Checking the OpenLCB CAN Frame Level Protocols

October 5, 2025

1 Introduction

This note documents the procedure for checking an OpenLCB implementation against the CAN Frame Transfer 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 Frame Level Procedure

2.1 Initialization

This section's checks cover Frame Transfer Standard sections 4, 6.1 and section 6.2.1.

The checks assume that the node reserves a single alias at startup.

Follow the prompts when asked to reset or otherwise initialize the DBC.

The checker waits up to 30 seconds for the node to restart and go through a node reservation sequence.

1. All frames carry the same source alias
2. The sequence of four CID frames, a RID frame, and AMD frame are sent
3. The Node ID in the four CID frames matches the Node ID in the AMD frame
4. The CID frames and RID frame have no data content
5. That the AMD Node ID matches that of the node being checked

6. Neither the alias[1] nor the Node ID[2] is zero.

2.2 AME Sequences

This section's checks cover Frame Transfer Standard sections 4, 6.1 and section 6.2.3.

The checks assume that the node has previously reserved at least one alias and is in the Permitted state.

The checker sends an AME frame with no NodeID and checks for:

1. An AMD frame in response
2. That carries the Node ID of the DBC

The checker sends an AME frame with the Node ID of the DBC and checks the response for:

1. An AMD frame in response
2. That carries the Node ID of the DBC

The checker sends an AME frame with a Node ID different from the Node ID of the DBC and checks for no response.

2.3 Alias Conflict

This section's checks cover Frame Transfer Standard sections 4, 6.1 and section 6.2.5.

The checks assume that the node has previously reserved at least one alias and is in the Permitted state.

The checker sends an AME frame to acquire the DBC's current alias from the AMD response.

The checker sends an CID frame with the DBC's alias and checks for

1. An RID frame in response
2. That carries the alias of the DBC.

The checker sends an AMD frame with the DBC's alias and checks for

1. An AMR frame in response
2. That carries the source alias of the DBC.

[1] See section 6.3 of the Standard
[2] See section 5.12 of the Unique Identifiers Standard.

At this point, Frame Transfer Standard section 6.2.5 specifies that the node must stop using that alias. Many nodes will reserve a different one at this point.

If an alias reservation sequence does start, the first frame will be checked for a proper CID frame. In addition, the checker will check that the newly reserved alias is different from the original one.

Having reserved an alias, many nodes will put it into use with an AMD frame. If one is received, it will be checked to see if it contains the new alias and the DBC's node ID.

Once traffic on the bus has paused for one second, indicating the completion of any allocation(s), the checker will emit an AME frame.

1. If an AMD has been received, there should be exactly one reply with the DBC's node ID. It should carry the new alias.
2. If an AMD has not been received, there should be no replies with the DBC's node ID or with the new alias.

Note: The Standards allow other initialization sequences, and this check is not completely general. If a DBC doesn't pass, the sequence should be carefully examined in light of the CAN Frame Transfer Standard to determine if it's just an issue with the check logic.

2.4 Reserved Frame Bit

This section's checks cover Frame Transfer Standard sections 4, 6.1 and section 6.2.3., specifically that the 0x1000_0000 bit in the CAN header is properly ignored.

The checker sends an AME frame with zero in the 0x1000_0000 bit and with no NodeID and checks for:

1. An AMD frame in response,
2. That carries the Node ID of the DBC,
3. With the 0x1000_0000 bit set to one.

2.5 Capacity Check

It is an unspoken assumption of the Standard that a node will receive and process all CAN frames, even when the CAN bus is 100% busy. (Though see Section 4 of the Technical Note for discussion on this.)

This check uses Message Network-level messages ant Event Transport messages.

To confirm that, the checker:

1. Sends 300 PCER messages with an Event ID of 00.00.00.00.00.00.00.01, followed by a Verify Node ID Number Addressed message to the device being checked, followed by 300 PCER messages with an Event ID of 00.00.00.00.00.00.00.01. All 601 messages are sent at the full CAN rate, with no inter-frame gaps.
2. It then looks for whether a Verified Node ID Number message was transmitted in response to the Verify Node ID Number Addressed message. If not, the check fails.
3. It then sends 300 Verify Node ID Number Addressed messages addressed to another (not-present) node, followed by a Verify Node ID Number Global message, then 300 Verify Node ID Number Addressed messages addressed to another (not-present) node. All 601 messages are sent at the full CAN rate, with no inter-frame gaps.
4. It then looks for whether a Verified Node ID Number message was transmitted in response to the Verify Node ID Number Global message. If not, the check fails.
5. It then sends 300 Identify Consumer messages with an Event ID of 00.00.00.00.00.00.00.01 followed by a global Verify Nodes message, then 300 Identify Consumer messages with an Event ID of 00.00.00.00.00.00.00.01. All 601 messages are sent at the full CAN rate, with no inter-frame gaps.
6. It then looks for whether a Verified Node ID Number message was transmitted in response to the global Verify Nodes message. If not, the check fails.

2.6 Compatibility with Standard CAN frames

This checks the compatibility with standard CAN frames defined in section 4 of the standard.

The checker sends each valid standard CAN frame header, 2048 values, sequentially. As it does this, it checks for a response from the node being checked. Any response fails the check.