NAME

cec-compliance - An application to verify remote CEC devices

SYNOPSIS

cec-compliance [*-h*] [*-d <dev>*] [other options]

DESCRIPTION

The cec-compliance utility can be used to test how well remote CEC devices comply with the CEC specification. It can also be used to test the local CEC adapter (with the *-A* option).

By default it will run through all tests, but if one or more of the feature test options is given, then only those tests will be performed. A set of core tests is always run.

The CEC adapter needs to be configured before it is used to run tests with **cec-compliance**. Use **cec-ctl** for configuration.

If the CEC adapter has claimed several logical addresses, the test set is run from each logical address in succession. The remote device needs to report a valid physical address in order to run tests on it.

When running compliance tests, **cec-follower** should be run on the same adapter. **cec-follower** will reply to messages that are not handled by **cec-compliance**. **cec-follower** will also monitor the device under test for behaviors that are not compliant with the specification. Before each test-run **cec-follower** should be restarted if it is already running, to initialize the emulated device with a clean and known initial state.

Some tests require interactive mode (with the *-i* option) to confirm that the test passed. When in interactive mode, the user is asked to observe or perform actions on the remote device. Some tests also give conclusive test results when run in interactive mode.

When testing the local CEC adapter's compliance with the CEC API, there must be at least one remote device present in order to test transmitting and receiving.

The compliance tests can have several possible outcomes besides passing and failing:

OK The test passed.

OK (Unexpected) The test passed, but it was unexpected for the device under test to support it. This might for example occur when a TV replies to messages in the Deck Control feature.

OK (Not Supported) The feature that was tested is not supported by the device under test, and that feature was not mandatory for the device to pass.

OK (Presumed) Nothing went wrong during the test, but the test cannot positively verify that the required effects of the test occurred. The test runner should verify that the test passed by manually observing the device under test. This is typically the test result for tests that send messages that are not replied to, but which induce some side effect on the device under test, such as a TV switching to another input or sending a Remote Control command.

OK (Refused) The device supports the feature or message being tested, but responded <Feature Abort> ["Refused"] to indicate

that it cannot perform the given operation. This might for example occur when trying to test the One Touch Record feature on a TV with copy protection enabled.

FAIL The test failed and was expected to pass on the device.

OK (Expected Failure) Failed but this was expected. This can only happen if the **−−expect** option was used that specified that a particular test would return a FAIL result.

FAIL (Expected X, got Y) The test returned a different result than was expected. This can only happen if the **−−expect** option was used that specified that a particular test would return a specific non-FAIL result.

Some tests depend on other tests being successful. These are not run if the tests they depend on failed, and they will not be shown in the test listing.

OPTIONS

−d, **−−device** <*dev*>

Use device <dev> as the CEC device. If <dev> is a number, then /dev/cec<dev> is used.

−D, **−−driver** <*drv*>

Use a cec device that has driver name <*drv*>, as returned by the CEC_ADAP_G_CAPS ioctl. This option can be combined with **−a** to uniquely identify a CEC device without having to rely on the device node number.

−a, **−−adapter** <*adap-name*>

Use a cec device that has adapter name <*adap-name*>, as returned by the CEC_ADAP_G_CAPS ioctl. This option can be combined with**−D** to uniquely identify a CEC de vice without having to rely on the device node number.

−E, **−−exit−on−fail**

Exit this application when the first failure occurs instead of continuing with a possible inconsistent state.

−l, **−−list−tests**

List all tests and the possible test results. This is used by the **−−expect** option.

−e, **−−expect** <*test*>=<*result*>

−n, **−−expect-with-no-warnings** <*test*>=<*result*> Fail if the test gave a different result. The **−−list−tests** option lists all the possible tests and what result values can be used.

This can be used in test scripts to verify that a specific result was returned by the test. One use-case is to verify that an optional feature is actually supported, so an *OK* result instead of an *OK (Not Supported)* result is expected.

