary

1 Introduction (Informative)

(nothing yet)

2 Intended Use (Informative)

Provides quick access to basic information on a node, so that a system can get quick summaries of all the nodes present without spending the time to do a full CDI read and then configuration read.

3 References and Context (Normative)

For more information on format and presentation, see:

- OpenLCB Common Information Technical Note

There are separate documents that discuss (a) Configuration Protocol and the full mechanisms for Configuration Description Information. In this document, we discuss an abbreviated form of configuration description information that is useful for rapid retrieval of basic information from an OpenLCB node.

4 Memory space configuration (Normative)

This describes version 1 of the data format.

If PIP says memory access configuration protocol is present, and that this protocol is present, then the data must be available in these places.

4.1 Constant data

Read from space 0xFC

Byte 0: ACDI version, default 1

Next: Manufacturer name as null-terminated string

Next: Manufacturer-issued node type name as null-terminated string

Next: Manufacturer-defined node hardware version, as null-terminated string

Next: Manufacturer-defined node software version, as null-terminated string

(size is up to manufacturer, as it's the whole space, recommended that it fits into a 64-byte read)

4.2 User-modifiable data

Read from space 0xFB

Byte 0: ACDI version, default 1

Next: User provided node name as null-terminated string (20 byte limit, including null)

Next: User provided free-form comment as null-terminated string (40 byte limit, including null)

4.3 Icon data

Space 0xFA

As a PNG sequence (need more info on PNG options?)

5 Retrieval Protocol (Normative)

Upon receipt of an addressed messages with a SNII Request MTI, the node returns a single message with a SNII Reply MTI carrying the information of the two types listed above, concatenated.

On CAN, this is done via the usual mechanism of concatenated frames, with a short or empty one indicating end.

Name	Dest ID	Event ID	Simple Node	Common MTI	CAN format	Data Content
SNI Request	Y	N	N	0x3520	0x1Edd,dsss 52	(none)

Name	Dest ID	Event ID	Simple Node	Common MTI	CAN format	Data Content
SNI Reply	Y	N	N	0x3530	0x1Edd,dsss 53	Up to 7 bytes

6 Open Questions

We're providing two retrieval methods: Via the 0x3520/0x3530 MTIs, and via configuration read from specific spaces. Are both needed? Is that overly complex? Should PIP show them separately? JMRI really likes the Request/Reply, because it's really fast: One request queues up retrieval of the entire thing, without having to deal with datagrams. JMRI is a big boy and can handle the large reply volume.

But the config read mechanism (via individual datagrams) might be better for small nodes making this request. Hence the desire for the two mechanisms.

Should length limits on the manufacturer strings be provided? Total length?

Make clear that it's multiple frames on CAN, and can be either multiple or a single message on other protocols. How do you know where the end is if it's multiple messages or CAN frames messages? You don't. You just accept & process the data as it arrives.

Table of Contents

