**Standard Component Conformance Test Architecture**

This document describes the architecture used to construct standard component tests for OpenMax IL 1.1. Due to deliberate code re-use, this construction should lend itself to rapid development of new tests.

Note that, aside from the addition of tests to the structure that lists the tests, no changes to the test harness (from the one used for 1.0) were necessary to accommodate these new tests.

**Files**

* *OMX\_CONF\_StdCompCommon.h*: Functions, structures, and macros common to all standard component tests.
* *OMX\_CONF\_StdAudioCapturerTest.c*: All audio capturer class tests.
* *OMX\_CONF\_StdAudioDecoderTest.c*: All audio decoder class tests.
* *OMX\_CONF\_StdBinaryReaderTest.c*: All video, image, and audio binary reader component tests.
* *OMX\_CONF\_StdBinaryWriterTest.c*: All video, image, and audio binary writer component tests.
* *OMX\_CONF\_StdCompCommon.c*: Routines for testing attributes common to all ports and the generalized standard component test function.
* *OMX\_CONF\_StdCompCommonAudio.c*: Routines for testing attributes common:
  + to an audio port (irrespective of audio format)
  + to port of a specific audio format (e.g. MP3)
* *OMX\_CONF\_StdCompCommonOther.c*: Routines for testing attributes common:
  + to an other port (irrespective of “other” format)
  + to port of a specific “other” format (e.g. time)
* *OMX\_CONF\_StdCompCommonVideo.c*: Routines for testing attributes common:
  + to an video port (irrespective of video format)
  + to port of a specific video type (e.g. AVC)
* *OMX\_CONF\_StdVideoDecoderTest.c*: All video decoder class tests.
* *OMX\_CONF\_StdVideoEncoderTest.c*: All video encoder class tests.

**Test Architecture**

There are two types of test entrypoints (e.g. a test that may be designated on the conformance test harness command line):

1. *Standard component-level entrypoint*: an entrypoint that, given a component under test (CUT), confirms conformance to a particular standard component definition (e.g. audio\_decoder.mp3). Example: OMX\_CONF\_StdMp3DecoderTest
2. *Standard class-level entrypoint*: an entrypoint that, given a CUT, iterates through all component aliases (e.g. mp3, amrwb, aac, etc) for a given class (e.g. audio\_decoder) on that CUT. If the alias is exposed by the CUT the test confirms compliance for that alias (leveraging the standard component-level entrypoints). Example: OMX\_CONF\_StdAudioDecoderTest which leverages OMX\_CONF\_StdMp3DecoderTest, OMX\_CONF\_StdAacDecoderTest, etc.

Each Standard component-level entrypoint leverages a common function called StdComponentTest\_StdComp that:

1. Sets up the component to the point where port parameters/configs may be manipulated. These operations are generic to all components and standard component tests.
2. Performs set/get parameter/config call specific to the entrypoint via a callback to StdComponentTest\_StdComp. Example callback: StdAudioDecoderTest\_Mp3Decoder
3. Performs clean-up on the component. These operations are generic to all components and component tests.

The callback used in step 2 above consists of a set of parameter/config checks for each mandated port of the component. Thus an mp3 decoder test would include check for an mp3 type input port (supporting the relevant mp3 parameters/configs at the mandated port index) and a pcm type output port (supporting the relevant pcm parameters/configs at the mandated port index).

The test architecture exploits the fact that, for instance, the characteristics of an “mp3 port” are the same regardless of which standard component it occurs on. Thus the test architecture includes subroutines verify ports of specific types that are re-used throughout the tests. Examples: StdCompCommonAudio\_Mp3PortParameters used by both the mp3 decoder and mp3 encoder tests, StdCompCommonVideo\_AvcPortParameters used by both the AVC decoder and AVC encoder tests, etc.

**Adding a new test**

1. If not already present, add a new standard class-level entrypoint that iterates through all CUT aliases and calls the test associated with the alias. Example: OMX\_CONF\_StdAudioPostProcessorTest
2. Create a new standard component-level entrypoint and add a call to it in the class-level entrypoint. Example: OMX\_CONF\_StdAudioPostProcessorReverbTest.
3. Create a callback to test for all the necessary ports. From the component level entrypoint call StdComponentTest\_StdComp with this callback as parameter. Example: Create StdAudioPostProcessorTest\_Reverb which might leverage StdCompCommonAudio\_PcmPortParameters to verify that the component’s audio input and audio output ports support the right pcm parameters and configs. Inside OMX\_CONF\_StdAudioPostProcessorReverbTest call StdComponentTest\_StdComp with StdAudioPostProcessorTest\_Reverb as a parameter.