It can also be used to accept known failures. In that case the test will not fail, but return an *OK (Expected Failure)* result.

The **−−expect-with-no-warnings** variant is more strict and will also check that the test produced no warnings.

−v, **−−verbose**

Turn on verbose reporting.

−−version

Show version information.

−w, **−−wall−clock**
Show timestamps as wall-clock time.

−S, **−−show−timestamp**
For each test, show the timestamp of when the test started.

−T, **−−trace**
Trace all called ioctls. Useful for debugging.

−h, **−−help**
Prints the help message.

−W, **−−exit−on−warn**
Exit this application when the first warning occurs instead of continuing.

−s, **−−skip−info**
Skip the Driver Info output section.

−C, **−−color** <*when*>
Highlight OK/warn/fail/FAIL strings with colors. OK is marked green, warn is marked bold, and fail/FAIL are marked bright red if enabled. <*when*> can be *always*, *never*, or *auto* (the default).

−N, **−−no−warnings**
Turn off warning messages.

−r, **−−remote** <*la*>
As initiator test the remote logical address <la> or all LAs if no LA was given.

−i, **−−interactive**
Interactive mode when doing remote tests.

−R, **−−reply−threshold** <*timeout*>
Warn if replies take longer than this threshold (default 1000ms).

−t, **−−timeout** <*secs*>
Set the standby/resume timeout to the given number of seconds. Default is 60s.

−A, **−−test−adapter**
Test the CEC adapter API

−F, **−−test−fuzzing**
Test the remote CEC adapter by randomly creating CEC messages. This runs forever until an error occurs.

−−test−core
Test the core functionality

−−test−audio−rate−control
Test the Audio Rate Control feature

−−test−audio−return−channel−control
Test the Audio Return Channel Control feature

−−test−capability−discovery−and−control
Test the Capability Discovery and Control feature

−−test−deck−control
Test the Deck Control feature

−−test−device−menu−control
Test the Device Menu Control feature

−−test−device−osd−transfer
Test the Device OSD Transfer feature

−−test−dynamic−audio−lipsync
Test the Dynamic Audio Lipsync feature

−−test−osd−display
Test the OSD Display feature

−−test−one−touch−play
Test the One Touch Play feature

−−test−one−touch−record
Test the One Touch Record feature

−−test−power−status
Test the Power Status feature

−−test−remote−control−passthrough
Test the Remote Control Passthrough feature

−−test−routing−control
Test the Routing Control feature

−−test−system−audio−control
Test the System Audio Control feature

−−test−system−information
Test the System Information feature

−−test−timer−programming
Test the Timer Programming feature

−−test−tuner−control
Test the Tuner Control feature

−−test−vendor−specific−commands
Test the Vendor Specific Commands feature

−−test−standby−resume
Test standby and resume functionality. This will activate testing of Standby, Give Device Power Status and One Touch Play.

EXIT STATUS

On success, it returns 0. Otherwise, it will return the error code.

EXAMPLE

We want to test the compliance of a TV when it is interacting with a Playback device. The device node of the CEC adapter which the TV is connected to is /dev/cec1.

The local CEC adapter first needs to be configured as a Playback device, and it must have an appropriate physical address. It is important that the physical address is correct, so as to not confuse the device under test. For example, if the CEC adapter is connected to the first input of the TV, the physical address 1.0.0.0 should generally be used.

```
cec-ctl -d1 --playback --phys-addr 1.0.0.0
```

Most CEC adapters will automatically detect the physical address, and for those adapters the *--phys-addr* option is not needed.

Next, **cec-follower** also has to be started on the same device:

```
cec-follower -d1
```

cec-compliance can now be run towards the TV by supplying the *-r* option with the logical address 0:

cec-compliance -d1 -r0

BUGS

This manual page is a work in progress.

Bug reports or questions about this utility should be sent to the linux-media@vger.kernel.org mailinglist.

SEE ALSO

cec-follower(1), **cec-ctl**(1)